

# **Teradata Vantage™ - Advanced SQL Engine Security Administration**

---

Release 17.10

July 2021

# Copyright and Trademarks

Copyright © 2000 - 2021 by Teradata. All Rights Reserved.

All copyrights and trademarks used in Teradata documentation are the property of their respective owners. For more information, see [Trademark Information](#).

## Product Safety

Safety type	Description
	Indicates a situation which, if not avoided, could result in damage to property, such as to equipment or data, but not related to personal injury.
	Indicates a hazardous situation which, if not avoided, could result in minor or moderate personal injury.
	Indicates a hazardous situation which, if not avoided, could result in death or serious personal injury.

## Third-Party Materials

Non-Teradata (i.e., third-party) sites, documents or communications ("Third-party Materials") may be accessed or accessible (e.g., linked or posted) in or in connection with a Teradata site, document or communication. Such Third-party Materials are provided for your convenience only and do not imply any endorsement of any third party by Teradata or any endorsement of Teradata by such third party. Teradata is not responsible for the accuracy of any content contained within such Third-party Materials, which are provided on an "AS IS" basis by Teradata. Such third party is solely and directly responsible for its sites, documents and communications and any harm they may cause you or others.

## Warranty Disclaimer

**Except as may be provided in a separate written agreement with Teradata or required by applicable law, the information available from the Teradata Documentation website or contained in Teradata information products is provided on an "as-is" basis, without warranty of any kind, either express or implied, including the implied warranties of merchantability, fitness for a particular purpose, or noninfringement.**

The information available from the Teradata Documentation website or contained in Teradata information products may contain references or cross-references to features, functions, products, or services that are not announced or available in your country. Such references do not imply that Teradata Corporation intends to announce such features, functions, products, or services in your country. Please consult your local Teradata Corporation representative for those features, functions, products, or services available in your country.

The information available from the Teradata Documentation website or contained in Teradata information products may be changed or updated by Teradata at any time without notice. Teradata may also make changes in the products or services described in this information at any time without notice.

## Machine-Assisted Translation

Certain materials on this website have been translated using machine-assisted translation software/tools. Machine-assisted translations of any materials into languages other than English are intended solely as a convenience to the non-English-reading users and are not legally binding. Anybody relying on such information does so at his or her own risk. No automated translation is perfect nor is it intended to replace human translators. Teradata does not make any promises, assurances, or guarantees as to the accuracy of the machine-assisted translations provided. Teradata accepts no responsibility and shall not be liable for any damage or issues that may result from using such translations. Users are reminded to use the English contents.

## Feedback

To maintain the quality of our products and services, e-mail your comments on the accuracy, clarity, organization, and value of this document to: [docs@teradata.com](mailto:docs@teradata.com).

Any comments or materials (collectively referred to as "Feedback") sent to Teradata Corporation will be deemed nonconfidential. Without any payment or other obligation of any kind and without any restriction of any kind, Teradata and its affiliates are hereby free to (1) reproduce, distribute, provide access to, publish, transmit, publicly display, publicly perform, and create derivative works of, the Feedback, (2) use any ideas, concepts, know-how, and techniques contained in such Feedback for any purpose whatsoever, including developing, manufacturing, and marketing products and services incorporating the Feedback, and (3) authorize others to do any or all of the above.

# Contents

<b>Chapter 1: Introduction to Security Administration</b>	<b>10</b>
Changes and Additions	10
<b>Chapter 2: Teradata Generic Security Services</b>	<b>12</b>
Implementation Overview	12
TDGSS Configuration Files	12
Setting Up TDGSS	14
Modifying the User Configuration File	14
Security Administration Tools	15
Legacy TeraGSS	16
<b>Chapter 3: Setting Up the Administrative Infrastructure</b>	<b>17</b>
Implementation Process	17
About System-Generated Users	17
Working with System-Level Space Allocation	19
Working with Administrative Users	22
Setting Up the Database Administrator User	29
Working with the Common Criteria Standard	33
Avoiding Potential Security Hazards	36
Controlling Physical Access	36
Controlling Access to the Operating System	36
<b>Chapter 4: Implementing User Authentication and Authorization</b>	<b>38</b>
About Teradata Authentication	38
About External Authentication	38
About External Authentication Controls	41
About External Authentication Requirements	42
Using External Authentication from Mainframe Clients	44
About Authentication and Authorization of Middle-Tier Application Users	44
About Authentication of Users Logging On through Unity	45
Working with Kerberos Authentication	45
Installing and Configuring Kerberos	46
Working with Kerberos Setup on the KDC	50
Setting Up Kerberos for Teradata Vantage on a Windows Kerberos KDC	51
Setting Up Kerberos on a Linux MIT Kerberos KDC	58
Configuring Teradata Vantage and Unity Servers for Kerberos Authentication	61
<b>Chapter 5: Configuring Single Sign-On</b>	<b>75</b>
Single Sign-On Flow	75

Configuration for Browser Authentication .....	78
Configuration for OpenID Connect .....	79
Configuration of Static Keys .....	84
User Name Mappings .....	89
SSO Security Hardening .....	90
Related Information .....	91
<b>Chapter 6: Creating Users and Granting Privileges .....</b>	<b>92</b>
User Implementation Process .....	92
Prerequisites .....	92
About Database User Functional Categories .....	93
Assessing User Needs .....	93
Working with Database Profiles .....	94
About Assigning Profiles to Users .....	96
Using a Profile to Set a Default Query Band .....	97
Dropping Profiles .....	98
About Database User Types .....	98
Creating Permanent Database Users .....	99
Working with Directory Users .....	101
Working with Middle-Tier Application Users .....	103
About Teradata Vantage User Privileges .....	106
Working with User Privileges in Teradata Vantage .....	109
Using Roles to Manage Privileges .....	112
Working with Roles for Proxy Users .....	116
Using Roles for Directory Users .....	117
Switching Roles During a Session .....	119
Other Options for Restricting Database Access .....	119
<b>Chapter 7: Managing Database Passwords .....</b>	<b>122</b>
Prerequisites .....	122
Password Management Process .....	122
Working with Password Formatting .....	122
Format Rules for Object Naming .....	123
Managing Common Password Problems .....	125
Working with Password Controls .....	127
Setting Password Controls .....	141
<b>Chapter 8: Logging on to Teradata Vantage .....</b>	<b>145</b>
Prerequisites .....	145
Logon Implementation Process .....	145
About Default Logon Privileges .....	145
Using Logon Elements .....	146
About Network Logons .....	151
Logging on Using Teradata Authentication and Authorization .....	153
Logging on Using LDAP Authentication and Authorization .....	154

Logging on Using Sign-on As . . . . .	157
Logging on Using Single Sign-on with Kerberos . . . . .	161
Logging on Using Teradata Negotiating (TDNEGO) . . . . .	164
Using Teradata Wallet to Store and Retrieve Logon Elements . . . . .	171
Storing Logon Information in Teradata Wallet . . . . .	172
Retrieving Logon Information from Teradata Wallet Using BTEQ . . . . .	174
Retrieving a Password from Teradata Wallet Using ODBC Driver for Teradata . . . . .	175
Working with Logon Variations by Application . . . . .	175
Logging On from Mainframe Systems . . . . .	177
Logging On from Teradata Vantage Nodes . . . . .	177
Logging On through Unity . . . . .	178
Logging On from .NET Clients . . . . .	178
Using Operating System Logons . . . . .	178
Using Logon Error Handling Options . . . . .	178
<b>Chapter 9: Directory Management of Database Users . . . . .</b>	<b>180</b>
Directory Database User Implementation Process . . . . .	180
Working with Directory User Management Options . . . . .	181
About Directory User Characteristics . . . . .	195
<b>Chapter 10: Evaluating the System for Directory Management of Users . . . . .</b>	<b>199</b>
Evaluation Process . . . . .	199
About Certified Directories . . . . .	199
Checking the Network Setup . . . . .	200
Querying the Directory Server . . . . .	201
Common Errors with Active Directory, ADAM, or AD LDS . . . . .	206
Common Errors with Sun Java Directory Server . . . . .	210
<b>Chapter 11: Provisioning Directory Users with Teradata Schema Extensions . . . . .</b>	<b>213</b>
Provisioning Process . . . . .	213
About Teradata Schema Extensions . . . . .	214
Installing Teradata Schema Extensions in a Certified Directory . . . . .	218
About Teradata Schema Objects in the DIT . . . . .	224
Creating the Top-Level Objects in the DIT . . . . .	227
Creating Containers and Inserting Objects . . . . .	228
Mapping Directory Users to Teradata Vantage Objects . . . . .	233
<b>Chapter 12: Using Native Directory Schema to Provision Directory Users . . . . .</b>	<b>236</b>
Process Overview . . . . .	236
About Required Teradata Objects . . . . .	237
Creating the RootNode and System Objects . . . . .	237
Creating Containers and Inserting Database Objects . . . . .	239
Mapping Additional Directory Users to Vantage User, Role, and Profile Objects . . . . .	242
<b>Chapter 13: Changing the TDGSS Configuration . . . . .</b>	<b>244</b>

Prerequisites . . . . .	244
Configuration Change Process . . . . .	244
About the TDGSS Configuration Files . . . . .	244
Using the dumpcfg Utility to Check the Current Configuration . . . . .	246
About Editing Configuration Files . . . . .	248
Working with LDAP Mechanism Properties . . . . .	250
Making Changes to TdgssUserConfigFile.xml on Database Nodes . . . . .	252
Returning to an Old Configuration . . . . .	254
<b>Chapter 14: Testing Directory Authentication and Authorization Setup . . . . .</b>	<b>255</b>
Working with tdgsstestcfg . . . . .	255
Working with tdgssauth . . . . .	255
Working with tdsbind . . . . .	271
Using BTEQ to Verify Directory User Mapping . . . . .	280
Working with Ldapsearch . . . . .	281
<b>Chapter 15: LDAP Binding Options . . . . .</b>	<b>292</b>
Evaluation and Implementation Process . . . . .	292
Using DIGEST-MD5 Binds [Deprecated] . . . . .	292
Using Simple Binds . . . . .	292
Using Service Binds . . . . .	293
Using Anonymous Binds . . . . .	299
<b>Chapter 16: TLS Protection Options . . . . .</b>	<b>301</b>
Using TLS with a Directory Server . . . . .	301
Using TLS with Client to Database Connections . . . . .	323
<b>Chapter 17: Optimizing Directory Searches . . . . .</b>	<b>338</b>
About Search Optimization Options . . . . .	338
Configuring LDAP Properties to Narrow the Search Base . . . . .	338
Working with Directory User Identification Options . . . . .	339
Using Identity Mapping . . . . .	340
Using Identity Searches . . . . .	344
Using Multiple IdentityMap and IdentitySearch Elements in Combination . . . . .	347
<b>Chapter 18: Configuring LDAP for Site-Aware Authentication . . . . .</b>	<b>351</b>
Prerequisites . . . . .	351
Configuring Site Aware User Authentication in a Windows Domain . . . . .	351
Configuring Site Aware Authentication in a Global Catalog . . . . .	357
<b>Chapter 19: Configuring LDAP to Use Multiple Directory Services . . . . .</b>	<b>367</b>
Prerequisites . . . . .	367
Implementation Overview . . . . .	367
Adding Multiple Directory Services to the TDGSS Configuration . . . . .	368
Completing the <LdapConfig> Configuration Change . . . . .	372

Using <LdapConfig> with Unity .....	373
<b>Chapter 20: Network Security Policy .....</b>	<b>374</b>
Prerequisites .....	374
Implementation Overview .....	374
About Network Security Policy .....	374
Directory Schema Considerations .....	376
Standard Directory Entries and Security Policy .....	376
Using LDAP Directory Objects in Policies .....	378
Configuring Top-Level Security Policy Objects .....	380
Configuring a Security Mechanism Policy .....	382
About Confidentiality and Integrity QOP Policy .....	386
Configuring ipNetworks and Network Groups .....	388
Configuring an Integrity QOP Policy .....	395
Configuring a Confidentiality QOP Policy .....	400
Configuring Options Policies .....	406
Security Policies in the TDGSS Configuration .....	411
Configuring Security Policies in the TdgssUserConfigFile.xml .....	413
Configuring the Gateway to Allow Logons from Older Interfaces or Proxies .....	420
Requiring Confidentiality .....	422
Investigating Security Policy Assignments .....	423
Monitoring QOP Security Policy .....	428
<b>Chapter 21: Using Logon Controls .....</b>	<b>430</b>
Logon Control Implementation Options .....	430
About Logon Privileges .....	430
Granting and Revoking Logon Privileges .....	431
Controlling Logons through a Middle-Tier Application .....	432
Restricting Logons by Host Group .....	435
Restricting Logons by IP Address .....	436
Creating XML-Based IP Restrictions .....	438
Designing IP XML Restrictions .....	439
Creating an IP XML Restriction Document .....	456
Saving a Completed XML IP Restriction Document .....	463
Enabling XML-Based IP Restrictions with the ipxml2bin Utility .....	463
Testing XML-Based IP Restrictions .....	464
Creating IP Restrictions in a Directory .....	465
Designing Directory-Based IP Restrictions .....	466
Mapping IP Filters to Directory Users .....	469
Enabling Directory-Based IP Restrictions with the ipdir2bin Utility .....	469
Testing Directory-Based IP Restrictions .....	471
Editing or Disabling IP Restrictions .....	474
<b>Chapter 22: Encryption .....</b>	<b>476</b>
About Password Encryption .....	476

About Message Encryption . . . . .	476
Working with Quality of Protection Options . . . . .	477
Preparing to Change the QOP Configuration . . . . .	480
Changing the QOP Configuration . . . . .	480
Full Disk Encryption . . . . .	482
<b>Chapter 23: Monitoring Database Access . . . . .</b>	<b>484</b>
Access-Monitoring Implementation Process . . . . .	484
About Default Logging . . . . .	484
Working with Access Logging . . . . .	485
About DBC.AccLogTbl Entries . . . . .	487
Disabling Access Logging with the END LOGGING Statement . . . . .	488
Using Access Logging for Directory-Based Users . . . . .	488
Using Access Logging for Auto-Provisioned Users . . . . .	490
Using Access Logging for Proxy Users . . . . .	490
Access Logging for Viewpoint Users . . . . .	491
Sample Implementation of Access Logging . . . . .	492
Monitoring Query Band Logs . . . . .	493
Monitoring Security Policy Violations . . . . .	494
Using Network Encryption Auditing . . . . .	494
Auditing Logons by Clients that Cannot Automatically Follow Security Policy . . . . .	495
Using External Monitoring Software . . . . .	497
Investigating Database Access Attempts . . . . .	497
About Access Log Maintenance . . . . .	502
<b>Chapter 24: Implementing Row Level Security . . . . .</b>	<b>504</b>
About Row-Level Security . . . . .	504
Row-Level Security Implementation Process . . . . .	505
About Security Labels . . . . .	506
Defining Security Labels for Users and Rows . . . . .	507
Working with Row Level Security UDFs . . . . .	507
Working with Security Constraint Administrative Privileges . . . . .	510
Working with Security Constraints . . . . .	511
Working with Constraint Assignments . . . . .	514
Working with Security Constraint Columns . . . . .	518
Working with Constraint OVERRIDE Privileges . . . . .	520
Working with Row-Level Security Effects . . . . .	521
Determining the Session Constraint Values . . . . .	527
About Row-Level Security and Bulk Table Loads . . . . .	532
About Session Constraints and Bulk Table Loads . . . . .	532
Using Access Logging with Row Level Security . . . . .	532
About Constraint-Related System Tables and Views . . . . .	534
<b>Chapter 25: Implementing Teradata Secure Zones . . . . .</b>	<b>538</b>
Teradata Secure Zones Overview . . . . .	538



Secure Zone Objects .....	538
Secure Zone User Types .....	539
Privileges in Teradata Secure Zones .....	540
Implementing Teradata Secure Zones .....	541
Example Scenario .....	544
Setting Up a Zone for Testing .....	546
Security Considerations .....	547
Using Logging to Monitor Zones .....	548
<b>Appendix A: TDGSS Configuration Files, Valid Settings, and Editing Guidelines .....</b>	<b>550</b>
<b>Appendix B: Diagnostic Tools .....</b>	<b>638</b>
<b>Appendix C: Privilege Dictionary .....</b>	<b>672</b>
<b>Appendix D: Teradata GSS Administrative Package .....</b>	<b>695</b>
<b>Appendix E: Password Restricted Words .....</b>	<b>697</b>
<b>Appendix F: Additional Information .....</b>	<b>715</b>

# Introduction to Security Administration

Teradata Vantage™ is our flagship analytic platform offering, which evolved from our industry-leading Teradata® Database. Until references in content are updated to reflect this change, the term Teradata Database is synonymous with Teradata Vantage.

Advanced SQL Engine is a core capability of Teradata Vantage, based on our best-in-class Teradata Database. Advanced SQL refers to the ability to run advanced analytic functions beyond that of standard SQL.

The purpose of this document is to help:

- Describe the available Teradata Vantage system security features.
- Define and implement a site-specific authentication, authorization, and security policy strategy.
- Manage users and their database privileges.
- Monitor security events and take corrective action.
- Prevent unauthorized users from gaining access to the system.

## Changes and Additions

Date	Description
July 2021	<ul style="list-style-type: none"> <li>• TLSv1.2 is supported between clients and the database server. See <a href="#">Using TLS with Client to Database Connections</a>.</li> <li>• Single Sign-On and JWT:               <ul style="list-style-type: none"> <li>◦ See the new information in <a href="#">Configuring Single Sign-On</a>.</li> <li>◦ JWT dynamically updates JSON Web Keys, if enabled. See <a href="#">Local Validation</a>.</li> <li>◦ JWT supports logons from third-party applications with a token from an external Identity Provider, see <a href="#">Validation by Token Exchanger</a>.</li> <li>◦ For new JWT mechanism properties, see <a href="#">JWT Support Properties</a>.</li> </ul> </li> <li>• Previously, when the TDGSS configuration changed, a TPA reset was required for the new values in the TDGSSCONFIG GDO to take effect. Now, the following can be modified without a TPA reset:               <ul style="list-style-type: none"> <li>◦ Any attribute or property whose name begins with "Ldap" for KRB5 and LDAP</li> <li>◦ MechanismEnabled property for KRB5, LDAP, JWT, and PROXY</li> <li>◦ AuthorizationSupported property for KRB5 and LDAP</li> <li>◦ LDAP Service ID and password with no impact to user LDAP logons</li> <li>◦ The following properties in the PROXY mechanism: CertificateFile, PrivateKeyFile, PrivateKeyPassword, PrivateKeypasswordProtected, CACertFile, CACertDir, and SigningHashAlgorithm.</li> <li>◦ Any JWT mechanism property whose name begins with "JWT"</li> <li>◦ All canonicalizations including the lightweight authorization structures</li> </ul> </li> </ul> <p>The following configuration changes still require a tpareset:</p> <ul style="list-style-type: none"> <li>◦ Changes to any mechanism property not mentioned above require a tpareset</li> <li>◦ QoP configuration</li> </ul>

Date	Description
	<ul style="list-style-type: none"> <li>◦ Local or global policy configuration, including service name changes</li> <li>◦ TDNEGO and SPNEGO See <a href="#">Modifying the User Configuration File</a>.</li> <li>• tdgsstestcfg is a new tool to test configuration changes before making them permanent with run_tdgssconfig, see <a href="#">Working with tdgsstestcfg</a>.</li> <li>• tdsbind is deprecated. Teradata recommends using the tdgssauth tool instead of tdsbind. tdgssauth can test more security mechanisms than tdsbind and it more accurately validates security mechanism configurations because it uses actual TDGSS services while performing the offline test of the new configuration. See <a href="#">Working with tdgssauth</a>.</li> <li>• tdgssgetinfo is a new diagnostic tool that collects and displays information used to determine the health of the TDGSS or TeraGSS installed on the system. See <a href="#">tdgssgetinfo</a>.</li> <li>• See <a href="#">X.509 Certificates Ownership and Permissions</a> for the recommended ownership and permissions for X.509 certificates and private key files.</li> </ul>
June 2020	<ul style="list-style-type: none"> <li>• The SASL/DIGEST-MD5 authentication protocol used by LDAP is deprecated. Teradata strongly recommends you stop using SASL/DIGEST-MD5, and instead use simple binding with TLS protection.</li> <li>• TDNEGO now supports JWT (JSON web token) authentication.</li> <li>• New LDAP Mechanism property: LdapServicePasswordFile. Allows you to provide an encrypted list of passwords in an editable file, which enables switching LDAP passwords without requiring a database restart.</li> </ul>

# Teradata Generic Security Services

Teradata Generic Security Services (TDGSS) is the software subsystem that manages Teradata Vantage Advanced SQL Engine network security. TDGSS is directly involved in:

- User authentication and authorization
- Encryption of network traffic
- Enforcement of security policy

TDGSS simplifies security management by providing:

- Configurable authentication mechanisms
- The option to provision and manage users in an LDAPv3-compliant directory
- A set of tools for configuring and administering security functions

TDGSS is an extension of the industry-standard GSS-API, as defined by Internet Engineering Task Force (IETF) Request for Comments (RFC) 2743, "Generic Security Service Application Program Interface Version 2, Update 1."

## Implementation Overview

- Only a TDGSS package is required on Advanced SQL Engine nodes and Unity. Both the SQL Engine and Unity require Teradata Tools and Utilities (TTU) support and the client-side component of TDGSS is embedded in all client drivers.
- TDGSS is fully functional without further set up or configuration, if the default Teradata authentication mechanism (TD2) is used. For authentication and authorization requirements options, see [Implementing User Authentication and Authorization](#).
- Review the external (non-Teradata) authentication and authorization options in [Implementing User Authentication and Authorization](#). If you use any of the external authentication options, you may need to complete additional configuration tasks.
- Review the explanation of the TDGSS configuration files to ensure that you understand how they function. See [TDGSS Configuration Files](#).
- Optionally use tools included with TDGSS for various security administration tasks. For a list of security-related tools, see [Security Administration Tools](#).

---

### Note:

For normal operation, there is no configuration requirement for the client side component of TDGSS. All configuration steps apply only to Unity and the SQL Engine.

---

## TDGSS Configuration Files

The TDGSS configuration files contain the settings that control external authentication and authorization, network security policy, and quality of protection when encryption is used. These configuration files include the following:

- TDGSS library configuration file
- TDGSS user configuration file

The configuration files are in XML format. You can edit the user configuration file to customize the configuration for your site. For more information, see [Changing the TDGSS Configuration](#).

## User and Library Configuration File Content

TDGSS Configuration File Type	Description
Library file	<p>Contains the default TDGSS configuration for the current Advanced SQL Engine release. The TdgssLibraryConfigFile.xml, defines the security elements, properties, and values available on the system.</p> <p>The TdgssLibraryConfigFile.xml file is located in the /opt/teradata/tdgss/etc directory on Unity and SQL Engine nodes.</p> <p><b>Note:</b> Do not edit the library configuration file.</p>
User file	<p>Contains the system-specific TDGSS configuration, including any user edits. Settings in this file override those in the library configuration file.</p> <p>The TdgssUserConfigFile.xml file is located in the /opt/teradata/tdgss/etc directory on Unity and SQL Engine nodes. Additionally, Unity (if used) copies the TdgssUserConfigFile.xml file to TdgssUnityConfig.xml in /etc/opt/teradata/config/unity and uses that file for its TDGSS configuration file.</p> <p>Editing of this file is permitted, but not required. You can edit the user configuration file to:</p> <ul style="list-style-type: none"> <li>• Control availability of mechanisms to all clients by setting the MechanismEnabled property</li> <li>• Control the source for authorization of user privileges for externally authenticated users by setting the AuthorizationSupported property</li> <li>• Customize LDAP properties, as required for directory user authentication and authorization</li> <li>• Change mechanism property values</li> <li>• Add new mechanisms and properties</li> </ul> <p>Property value settings must be the same for all nodes.</p> <p><b>Note:</b> If you are using Unity you must set up the PROXY mechanism on both Unity servers and the SQL Engine nodes.</p> <p>An upgrade to a new TDGSS release does not overwrite the user configuration file to allow customizations to be retained.</p> <p>For more information, see <a href="#">Changing the TDGSS Configuration</a>.</p>

## Setting Up TDGSS

No special setup is required if you use the default Teradata authentication.

Additional setup of TDGSS configuration files may be required on SQL Engine Nodes and on Unity, depending on the features you implement.

### TDGSS Channel Binding

TDGSS channel binding enables binding TDGSS authentication mechanisms to secure channels at lower network layers for those mechanisms that support channel binding, such as the PROXY mechanism. Channel binding verifies the endpoints of the lower level network layers to eliminate man-in-the-middle attacks. In the case of the PROXY mechanism, channel binding also makes it more difficult to use stolen certificates to pretend to be a legitimate endpoint. For increased security between Unity and the gateway, enable channel bindings on the managed Vantage servers. Use Gateway Control to enable channel binding:

```
gtwcontrol -j yes
```

The domain component (DC) of a Unity certificate for the PROXY mechanism is required to contain the name of the machine where Unity runs, but the name is not validated by default. Channel bindings can be used to validate and enforce that one of the IP addresses for the host name actually matches the IP address of the peer.

---

#### Note:

Enabling this option may impact logon performance due to host name resolution, depending on the customer host name lookup configuration.

---

For information on Gateway Control, see *Teradata Vantage™ - Database Utilities*, B035-1102.

## Modifying the User Configuration File

Globally Distributed Objects, (GDOs) are named objects that are kept consistent across all nodes and vprocs in a Teradata Database system. GDOs store system settings and configuration information that is shared by all nodes of the system.

The TPA Reset utility, `tpareset`, resets Teradata Database. `tpareset` is used when certain values in the `TdgssUserConfigFile.xml` configuration file are modified.

For many TDGSS configuration changes a TPA reset is not required for the new values in the TDGSSCONFIG GDO to take effect. The following can be modified without a TPA reset:

- Any attribute or property whose name begins with "Ldap" for KRB5 and LDAP
- MechanismEnabled property for KRB5, LDAP, JWT, and PROXY
- AuthorizationSupported property for KRB5 and LDAP
- LDAP Service ID and password with no impact to user LDAP logons

- The following properties in the PROXY mechanism:
  - CertificateFile
  - PrivateKeyFile
  - PrivateKeyPassword
  - PrivateKeypasswordProtected
  - CACertFile
  - CACertDir
  - SigningHashAlgorithm
- Any JWT mechanism property whose name begins with "JWT"
- All canonicalizations including the lightweight authorization structures

The following configuration changes require a tpareset:

- Changes to any mechanism property not mentioned above require a tpareset
- QoP configuration
- Local or global policy configuration, including service name changes
- TDNEGO and SPNEGO

run\_tdgssconfig is executed when the configuration file is updated. It indicates if a TPA reset is required.

## Security Administration Tools

The following security administration tools are included with the installation of TDGSS.

Tool	Description
dumpcfg	Allows you to view the TDGSS or TeraGSS configuration settings. These settings are stored in tdgssconfig.gdo, a binary-format <i>globally distributed object</i> file used by the database, or a flat file named tdgssconfig.bin for Unity or TTU clients. See <a href="#">Using the dumpcfg Utility to Check the Current Configuration</a> .
ipdir2bin	Adds directory-based IP restrictions to the IP GDO. See <a href="#">Enabling Directory-Based IP Restrictions with the ipdir2bin Utility</a> .
ipxml2bin	Adds XML based IP restrictions to the IP GDO. See <a href="#">Enabling XML-Based IP Restrictions with the ipxml2bin Utility</a> .
ldapadd	Standard LDAP tool used to add objects to the directory. See the sections beginning with <a href="#">Creating the Top-Level Objects in the DIT</a> .
ldapmodify	Used when adding Teradata schema extensions to a directory. See <a href="#">Installing Teradata Schema Extensions in a Certified Directory</a> .
ldapsearch	Used when testing directory access to find directory objects, such as a user or the RootDSE Object. See <a href="#">About Ldapsearch</a> .

Tool	Description
nodenames	Obtains the list of host names that are used when generating signed certificates. Used by and with tlsutil. See <a href="#">About nodenames</a> .
run_tdgssconfig	Required by Unity to enable edits to TdgssUnityConfig.xml, for example, when you add a new mechanism or configure a mechanism property. See <a href="#">Making Changes to TdgssUserConfigFile.xml on Database Nodes</a> . Note that TdgssUnityConfig.xml has the exact same format as TdgssUserConfigFile.xml, but is used specifically for Unity configuration.
tdgssauth	Test and verify that security mechanism configurations are valid before bringing them live. You can use it with LDAP, Kerberos, and TDNEGO on Unity servers and Advanced SQL Engine nodes. See <a href="#">Working with tdgssauth</a> .
tdgssgetinfo	Collects and displays information used to determine the health of the TDGSS or TeraGSS installed on the system. See <a href="#">tdgssgetinfo</a> .
tdsbind	Deprecated. <b>Note:</b> Teradata recommends using tdgssauth instead of tdsbind.
tdspolicy	Identifies security policy restrictions that apply to a specified user, profile, and IP address. See <a href="#">Investigating Security Policy Assignments</a> . <b>Note:</b> tdgssauth can be used instead of tdspolicy.
tdspasswd	Generates and stores passwords in encrypted form: <ul style="list-style-type: none"> <li>When configuring LdapServicePassword, for example, when creating a service bind. See <a href="#">Using Service Binds</a>.</li> <li>For changing a user password.</li> </ul>
tdgsstestcfg	Tests that TdgssUserConfigFile.xml changes are valid before making them permanent with run_tdgssconfig. See <a href="#">Working with tdgsstestcfg</a> .
tlsutil	Creates and installs signed certificates and private keys on SQL Engine. Used for TLS configuration. See <a href="#">About tlsutil</a> .

## Legacy TeraGSS

The old TeraGSS package was the counterpart of TDGSS in the client interfaces. Starting with TTU 16.10 there is no longer a separate TeraGSS package. Instead, the TeraGSS logic is embedded in all client drivers.

Configuration of TeraGSS is normally not required so we no longer deliver configuration files and tools. However, if configuration is needed, an optional Teradata GSS Administrative Package may be installed on the client and configuration may be performed. Note, Teradata recommends not configuring the clients.

For more information, see [Teradata GSS Administrative Package](#).



# Setting Up the Administrative Infrastructure

## Implementation Process

1. Review the description of system-generated users. See [About System-Generated Users](#).
2. Allocate sufficient space to system-generated users to perform system-level functions.
  - a. [Creating the Spool Space Reserve](#)
  - b. [Assigning Perm Space to the Crashdumps Database](#)
3. Create the principal administrative users, assign space, and grant administrative privileges.
  - a. [Creating the Security Administrator User](#)
  - b. [Setting Up the Database Administrator User](#)
4. Review the following topics for additional suggestions on secure database management.
  - [Avoiding Potential Security Hazards](#)
  - [Controlling Physical Access](#)
  - [Controlling Access to the Operating System](#)

## About System-Generated Users

The database automatically creates system-level database users during initial system setup:

- User DBC (the master administrative user) performs the initial steps for database setup and maintains the data dictionary tables.
- PUBLIC, the default user identity, has a set of privileges that are available to all users, primarily the SELECT privilege on most Data Dictionary views.
- Other system-generated users, such as SystemFE, perform automated system-level tasks in the database. You can optionally configure system users to perform additional tasks.

Depending on the release, installing or upgrading may automatically create a new user group and user that is granted access permissions to run certain OS-level processes required by Teradata Vantage. You may need to manually grant additional OS permissions to the user and to other administrative users. Also see [Working with OS-Level Security Options](#).

## User DBC

The primary system user, DBC, is the owner or parent of all users, databases, and other objects in the database. By default, user DBC has all privileges for interacting with the database, and can grant access privileges to lower level users.

Initially, user DBC contains:

- All available SQL Engine-managed disk space
- Data Dictionary system tables
- System views that allow retrieval of data from the underlying system tables

The primary active function of user DBC is to initially create the administrative users, transfer the ownership of most available disk space to them, and grant them privileges in the database. Administrative users should then perform most administrative functions under their own user names and not as DBC.

---

**Note:**

Because user DBC automatically has a wide range of database privileges, you should restrict access to the user DBC password to only the most trusted administrative personnel.

---

---

**Note:**

Never alter the database privileges for user DBC. If you change DBC privileges, you may cause installation, upgrade, maintenance, archive, or other procedures to fail, and require Teradata Support Center assistance to correct the problem.

---

## Other System-Generated Users

These additional system-generated users are automatically created at system startup.

- SysAdmin
- SYS\_CALENDAR
- SystemFE
- ALL
- CRASHDUMPS
- DEFAULT
- EXTUSER
- EXTERNAL\_AP
- PUBLIC
- TDPUSER

---

**Note:**

Most system-generated users perform important system-level functions. Do not alter the database privileges for any system user unless the documentation or Teradata Services personnel specifically direct you to do so. You should limit access to system-generated users to trusted administrators.

---

An exceptions to this rule are EXTUSER, PUBLIC, and SYS\_CALENDAR, which you can administer according to site policy.

For more information on system-generated users, see *Teradata Vantage™ - Database Administration*, B035-1093.

## Working with System-Level Space Allocation

User DBC initially owns all storage space available to the Teradata Vantage Advanced SQL Engine . You must reallocate this space to individual users and databases.

### About Database Space Types

Teradata Vantage allows you to allocate several types of space:

Type	Description
Permanent space (perm space)	Stores data and database objects. Tables use the perm space owned by the containing database, as needed. Owners of perm space own any object that inhabits the space.
Spool space	The space where database queries execute. By default, executing queries draw spool space from any unused system space.
Temporary space (temp space)	Stores temporary tables and volatile tables. Allocation and use of temp space is optional.

For more information, see space-related topics in *Teradata Vantage™ - Database Administration*, B035-1093.

## Recommended Space Allocations for System-Generated Users

Although primary system users receive a space default allocation during initial system setup, you must manually adjust the space for the Crashdumps and DBC users.

System-generated User and Function	Perm Space Required
Crashdumps Temporarily stores PDE memory dumps.	Approximately 1 GB default For information on allocating crashdumps space, see <a href="#">Assigning Perm Space to the Crashdumps Database</a> .
DBC Initially owns all usable disk space. Contains: <ul style="list-style-type: none"> <li>• Data Dictionary tables</li> <li>• Views and macros</li> <li>• System tables</li> <li>• Logs and transient journals</li> </ul>	After allocation of space to administrative users, DBC should retain approximately 10% of the total available space to contain system tables.  <b>Note:</b> The space retained by user DBC space is not usable for any other functions. To allocate administrator space, see: <ul style="list-style-type: none"> <li>• <a href="#">Creating the Security Administrator User</a>.</li> <li>• <a href="#">Setting Up the Database Administrator User</a></li> </ul>
SysAdmin	~40 MB default

System-generated User and Function	Perm Space Required
Contains tables, views and macros for network-based load jobs. <b>Note:</b> Do not reuse the name SysAdmin.	
Sys_Calendar Contains tables and views for date-related functions.	~15 MB default
SystemFe Contains diagnostic tables and macros for use by Teradata Customer Service.	~60 MB default

## Creating the Spool Space Reserve

You should create a spool space reserve from the space available to user DBC, to ensure that there is adequate space to execute queries.

### Note:

System-wide use of spool can exceed the spool reserve, if necessary, as long as there is other uncommitted DBC perm space.

The following procedure sets the spool space reserve at 20% of total available DBC space, which you may need to adjust later, depending on the number of simultaneous queries and the size of the tables in the database.

1. Determine the perm space owned by user DBC, for example:

```
SELECT SUM(MAXPERM) FROM DBC.DISKSPACE WHERE DataBaseName = 'dbc';
```

2. Calculate a value in bytes equal to 20% of the DBC space (the figure obtained in step 1) to set aside as spool space reserve, for example:

$$(6347.75 \text{ GB}) * (.20) = 1269.55 \text{ GB}$$

3. Create the Spool Reserve database and allocate perm space according the figure you calculated, for example:

```
CREATE DATABASE Spool_Reserve FROM DBC
AS PERM = 1269.55 GB
NO FALLBACK
NO BEFORE JOURNAL
NO AFTER JOURNAL ;
```

**Note:**

Specifying spool space in a CREATE USER or CREATE PROFILE statement defines the portion of the overall available spool that is available to users or profile members.

For information on peak spool space utilization (Peak Spool), see *Teradata Vantage™ - Data Dictionary*, B035-1092.

## Assigning Perm Space to the Crashdumps Database

Crashdumps appear as table images, which contain information that Teradata Services uses to investigate crashes.

**Note:**

Crashdumps accumulate as long as there is available space. You must delete old crashdump files manually.

To assign perm space to the crashdumps:

1. Calculate optimal perm space for the crashdumps database using the formula:

Crashdumps Perm Space = (Number of TPA Nodes) \* (Memory in GB per node) \* (6)

This equation is based on the following assumptions:

- 32 GB of memory per node
- A crashdump uses approximately 50% of node memory capacity
- Allocate enough space for at least 3 crashdumps
- If you specify FALLBACK protection for tables, the required space doubles

Based on these assumptions, the calculation for a four-nodes system is:

Crashdumps Perm Space = (4) \* (32 GB) \* (6), or 768 GB.

2. Modify the default perm space value for the crashdumps user from the 1 GB default to the calculated value:

```
MODIFY USER Crashdumps AS
  PERM=perm_spaceSTARTUP = ''
  FALLBACK
  NO BEFORE JOURNAL
  NO AFTER JOURNAL
  COLLATION = HOST
  DEFAULT CHARACTER SET = LATIN
  DATEFORM = INTEGERDATE
  TIME ZONE = NULL ;
```

where *perm\_space* is the amount of space you calculated. All other syntax elements shown are default settings.

**Note:**

Periodically monitor crashdumps space utilization and adjust space allocations as necessary.

## Related Information

For step...	Information on...	Is available in...
1 and 2	The use and maintenance of the crashdumps database	<i>Teradata Vantage™ - Database Administration</i> , B035-1093
	Details about perm space	
2	The MODIFY DATABASE statement	<i>Teradata Vantage™ - SQL Data Definition Language Syntax and Examples</i> , B035-1144

## Working with Administrative Users

All Teradata Vantage sites should have at least one user with clearly defined administrative duties.

Teradata recommends that you divide administrative duties between the security administrator and the database administrator, regardless of the number of administrators at your site.

For a list of duties, see:

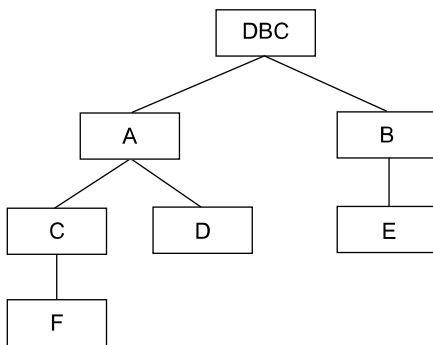
- [About Security Administrator Responsibilities](#)
- [About Database Administrator Responsibilities](#)

For setup suggestions, see:

- [Creating the Security Administrator User](#)
- [Setting Up the Database Administrator User](#).

User DBC should create both the database administrator and security administrator users, and should allocate DBC disk space that is not needed for system functions to the administrative users. Then the administrative users can create lower level administrators and other database users, and allocate space to them.

When administrators create users and databases, a hierarchical relationship results.



DBC directly creates the security administrator (A) and database administrator (B) users. DBC is the immediate owner (parent) of A and B, and owns everything in the hierarchy.

The security administrator (A) owns database (C) and assistant administrator (D), and then creates end user (F) in database (C).

- A is the immediate owner, or parent, of C and D.
- C is the immediate owner, or parent, of F.

The database administrator (B) creates database (E). B is the immediate owner of E.

Owners have implicit privileges on the objects they own. See [Ownership Privileges](#).

## About Security Administrator Responsibilities

- Work with the database administrator to review site security requirements and evaluate the Teradata Vantage security features that support those requirements.
- Review the options for managing system user authentication and authorization, and select a strategy, based on site security policy.
- Develop a security policy based on how the site uses Teradata Vantage capabilities to meet security needs. Distribute the policy to administrators.
- Set up and manage the TDGSS configuration files to support the user authentication and authorization strategy.
- Create and maintain Vantage users, roles, profiles and security constraints.

---

### Note:

Although the security administrator creates these objects, the database administrator has knowledge of user needs and the required privileges on Vantage objects, and is also responsible for determining and managing user privileges.

---

- Coordinate with the database administrator and directory administrator to set up optional directory management of Vantage users, including provisioning users in the directory, configuring associated mechanism properties, and setting up binding, protection, and user identification options.
- Manage user logon permissions and password controls.

- Set up optional access restrictions by IP address.
- Set up Vantage access logging and monitor the output for security violations.
- Create a set of site security procedures and distribute them to users.
- Take action to repel security threats, including revising user privileges, revoking logons, and dropping users. The security administrator should only take enforcement actions with the knowledge and participation of the database administrator.

For information on creating the security administrator and granting user privileges sufficient to carry out these responsibilities, see [Creating the Security Administrator User](#).

## About Database Administrator Responsibilities

- Work with the security administrator to establish user management policy.
- Create and manage databases, tables, views, macros, stored procedures, user defined functions, journals, query logs, and other database objects.
- Grant privileges to roles and users on database objects.
- Allocate space to users and databases, and manage space usage.
- Create and manage accounts.
- Manage data load and export.
- Manage data archive and restore.
- Monitor and tune system performance.
- Troubleshoot user problems.
- Manage periodic database maintenance tasks.

See [Setting Up the Database Administrator User](#).

## Creating the Security Administrator User

You should create the principal security administrator, SECADMIN, to carry out initial setup of database security.

---

### Note:

The following examples use the name SECADMIN, but it is not required.

---

For sites that require more than one security administrator:

- After it creates and provisions individual administrative users, SECADMIN can function mainly as a database, owning security administrators and security-related tables. Individual administrators can perform security duties.
- Use roles to define privileges for specific administrative functions.

Perform the following steps to create the security administrator user and grant the minimum privileges necessary to carry out security administration duties:

1. Log on to the database as user DBC.



2. Update `ExpirePassword` in `DBC.SysSecDefaults` to a non-zero value so the temporary password assigned to the security administrator user (and all other temporary passwords) expires at first logon, requiring users to create a private password. For example:

```
UPDATE DBC.SysSecDefaults SET ExpirePassword = 90 ;
```

---

**Note:**

The actual value of `PasswordExpire` should conform to your site security policy.

Subsequently, the password for each user expires each time the increment of days passes.

See [ExpirePassword](#).

---

3. Create the security administrator user with a `CREATE USER` statement:

```
CREATE USER secadmin FROM space_owner AS
PERM = perm_space
PASSWORD = temp_password
[ SPOOL = spool_space ]
[ ACCOUNT = 'account' ]
[ DEFAULT DATABASE = secadmin ]
[ FALLBACK ];
```

***secadmin***

The name of the primary security administrator user (for example, `SECADMIN`).

***space\_owner***

The owner of the space in which you create the user *secadmin*.

***perm\_space***

The space in bytes that contains all objects that user *secadmin* creates or owns.

Recommendation: Specify approximately 2-3% of available system (*space\_owner*) space, in which the security administrator can create users and databases and tables for security-related data.

For information on how to determine *secadmin* space, see [Creating the Spool Space Reserve](#).

***temp\_password***

The temporary password for user *secadmin*.

Recommendation: Use any simple password that follows the default system password controls and site policy. The set up in step 2 causes the temporary password to expire at first logon.

#### ***spool\_space***

[Optional] Limits the portion of the spool reserve that *secadmin* can use to execute queries.

Recommendation: Specify approximately 20% of the spool reserve.

#### ***account***

[Optional] Account strings are an optional method of controlling the granularity of resource accumulations that are reported in the DBC.Acctg table. Account strings allow the collection of CPU and I/O that is reported in DBC.Acctg to be grouped by application time of day, and priority. Variable substitution parameters included in the account string will be resolved at execution time.

A secondary use of account strings is to direct the classification of a query to a default Timeshare workload in TASM or TIWM. This classification will be effective only in cases where normal classification processes do not match the query to a user-defined workload.

An account string defines the following characteristics for profile member sessions in the database:

- Session priority
- Account ID (to assign charges for system time and resources)
- Date stamp
- Time stamp

Recommendation: Define at least one account string per profile. If necessary, you can add other accounts later using a MODIFY PROFILE statement.

For more information about accounts, see *Teradata Vantage™ - Database Administration*, B035-1093.

#### **DEFAULT DATABASE = *secadmin***

[Optional] The user or database that contains the space in which the database stores or searches for new or target objects unless a different database is specified in the transaction SQL.

Recommendation: Specify SECADMIN, which contains all databases that the administrator needs to access except DBC tables.

**FALLBACK**

[Optional] Directs the system to automatically create a duplicate of each table stored in the database space, to provide backup in the event of a failure.

Recommendation: The data in *secadmin* is critical, so you should specify FALLBACK to set up security tables for automatic duplication.

You can specify other syntax options in the initial CREATE USER statement or later in a MODIFY USER statement.

4. Submit the following SQL statements to grant user *secadmin* the basic privileges recommended by Teradata for security administrators.

**Note:**

The example SQL statements below provide *secadmin* with the privileges recommended for the duties defined in [About Security Administrator Responsibilities](#). You can assign some or all of these privileges to the database administrator in addition to, or instead of, the security administrator, as required by site security policy.

```
GRANT USER ON SECADMIN TO SECADMIN          /* maintain users */ ;
GRANT ROLE TO SECADMIN                      /* maintain roles */ ;
GRANT PROFILE TO SECADMIN                   /* maintain profiles */ ;
GRANT SELECT ON DBC TO SECADMIN             /* select on dictionary
tables */;
GRANT UPDATE ON DBC.SysSecDefaults TO SECADMIN /* password
characteristics */ ;
GRANT EXECUTE ON DBC.LogonRule TO SECADMIN   /* logon rules */ ;
GRANT EXECUTE ON DBC.AccLogRule TO SECADMIN  /* access logging */ ;
GRANT DELETE ON DBC.AccLogTbl TO SECADMIN    /* delete audit entries */ ;
GRANT DELETE ON DBC.DeleteAccessLog TO SECADMIN /* delete audit entries */ ;
GRANT DELETE ON DBC.EventLog TO SECADMIN     /* delete event log */ ;
```

5. Log off as user DBC.
6. Immediately log back onto Teradata Vantage as user *secadmin*. Create a private password at the prompt.
7. Enter the following SQL statement to log the attempt to execute of any BEGIN/END LOGGING or GRANT/REVOKE LOGON statement:

```
BEGIN LOGGING WITH TEXT ON EACH ALL ON MACRO DBC.LogonRule,
MACRO DBC.AccLogRule ;
```

8. Log off Teradata Vantage.

**Note:**

You can assign other privileges to the security administrator based on system needs.

## About the CTCONTROL Privilege

The CTCONTROL privilege links an administrator to a middle-tier application that is set up to conduct trusted sessions. The CTCONTROL privilege is required for any administrator who needs to use the GRANT CONNECT THROUGH statement to complete the setup of trusted sessions on middle-tier applications. You can grant the CTCONTROL privilege only to permanent users, and not to roles or other database objects.

## Granting CTCONTROL Privileges

You can use the GRANT CTCONTROL statement to provide users the privilege to set up and administer trusted sessions, for example:

```
GRANT CTCONTROL ON trusted_user TO database_username;
```

### *trusted\_user*

The user name for the middle-tier application being set up for trusted sessions.

### *database\_username*

The name of a permanent user, for example SECADMIN, who gains the privilege to administer proxy users and associated roles for the trusted user application.

---

### Note:

The GRANT CTCONTROL statement can specify only one trusted user, but up to 25 administrator users, per statement.

---

For details on use of the GRANT CTCONTROL statement, see *Teradata Vantage™ - SQL Data Control Language*, B035-1149.

## Example: GRANT a User CTCONTROL Privilege

This example grants user Admin\_01 the privilege to grant the CONNECT THROUGH privilege to users of a specific middle-tier application, which uses mta as its logon username.

```
GRANT CTCONTROL
ON mta
TO Admin_01 ;
```

## Related Information

For information on...	See...
Creating trusted users and proxy users	<a href="#">Working with Middle-Tier Application Users.</a>
Setting up administrators to manage row level security constraints	<a href="#">Working with Security Constraint Administrative Privileges.</a>

## Setting Up the Database Administrator User

Create the principal database administrator, user DBADMIN, allocate and grant privileges to the user before proceeding with implementation of database security. User DBADMIN can then create databases, tables, users, and other objects in the space it owns.

## Creating the Database Administrator Profile

Profiles define the system resources available to member users, and can also set password control parameters.

1. Log on to the database as user DBC.
2. Create the profile for the database administrator:

```
CREATE PROFILE admin_profile AS
[ SPOOL = spool_space ]
[ TEMPORARY = temp_space ]
[ ACCOUNT = 'account' ];
```

### ***admin\_profile***

The name of the database administrator profile (for example, DBADMIN).

Recommendation: Administrative procedures in this publication use the name AdminProfile for the database administrator profile.

### **Note:**

User DBADMIN is the only member of this profile. Because of space and ownership concerns, you should create one or more separate profiles for other administrative users.

### ***spool\_space***

[Optional] The amount of space available for intermediate query results or formatted answer sets to queries and volatile tables. The system borrows spool space from unused permanent space anywhere in the system.

Recommendation: Set this value equal to the amount of space in the Spool\_Reserve database defined in [Working with System-Level Space Allocation](#).

### ***temp\_space***

[Optional] Required only when you use global temporary tables, volatile tables and other features that require temporary space.

### ***account***

[Optional] Account strings are an optional method of controlling the granularity of resource accumulations that are reported in the DBC.Acctg table. Account strings allow the collection of CPU and I/O that is reported in DBC.Acctg to be grouped by application time of day, and priority. Variable substitution parameters included in the account string will be resolved at execution time.

A secondary use of account strings is to direct the classification of a query to a default Timeshare workload in TASM or TIWM. This classification will be effective only in cases where normal classification processes do not match the query to a user-defined workload.

An account string defines the following characteristics for profile member sessions in the database:

- Session priority
- Account ID (to assign charges for system time and resources)
- Date stamp
- Time stamp

Recommendation: Define at least one account string per profile. If necessary, you can add other accounts later using a MODIFY PROFILE statement.

For more information about accounts, see *Teradata Vantage™ - Database Administration*, B035-1093.

## **Creating User DBADMIN**

Use the following procedure to create the principal database administrator, DBADMIN. For sites that require multiple database administrators, create lower level administrative users who log on as individuals. Do not allow multiple users to log on as DBADMIN, which prevents the system from logging the actions of individuals.

1. Log on to the database as user DBC.
2. Create the database administrator:

```
CREATE USER DBADMIN FROM DBC
AS PERM = perm_space
PASSWORD = "temporary_password"
```

```
[ DEFAULT DATABASE = DBADMIN ]
[ PROFILE = AdminProfile ];
```

**DBADMIN**

The name of the database administrator user. You can specify a different username, but all the examples refer to the database administrator as DBADMIN.

**DBC**

The owner of the space in which you create user DBADMIN.

***perm\_space***

The space in bytes that contains all objects that user DBADMIN creates or owns. Because user DBADMIN creates and owns nearly all Vantage databases and tables, you should assign it the majority of space on the system.

Recommendation: Specify approximately 60% of available system (DBC) space. For information on how to determine DBC space, see [Creating the Spool Space Reserve](#).

***temporary\_password***

The temporary password for DBADMIN.

Recommendation: Use any simple password that follows the default system password controls and site policy. Each user is prompted to change the temporary password to a permanent, private password at first logon.

**DEFAULT DATABASE = DBADMIN**

[Optional] The user or database that contains the space in which the SQL Engine stores or searches for new or target objects, unless the transaction SQL specifies a different database.

Recommendation: Specify DBADMIN.

**PROFILE = AdminProfile**

[Optional] The name of the profile for DBADMIN.

Recommendation: Enter AdminProfile, the profile you created for user DBADMIN in [Creating the Database Administrator Profile](#).

**Related Information**

Information on...	Is available in...
Syntax and options for the CREATE USER statement	<i>Teradata Vantage™ - SQL Data Definition Language Detailed Topics</i> , B035-1184.

Information on...	Is available in...
Guidelines for determining administrator space requirements and long term managing of space	<i>Teradata Vantage™ - Database Administration</i> , B035-1093.
Default password control values and how to change them	<a href="#">Managing Database Passwords</a> .

## Granting Privileges to User DBADMIN

The following procedure grants privileges typically required by the database administrator user to carry out the duties listed in [About Database Administrator Responsibilities](#).

### Note:

You can adjust the database administrator privileges shown below to conform to your site security policy.

1. Log on as user DBC.
2. Grant object-level database privileges to the database administrator on all objects subsequently created in DBADMIN space.

```
GRANT EXECUTE, SELECT, INSERT, UPDATE, DELETE, STATISTICS, DUMP, RESTORE,
CHECKPOINT, SHOW, EXECUTE PROCEDURE, ALTER PROCEDURE, EXECUTE FUNCTION,
ALTER FUNCTION, ALTER EXTERNAL PROCEDURE, CREATE OWNER PROCEDURE, CREATE
TABLE, CREATE VIEW, CREATE MACRO, CREATE DATABASE, CREATE TRIGGER, CREATE
PROCEDURE, CREATE FUNCTION, CREATE EXTERNAL PROCEDURE, CREATE AUTHORIZATION,
DROP TABLE, DROP VIEW, DROP MACRO, DROP DATABASE, DROP TRIGGER, DROP
PROCEDURE, DROP FUNCTION, DROP AUTHORIZATION ON DBADMIN TO DBADMIN WITH
GRANT OPTION;
```

3. Grant the privilege to MODIFY users and profiles, which is required to administer such attributes as account, default database, profile (in a user definition), and space allocation. The DROP privilege is required to use the MODIFY command.

```
GRANT DROP USER ON DBADMIN TO DBADMIN WITH GRANT OPTION;
GRANT DROP PROFILE TO DBADMIN WITH GRANT OPTION;
```

4. Grant object-level privileges on DBC tables and views to DBADMIN.

```
GRANT EXECUTE, SELECT, STATISTICS, SHOW ON DBC TO DBADMIN WITH GRANT OPTION;
```

5. Grant additional system-level privileges that not included in other grants.

```
GRANT MONRESOURCE, MONSESSION, ABORTSESSION, SETSESSRATE, SETRESRATE TO
DBADMIN WITH GRANT OPTION;
```



- Grant privileges on Sys\_Calendar, which contains data for date-related system functions.

```
GRANT SELECT, INSERT, UPDATE, DELETE ON Sys_Calendar TO DBADMIN WITH
GRANT OPTION;
```

- Log off as user DBC.
- Immediately log on to the database as DBADMIN, using the temporary password specified in [Creating User DBADMIN](#).
- When prompted, create a private password for user DBADMIN.

## Related Information

Information on...	Is available in...
System tables and views	<i>Teradata Vantage™ - Data Dictionary</i> , B035-1092.
Syntax and options for the GRANT statement	<i>Teradata Vantage™ - SQL Data Control Language</i> , B035-1149.
Secure zones	<a href="#">Implementing Teradata Secure Zones</a> .

# Working with the Common Criteria Standard

## About the Common Criteria Standard

The Common Criteria international security standard defines the most stringent requirements for the security administrator, and incorporates the requirements of the US National Computer Security Center (NCSC).

Teradata Vantage includes the necessary security features in the Advanced SQL Engine software to generally comply with the Trusted Database Interpretation (TDI) at the C2 level, but Teradata makes no claim that the release of Teradata Vantage described in this document has been evaluated at the C2 level or that it complies with the TDI.

Certified auditors evaluate each Teradata Vantage major release for conformance to the Common Criteria International Security Standard, ISO/IEC 15408. This standard provides a common set of requirements for the security functions of IT products and systems, and for the assurance measures that evaluators apply to them during a security evaluation. The evaluation process establishes a level of confidence that the security functions of a product or system meet these requirements.

## Setting Up a System for Common Criteria Compliance

The following procedure describes how to set up and operate a Teradata Vantage system at a level of security equivalent to the Common Criteria evaluated configuration.

**Note:**

If your Vantage system is in operation, you may have already performed some of the needed steps during system implementation.

1. Establish a system security policy.
2. Establish the physical security controls for the system. See [Controlling Physical Access](#).
3. Establish operating system and network security controls for the database server.
4. Install the system hardware and software following Teradata-supplied installation documentation. Be sure to use the documentation that corresponds to the applicable hardware, software, and operating system versions.
5. Run the DIPACC script, provided on the software release media, to allow access logging. See [Setting Up the DBC.AccLogRule Macro](#). This script creates the AccLogRule macro in system user DBC, which allows you to enable access logging.

**Note:**

Do not run the tpareset command until instructed, later in this procedure.

6. Change the password control parameters defined in the DBC.SysSecDefaults table to the recommended default values:

```
UPDATE DBC.SysSecDefaults SET
  /* password must be at least 8 characters in length */
  PasswordMinChar = 8,
  /* password cannot exceed 30 characters */
  PasswordMaxChar = 30,
  /* digits required in a password */
  PasswordDigits = 'r',
  /* alpha, special characters required in a password */
  PasswordSpecChar = 'r',
  /* user name will be locked after 3 failed logons */
  MaxLogonAttempts = 3,
  /* user name will remain locked for 5 minutes */
  LockedUserExpire = 5,
  /* passwords will expire in 90 days */
  ExpirePassword = 90,
  /* a password cannot be reused for 270 days */
  PasswordReuse = 270
  /* dictionary words cannot be used in a password */
  PasswordRestrictWords = 'Y'
WHERE PrimeIndex = 1;
```

See also [About Password Controls](#).

7. Change the default PASSWORD parameter for usernames DBC, SYSTEMFE, and SYSADMIN (via MODIFY USER), and protect the new passwords in accordance with your security policy. For example, to modify user DBC:

The following example shows the SQL you can use to modify user DBC:

```
MODIFY USER DBC AS PASSWORD = xxx;
```

8. Grant the necessary rights to SECADMIN to carry out security administrator duties. Only the security administrator (and user DBC) should have these privileges.

```
GRANT USER ON SECADMIN TO SECADMIN          /* maintain users */ ;
GRANT ROLE TO SECADMIN                       /* maintain roles */ ;
GRANT PROFILE TO SECADMIN                    /* maintain profiles */ ;
GRANT SELECT ON DBC TO SECADMIN              /* select on dictionary
tables */ ;
GRANT UPDATE ON DBC.SysSecDefaults TO SECADMIN /* password
characteristics */ ;
GRANT EXECUTE ON DBC.LogonRule TO SECADMIN    /* logon rules */ ;
GRANT EXECUTE ON DBC.AccLogRule TO SECADMIN   /* access logging */ ;
GRANT DELETE ON DBC.AccLogTbl TO SECADMIN     /* delete audit
entries */ ;
GRANT DELETE ON DBC.DeleteAccessLog TO SECADMIN /* delete audit
entries */ ;
GRANT DELETE ON DBC.EventLog TO SECADMIN      /* delete event log */ ;
```

---

**Note:**

These are the minimum privileges required for the security administrator to fulfill Common Criteria requirements, but you can grant additional privileges if your security policy requires them.

---

9. Use the tpareset command to restart the system to activate access logging and the revised password controls. For more information, see *Teradata Vantage™ - Database Utilities*, B035-1102.
10. Initiate logging of all user attempts to access the security administrator macros, including access by the security administrator, to check for possible attempts to learn or compromise system security measures:

```
BEGIN LOGGING WITH TEXT ON EACH ALL
ON MACRO DBC.LogonRule, MACRO DBC.AccLogRule;
```

11. Establish any additional security logging that is required by your site security policy. See [Monitoring Database Access](#).
12. Establish any logon rules required by your security policy.
13. Implement an analysis and reporting process to examine the output of access logs, according to your site security policy. See [Investigating Database Access Attempts](#).

## Avoiding Potential Security Hazards

Observe the following when you set up and administer security policy:

- Use caution when you consider whether to include the WITH NULL PASSWORD option in a GRANT LOGON statement for a mainframe-based user.

Although the Teradata Director Program security exit must validate a null password parameter in a particular logon string, Teradata Vantage itself cannot verify the username and password.

- Never define SESSION POOLING to the Teradata Director Program (for mainframe access). Session pooling for mainframe users offers faster logons, but prevents Teradata Vantage from uniquely identifying and authenticating users.
- Never use the GRANT WITH GRANT OPTION statement for databases, objects, and non-administrative users. Although this requires more work for administrators, it forces the administrative staff to be involved in all GRANT requests by non-owners.
- Never alter privileges for user DBC. Doing so may cause installation, upgrade, maintenance, or archive procedures to end abnormally and consequently require help from the Teradata Support Center to correct the problem.

## Controlling Physical Access

The most basic level of security is to limit access by unauthorized persons to the physical components of the Teradata Vantage system. These components include database nodes and disk storage units.

- Protect the system components against deliberate damage by locating them in a secure room.
- Control access to external devices that connect to the system, such as remote terminals.

## Controlling Access to the Operating System

Restrict access to the operating system (root access) to a minimum number of administrators with special privileges. Establish operating system and network security controls to secure Teradata Vantage. Restrict users without special privileges from accessing the LAN through the operating system to prevent possible corruption of data files.

## About Security Hardening

Teradata performs operating system security hardening during initial system setup. Security hardening prevents users with access to Teradata Vantage from using this access to perform unauthorized activities at the operating system level.

## Working with OS-Level Security Options

During installation of Teradata Vantage, the system automatically creates the following default OS-level security structure.

Default User or Group	Description
Users	
teradata	Advanced SQL Engine runs as the teradata user, which is a member of the tdtrusted group.
tdatuser	Runs UDFs in protected mode and is a member of the tdatudf group.
Groups	
tdtrusted	<p>Has permission to run OS-level processes and utilities, and provides this permission to member users:</p> <ul style="list-style-type: none"> <li>• teradata (created by default to run the database)</li> <li>• Other administrative users that you create who require OS-level access, for example, to run utilities or change the TDGSS configuration.</li> </ul> <p><b>Note:</b> Although you can run OS-level utilities and processes as root, Teradata recommends that for secure operation you severely limit root access and create individual administrative user accounts in the tdtrusted group to run Teradata utilities and other OS-level functions. For information on starting utilities that need OS-level of access, see <i>Teradata Vantage™ - Database Utilities</i>, B035-1102.</p>
tdatudf	<p>Has permission to run UDFs in protected mode and provides this permission to member users:</p> <ul style="list-style-type: none"> <li>• tdatuser (created by default)</li> <li>• Other users you create who need to run UDFs in secure mode</li> </ul>

**Note:**

Although most OS-level tasks can be run by the users defined in the table above, you must use root access to:

- Install a new version of Teradata Vantage or Advanced SQL Engine
- Start the database when it is down

If your site security policy requires an alternative OS-level access strategy, contact your Teradata Customer Service representative for assistance.

# Implementing User Authentication and Authorization

The primary function of Teradata Vantage security is to authenticate and authorize Vantage users. You must select, and where required configure, the authentication and authorization options for your system, as follows:

1. Evaluate authentication and authorization options and devise a strategy. See:
  - [About Teradata Authentication](#).
  - [About External Authentication](#).
  - [About Authentication and Authorization of Middle-Tier Application Users](#).
2. Enable external authentication, if required. See:
  - [About External Authentication Controls](#).
  - [About External Authentication Requirements](#).
3. If you decide to use Kerberos authentication (KRB5 or SPNEGO mechanism), complete the required setup. See [Working with Kerberos Authentication](#).
4. If you decide to use the JSON Web Token (JWT) authentication mechanism, complete the required setup. See [Configuring Single Sign-On](#).
5. If you decide to authenticate and/or authorize users in an LDAPv3-compliant directory, see the topics beginning with [Directory Management of Database Users](#), and follow the setup instructions for the options you choose to implement.
6. Understand logon format requirements associated with various authentication and authorization methods. See [Logging on to Teradata Vantage](#).

For information about authentication options for mainframe-based users, see [Using External Authentication from Mainframe Clients](#).

## About Teradata Authentication

Authentication and authorization of users by Teradata Vantage is enabled by default, but you must complete the following additional tasks before users can log on:

- Define and provision users. See [Creating Users and Granting Privileges](#).
- Instruct users about the logon format required for Teradata authentication. See [Logging on Using Teradata Authentication and Authorization](#).

## About External Authentication

You can use external agents, for example Kerberos or an LDAPv3-compliant directory, to authenticate Vantage users that log on from Windows, Linux, and supported Teradata Tools and Utilities (TTU) UNIX clients (except IBM z/OS clients).

## External Authentication with Teradata Vantage Authorization

---

**Note:**

Use of Kerberos authentication requires that you complete a multi-part configuration procedure. See [Working with Kerberos Authentication](#).

---

### Single Sign-on (Kerberos Authentication)

1. A user logs on to the network with a domain username that matches a Teradata Vantage username, and a domain password. Kerberos authenticates the user. The user can then access any applications and data that support Kerberos authentication, including the Advanced SQL Engine.
2. The user connects to Vantage without resubmitting a username and password, although the connection must specify the Vantage system name (the *tpid*) and the security mechanism that corresponds to the authenticating agent. See [Logging on Using Single Sign-on with Kerberos](#).
3. The authenticating agent for the network logon forwards the domain username to Vantage.
4. Vantage authorizes the user with the database privileges available to the matching username.

### Sign-on As (Kerberos or LDAP Authentication)

1. A user logs on to Teradata Vantage with a domain username that matches a Vantage username, and a domain password. The default mechanism, or the mechanism named in the logon, defines the external agent that authenticates the user.

See [Logging on Using Sign-on As](#).

---

**Note:**

Sign-On As using Kerberos authentication (KRB5 or SPNEGO mechanism) is usable only from Windows clients.

---

2. The authenticating agent forwards the user credentials to Vantage.
  3. Vantage authorizes privileges associated with the matching username.
- 

**Note:**

You must create Vantage users and grant them database privileges before the system can authenticate and authorize them. See [Creating Users and Granting Privileges](#).

---

## External Authentication with Directory Authorization

### LDAP Directory Authentication and Authorization

1. A directory-based user logs on to Vantage with a directory username and password, and is authenticated by the directory. See [LDAP Logon Format Examples](#).
2. The directory authorizes database privileges to the directory user based on mapping of the user to Vantage user, external role, and profile objects in the directory.

### Kerberos External Authentication with Directory Authorization

1. A directory-based user logs on to Vantage with a domain username and password, and is authenticated by Kerberos (KRB5 or SPNEGO mechanism). See [Using Sign-on As with Directory Authorization](#).
2. The directory authorizes database privileges to the user based on mapping between the user and Vantage user, external role, and profile objects in the directory.

### Kerberos External Authentication with Directory Authorization (Single Sign-on)

1. A directory-based user logs on with a domain username, and is authenticated by Kerberos (KRB5 or SPNEGO mechanism). The user can then access any applications and data that support Kerberos authentication, including Teradata Vantage.
2. The user connects to Vantage without resubmitting logon credentials, although the connection to Vantage must specify the Vantage system name (tdpid) and the security mechanism that corresponds to the authenticating agent if it is not set as the default. See [Using Single Sign-on with Directory Authorization](#).
3. The directory authorizes database privileges to the user based on:
  - Mapping between the directory user and Vantage user, external role, and profile objects in the directory
  - EXTUSER privileges, if mapped only to EXTUSER

Also see [Working with Directory User Management Options](#).

---

#### Note:

Users that use this logon method must be defined to Kerberos, and must have an entry in the directory that TDGSS can find using an <Identity Map> or <Identity Search>.

---



## Related Information

For information on...	See...
External authentication for mainframe clients	<a href="#">Using External Authentication from Mainframe Clients.</a>
Logon format requirements for external authentication	<a href="#">About Network Logons.</a>

## About External Authentication Controls

You can use external authentication controls to:

- Disable external authentication and allow only Teradata authentication
- Enable both external authentication and Teradata authentication
- Disable Teradata authentication and allow only external authentication

The controls are set so that external authentication of users is globally enabled by default in Vantage, however in addition to the control settings, you must also do the following to use external authentication:

- GRANT LOGON ... WITH NULL PASSWORD privileges to individual Vantage users who use external authentication
- Verify that the corresponding external authentication mechanism is enabled on each client (note, all mechanisms are enabled by default on the client, so this applies only if you change the default configuration on the client), each Vantage system accessed by the client, and on all Unity servers through which Vantage clients log on.

---

### Note:

If users log on to Vantage through Unity, external authentication controls and WITH NULL PASSWORD privileges should be the same on all Vantage systems managed by Unity.

---

## Using the External Authentication Controls

Teradata Vantage provides the following methods to control external authentication:

- The DBSControl utility controls external authentication for all users.
- The Gateway Control utility controls external authentication for users that log on through a particular gateway or host group.

For a host group:

- If either the DBSControl or Gateway Control utility is set to disable external authentication, external authentication is disabled. Users entering through that gateway cannot be externally authenticated.

- If either the DBSControl or Gateway Control utility is set to allow only external authentication, users entering through that gateway can only be authenticated externally. Setting either of these controls to allow only external authentication also prevents mainframe connections unless the TDP is also set up for external authentication.

**Note:**

You can also use the Database Window interface, or comparable command-line interfaces to the Console Subsystem (CNS), such as cnstern and cnstool, to manage external authentication. The SET EXTAUTH command toggles the same internal control switch used by the DBSControl utility. You can use the GET EXTAUTH command to check whether or not external authentication is enabled.

For information on DBSControl, Gateway Control, Database Window and starting the utilities, see *Teradata Vantage™ - Database Utilities*, B035-1102.

## About External Authentication Requirements

### Kerberos Authentication with Teradata Vantage Authorization

- Verify that the KRB5 mechanism is enabled on all clients that use Kerberos authentication and on all Teradata Vantage systems to which they connect.
- The client from which the user logs on must be running Windows, Linux, or supported TTU UNIX clients (except IBM z/OS clients) and the system must be setup as shown in [Working with Kerberos Authentication](#).
- Set the Kerberos authentication mechanism to be used (KRB5 or SPNEGO) as the client default, or the user must specify it at logon.
- The Vantage and Kerberos clients must be set up as shown in [Working with Kerberos Authentication](#).
- DBS Control and Gateway Control must be set to allow external authentication. See [About External Authentication Controls](#).
- All users authenticated by Kerberos must have LOGON ... WITH NULL PASSWORD privileges defined in each Teradata Vantage system to which they can log on. See [Working with User Privileges in Teradata Vantage](#).
- The domain username used at initial logon to the network must match a Teradata Vantage username. For acceptable logon username forms, see [Logging on Using Single Sign-on with Kerberos](#).
- For Kerberos authenticated users logging on through Unity, see *Teradata® Unity™ Installation, Configuration, and Upgrade Guide for Customers*, B035-2523 and *Teradata® Unity™ User Guide*, B035-2520.

### LDAP Authentication with Teradata Vantage Authorization

- The directory should be LDAPv3-compliant. See [About Certified Directories](#).

- Verify that the LDAP mechanism is enabled on all Kerberos clients, on all Vantage systems to which they connect, and the Unity server, if used. Set the LDAP mechanism as the client default, or the user must specify it at logon.
- The directory username used at logon must match a Teradata Vantage username. For acceptable logon username forms, see [Logging on Using Sign-on As](#).
- The matching Vantage username must have LOGON ... WITH NULL PASSWORD privileges. See [Working with User Privileges in Teradata Vantage](#).
- The LDAP AuthorizationSupported property must be set to no in the TdgssUserConfigFile.xml on the Teradata Vantage system and in the TdgssUnityConfig.xml on the Unity server if used. See [Changing the TDGSS Configuration](#).
- For LDAP authenticated users logging on through Unity, see *Teradata® Unity™ Installation, Configuration, and Upgrade Guide for Customers*, B035-2523 and *Teradata® Unity™ User Guide*, B035-2520.

### Kerberos or LDAP Authentication with Directory Authorization

- The directory should be LDAPv3-compliant. See [About Certified Directories](#).
- The client from which the user logs on must be Windows, Linux, or UNIX (except IBM z/OS clients) and the system must be setup as shown in [Working with Kerberos Authentication](#).
- Verify that the MechanismEnabled property is set to yes for the authentication mechanism (KRB5, SPNEGO, or LDAP) on the database, the Unity server (if used), and on all clients that use the mechanism.
- Set the mechanism as the client default, or the user must select it at logon.
- The user must have LOGON ... WITH NULL PASSWORD privileges.
- The username must follow these requirements:
  - For Kerberos authentication the authorized username must match a Teradata Vantage user having WITH NULL PASSWORD privileges, but the username does not have to be the same as the authenticated username for the user. If there is no authorization, the Kerberos username and Teradata Vantage name must match and be granted WITH NULL PASSWORD. See [About Logon Privileges](#).
  - For LDAP authentication, the directory user must be mapped to a database user having WITH NULL PASSWORD privileges.

For username requirements, see the topics about logging on with the Kerberos and LDAP authentication in [Logging on to Teradata Vantage](#).

- Configure the authentication mechanism for directory authorization in the TdgssUserConfigFile.xml on all required databases and in the TdgssUnityConfig.xml on the Unity server, if used. See [Changing the TDGSS Configuration](#).
- Configure the directory to map directory users to Teradata Vantage directory objects to define authorization criteria.

## Related Information

For information on...	See...
Directory mapping using Teradata schema extensions	<a href="#">Provisioning Directory Users with Teradata Schema Extensions.</a>
Directory mapping using only the native directory schema	<a href="#">Using Native Directory Schema to Provision Directory Users.</a>
Additional configuration requirements that apply when externally authenticated users log on through Unity	For information about Unity, see <i>Teradata® Unity™ Installation, Configuration, and Upgrade Guide for Customers</i> , B035-2523 and <i>Teradata® Unity™ User Guide</i> , B035-2520.

## Using External Authentication from Mainframe Clients

External authentication of users that log on from mainframe clients requires a special setup of the Teradata Director Program (TDP). For security options on mainframe clients, see *Teradata® Director Program Reference*, B035-2416.

### Note:

If you decide to use any of the external authentication functions provided by the TDP, you must enable external authentication in Teradata Vantage.

Since logons from mainframe clients do not go through the gateway, you cannot use the Gateway Control utility to enable external authentication. Instead, you must use the DBSControl utility for mainframe clients. See *Teradata Vantage™ - Database Utilities*, B035-1102.

## About Authentication and Authorization of Middle-Tier Application Users

Middle-tier applications are situated between end users and Teradata Vantage. When a middle-tier application accesses Vantage, it must log on as a permanent Vantage user and establish a session pool, after which end users can access Vantage through the application.

All middle-tier application users are authenticated by the application, or the network in which the application resides. For information on session pools, see *Teradata Vantage™ - Database Administration*, B035-1093.

- If you implement trusted sessions, you can authorize database privileges for application users individually. For information on establishing trusted sessions and assigning user privileges, see [Working with Middle-Tier Application Users](#).
- If you do not implement trusted sessions, all application users take on the identity and privileges of the application logon username. This strategy is not recommended because in this situation Vantage is unable to track users as individuals.

## About Authentication of Users Logging On through Unity

In an environment that contains multiple Teradata Vantage systems, you can use Unity to process and selectively route user queries to the systems based on defined rules. The query processing by Unity is unknown to the user, who is authenticated and authorized in the standard way, according to the security mechanism (default or specified) in effect at the time of logon.

Unity supports logons using of the following authentication mechanisms:

- TD2
- KRB5
- SPNEGO
- LDAP

To use Kerberos or LDAP authentication, you must complete several setup tasks on the Unity server and on all connected Vantage systems. For information about Unity, see *Teradata® Unity™ Installation, Configuration, and Upgrade Guide for Customers*, B035-2523 and *Teradata® Unity™ User Guide*, B035-2520.

## Working with Kerberos Authentication

You must complete a multipart setup procedure to enable use of Kerberos authentication:

1. Install/update Kerberos in the required locations. See [Installing and Configuring Kerberos](#).
2. Configure the Key Distribution Centers (KDCs) used for Kerberos authentication. See the topics beginning with [Working with Kerberos Setup on the KDC](#).
3. Configure Teradata Vantage, and Unity servers (if used). See [Configuring Teradata Vantage and Unity Servers for Kerberos Authentication](#).

---

### Note:

Consider all setup procedures for applicability to your system, then perform the necessary procedures in the order presented.

---

## Prerequisites

- The KRB5 and SPNEGO (if used) mechanisms are enabled.
- The AuthorizationSupported property for the mechanisms is set to:
  - 'no' if users are authorized privileges by Teradata Vantage
  - 'yes' if users are authorized privileges in a directory
- External authentication is set up for Vantage. See [About External Authentication Controls](#) and [About External Authentication Requirements](#).

- Vantage clients and Teradata Vantage are connected to the network. Teradata Vantage clients are already capable of executing Kerberos logons elsewhere in the network, and the Vantage system is accessible to your client system.
- For sites that use Unity, complete the configuration of the PROXY connection and related procedures shown in *Teradata® Unity™ Installation, Configuration, and Upgrade Guide for Customers*, B035-2523, before doing the Kerberos configuration.
- KDCs are set up for Kerberos authentication (except for the specialized Teradata Vantage requirements shown in the procedures that follow), and are operational.
- KDCs must run either Windows Kerberos or MIT Kerberos on Linux. Heimdal Kerberos is not supported.
- Users who plan to access Vantage using Kerberos authentication are already fully set up to use Kerberos for other non-Vantage network logons. For Kerberos authentication the authorized username must match a Teradata Vantage user having WITH NULL PASSWORD privileges, but the Vantage username does not have to be the same as the authenticated username for the user. If there is no authorization, the Kerberos username and Vantage name must match and be granted WITH NULL PASSWORD. For a description of valid Kerberos username forms, see [Logging on to Teradata Vantage](#).
- If a Vantage (service) in one realm can be accessed by a client situated in a different realm, a cross-realm trust must exist between the realms.

## Client Operating Systems Supported for Kerberos Authentication

The following client operating systems are supported when using Kerberos to authenticate Teradata Vantage clients:

- Windows: All versions supported by the Teradata Tools and Utilities (client) software.
- UNIX: All versions supported by the Teradata Tools and Utilities (client) software (except IBM z/OS).
- Linux:
  - Red Hat Linux-i386
  - Red Hat Linux-x8664
  - SUSE Linux-i386
  - SUSE Linux-x8664

---

### Note:

Vantage does not support Kerberos authentication for SUSE Linux-390 and SUSE Linux-390.32 (mainframe Linux clients).

---

# Installing and Configuring Kerberos

## About Kerberos Minimum Versions

Linux clients must have at least the required minimum versions of Kerberos packages installed. For Teradata certified minimum versions, log in to and search for KCS004617.

## Setting Up Kerberos on Linux and UNIX Clients

### Prerequisite:

The following discusses how to configure Kerberos for Linux and UNIX clients (except IBM z/OS).

The Teradata Tools and Utilities (TTU) client packages must be installed on your client system prior to configuring Kerberos. See the appropriate installation guide for your platform:

Operating System	Installation Documentation
Red Hat Linux	<i>Teradata® Tools and Utilities for Linux Installation Guide (Amazon Linux 2, CentOS, OEL, RedHat, SLES, Ubuntu)</i> , B035-3160
SUSE Linux	<i>Teradata® Tools and Utilities for Linux Installation Guide (Amazon Linux 2, CentOS, OEL, RedHat, SLES, Ubuntu)</i> , B035-3160
AIX	<i>Teradata® Tools and Utilities for IBM AIX Installation Guide</i> , B035-3125
Apple OS X	<i>Teradata® Tools and Utilities for Apple macOS Installation Guide</i> , B035-3129
Solaris (Intel)	<i>Teradata® Tools and Utilities for Oracle Solaris on SPARC and AMD Opteron Systems Installation Guide</i> , B035-3136
Solaris (SPARC)	<i>Teradata® Tools and Utilities for Oracle Solaris on SPARC and AMD Opteron Systems Installation Guide</i> , B035-3136

### Note:

Kerberos installation and configuration on Linux and UNIX clients does not require any Vantage-specific customizations. The following steps are guidelines only. For specific instructions on installing and configuring Kerberos, follow your vendor's instructions.

1. Add the client system name and IP address to your Active Directory.
2. Install Kerberos on the client system:
  - Install a supported version of Kerberos for Linux and UNIX clients.
  - For AIX, install the NAS Kerberos package from IBM. The IBM Kerberos implementation of Network Authentication Services (NAS) is shipped on their AIX expansion pack.

Run:

```
Installp -aqXYgd . krb5.client
```

3. Edit /etc/resolv.conf on the client system and include the Active Directory server name and IP address.
4. Include Active Directory information in krb5.conf.
  - On Linux and UNIX clients add the following to krb5.conf, for example:

```
[libdefaults]
    default_realm = example.com
    clockskew = 300
    dns_lookup_realm = true
    dns_lookup_kdc = true

[realms]
AREALM.example.com = {
    kdc = kdc.example.com:88
    admin_server = AREALM.example.com:749
    default_domain = AREALM.example.com
}

[domain_realm]
    .example.com = AREALM.example.com
    example.com = AREALM.example.com
```

- On AIX clients, run:

```
mkkrb5clnt -c <KDC> -r <Realm> -s <Server> -d <Domain>
```

For example:

```
mkkrb5clnt -c kdc.example.com -r AREALM.example.com -s AREALM.example.com
-d example.com
```

5. Verify the connection. Run nslookup from the client system and look up the Active Directory server (the KDC).
6. Run kinit on the client system to get a ticket granting ticket for your user:

```
#kinit <username>
```

7. Run klist on the client system to confirm the ticket granting ticket for your user was issued:

```
#klist
```



## About Kerberos on Windows Clients and Windows KDCs

All necessary Kerberos packages are already installed on Windows clients and Windows KDCs as part of normal network setup.

## Configuring Local Security Policy Encryption Types for Kerberos

Windows versions that include the Local Security Policy setting for encryption types need to enable the setting for KRB5 support.

1. On the Windows KDC go to Security Options settings, for example, **Start > Control Panel > Administrative Tools > Local Security Policy > Local Policies > Security Options > Network security: Configure encryption types allowed for Kerberos**.
2. Enable all the algorithms that are available in the menu or enable all the algorithms that are allowed to be used by your site policy.

## Installing Kerberos on Teradata Vantage Nodes and Unity Servers

You must install at least the minimum supported SUSE Linux version of the krb5 and krb5-client packages on all database nodes, and on the Unity server, if used. The krb5 packages are included with the standard Teradata Vantage Linux operating system.

The default location for Kerberos is `/usr/lib64` on database nodes, and on the Unity server, if used. If you need to install Kerberos to an alternate location, see [Reconfiguring TDGSS for a Non-Standard Installation of Kerberos](#).

---

### Note:

Other krb5 packages may already exist on the machine, but do not affect this process.

---

## Reconfiguring TDGSS for a Non-Standard Installation of Kerberos

If you install Kerberos in a location other than the standard `/usr/lib64`, you must also update the `TdgssUserConFigFile.xml` for database, and the `TdgssUnityConfig.xml` for Unity servers (if used), as shown in the procedure that follows, after you install Kerberos to the alternate location. This procedure is required on each Teradata Vantage system, and on each Unity server (if used), so the configuration file can access the Kerberos library `libgssapi_krb5.so`.

---

### Note:

Teradata recommends that you complete all Kerberos setup tasks required on the Teradata Vantage nodes and on the Unity server, if used, before you do any other edits to KRB5 mechanism properties.

---

## Changing the Configuration on Teradata Vantage Nodes and Unity Server

On Teradata Vantage:

1. If you install Kerberos in a non-standard location, you must edit the TdgssUserConfigFile.xml. Perform steps 1 through 3 of [Changing the TDGSS Configuration](#).
2. The TdgssUserConfigFile.xml does not contain a KRB5 mechanism by default. Instead, go to the TdgssLibraryConfigfile.xml (located in the same directory as the TdgssUserConfigFile.xml), copy the Linux version of the KRB5 mechanism, and paste it into the TdgssUserConfigFile.xml.

---

### Note:

You only need to make a copy of the KRB5 mechanism because you are customizing the Kerberos configuration (in this case you are customizing for a non-standard location). If you using the default configuration of Kerberos you do not need to perform this step.

---

3. After copying the Linux version of KRB5 into the TdgssUserConfigFile.xml, note that the default path to the library is:

```
<RequiredLibrary Path="/usr/lib64/libgssapi_krb5.so"/>
```

On a Teradata Vantage node:

- a. Edit the path value for Linux KRB5 in the TdgssUserConfigFile.xml, to show the new (non-standard) file location used on your system.
- b. Run the run\_tdgssconfig utility to update the TDGSSCONFIG GDO.

```
/opt/teradata/tdgss/bin/run_tdgssconfig
```

---

### Note:

run\_tdgssconfig tells you when a TPA reset is required. For example, if you change <RequiredLibrary>, then you need a TPA reset.

---

- c. If run\_tdgssconfig indicates that a TPA reset is needed, run tpareset to activate the changes to the TDGSS configuration.

```
tpareset -f "use updated TDGSSCONFIG GDO"
```

You only need to perform this procedure once from any node in a Teradata Vantage system. Running the run\_tdgssconfig tool distributes the change to all database nodes.

On Unity servers: See *Teradata® Unity™ Installation, Configuration, and Upgrade Guide for Customers*, B035-2523 for Unity configuration information.

## Working with Kerberos Setup on the KDC

Users accessing a Teradata Vantage system may be authenticated by Kerberos through either a Windows Kerberos or MIT Kerberos KDC. MIT Kerberos KDCs are supported only on Linux.

KDCs must be configured as shown in:

- [Setting Up Kerberos for Teradata Vantage on a Windows Kerberos KDC](#)
- [Setting Up Kerberos on a Linux MIT Kerberos KDC](#)

System conditions affect how you set up a KDC for Kerberos authentication.

- If multiple KDCs can authenticate Vantage users, and the KDCs are in separate domains, you must configure a KDC in each domain.
- If Kerberos-authenticated users can access more than one Vantage system, you must configure all Teradata Vantage systems on each KDC you configure.
- If users log on through Unity you must configure each Unity server on each KDC you configure. If the Unity servers are in different domains, they must trust each other.

## Setting Up Kerberos for Teradata Vantage on a Windows Kerberos KDC

Setting up Kerberos for Teradata Vantage on a Windows Kerberos KDC requires:

1. [Creating a Computer Component for Database Nodes and Unity Server](#)
2. [Checking Node and Unity Registration in the Windows DNS](#), and if necessary, [Adding a Database Node or Unity Server to the Windows DNS](#)
3. [Creating an Active Directory User for Each Node and Unity Server](#)
4. [Determining the SPN for Each Node and Unity Server](#)
5. [Running ktpass to Create the Kerberos Keys](#)
6. [Moving the Kerberos Keys to a Teradata Vantage System or Unity Server](#)

If multiple Windows KDCs in different domains can authenticate Teradata Vantage users, you must duplicate this setup procedure on a Windows KDC in each domain.

## Creating a Computer Component for Database Nodes and Unity Server

You must manually identify all Vantage database nodes, and Unity servers (if used), on each Windows KDC that can authenticate database users. This and subsequent procedures assume that the Windows KDC is an Active Directory server. Requirements are similar for other Windows KDC implementations, although methods may vary.

1. On the Windows KDC (Active Directory server), click on **Start > Programs > Administrative Tools > Active Directory Users and Computers**.
2. Right click on **Computers**.

3. Click on **New > Computer**.
4. Enter the name of the Vantage node or Unity server and click **Next**.

There are no rules for naming a node or server because the name is tied to the IP address in the next task. However, you can use a structure such as `tdatsysa1-1`, where:

- `tdatsysa` is the system name (to differentiate among multiple systems).
- The first `1` denotes a Teradata Vantage system clique ID.
- The second `1` identifies a specific node within the clique.

---

**Note:**

For Unity servers, you can use a naming convention like `unity1`, to differentiate among multiple Unity servers and backups.

---

5. Click **Finish**.

---

**Note:**

Repeat this procedure for each node of each Teradata Vantage system and each Unity server that is served by the KDC. Maintain a list of node, system, and Unity server names for use in subsequent tasks.

---

## Checking Node and Unity Registration in the Windows DNS

Check if Vantage nodes, and the Unity server, if used, are registered in DNS.

1. From the command prompt on a Teradata Vantage Advanced SQL Engine database node, or from the Linux prompt on a Unity server, run `nslookup` using the naming convention for your system:

```
nslookup [node_name|unity_server_name]
```

2. Additional action is based on the output returned by `nslookup`:

- If a node or Unity server is not properly registered in DNS, for example,

```
nslookup tdatsysa1-1
Server:          141.206.3.8
Address:         141.206.3.8#53

** server can't find tdatsysa1-1: SERVFAIL
```

do the following step [Adding Database Nodes and Unity Servers to UNIX DNS](#).

- If all nodes and Unity servers are properly registered in DNS, they appear in the results from the `nslookup` command, for example,

```

nslookup node_name
Server:          141.206.3.8
Address:         141.206.3.8#53

Name: tdatsysa1-1.corp.teradata.com
Address: 141.206.3.58

```

go to [Creating an Active Directory User for Each Node and Unity Server](#).

## Adding a Database Node or Unity Server to the Windows DNS

Add the database nodes and/or Unity server to Windows DNS.

## Creating an Active Directory User for Each Node and Unity Server

Create an Active Directory user for each Teradata Vantage node and each Unity server you added to DNS in [Adding a Database Node or Unity Server to the Windows DNS](#). This step is necessary in determining the Service Principal Name (SPN).

### Note:

The Active Directory user for each database node and Unity server must have a password. Teradata recommends that you use a strong password. If site security policy allows it, you can use the same password for all node and Unity server users.

1. On the Active Directory server, click on **Start > Programs > Administrative Tools > Active Directory Users and Computers**.
2. Right click on the OU where the user is to be created.
3. Click on **New > User**.
4. In the dialog box, enter the name of the node or Unity server for both **User logon name** and **First name**. Use the names assigned in [step 4](#) of [Creating a Computer Component for Database Nodes and Unity Server](#), for example: tdatsysa1-1.
5. Click **Next**.
6. Enter a password and reenter it to confirm.
7. If site policy allows it, check the **Password never expires** box. Otherwise, check **User cannot change password**.
8. Click **Next**.
9. Repeat this procedure for each node of each Teradata Vantage system and each Unity server that is served by the KDC.

**Note:**

If you change the passwords for the users you created, for example, because password change is required by site security policy, you must also:

1. Regenerate the keys. See [Running ktpass to Create the Kerberos Keys](#).
2. Re-install them on the nodes, and on Unity servers, if used. See [Installing the Kerberos Keys](#).

## Determining the SPN for Each Node and Unity Server

Teradata Vantage clients use Service Principal Names (SPNs) to specify the Vantage node or Unity server to which they want to connect.

SPNs use the following format:

```
service_name/instance@realm
```

***service\_name***

Requests the service, that is, a Teradata Vantage or Unity server.

Example: TERADATA

***instance***

Specifies the Fully Qualified Domain Name (FQDN) for a database node or Unity server, composed of:

- A name created in step 4 of [Creating a Computer Component for Database Nodes and Unity Server](#), for example, tdatsysa1-1.
- The domain path to the node or Unity server, taken from step 3 of [Adding a Database Node or Unity Server to the Windows DNS](#), in the form: subdomain.domain.com

**Note:**

The domain information you enter for *instance* is case sensitive. Even if the domain name has upper case characters, you must specify it in the ktpass command using lowercase.

The domain information can include one or more additional sub-domain specifications if required to uniquely locate the node or Unity server.

Examples:

```
tdatsysa1-1.corp.teradata.com
unity1.corp.teradata.com
```

**realm**

The name of the Kerberos realm containing the node or Unity server.

**Note:**

The realm information must match the Windows domain exactly including case.

The realm specified for this term can include one or more additional sub-realm specifications if required to uniquely identify the realm.

Example: CORP.TERADATA.COM

**Note:**

Determine the SPNs for all nodes and Unity servers defined in step 4 of [Creating a Computer Component for Database Nodes and Unity Server](#), for all Teradata Vantage systems and Unity servers served by the KDC. Retain the SPN information for use in [Running ktpass to Create the Kerberos Keys](#).

## Running ktpass to Create the Kerberos Keys

Ktpass is a Microsoft program that creates the encryption keys that the KDC uses to negotiate KRB5 or SPNEGO logon transactions between a Teradata Vantage or Unity server, and the networked clients.

Run ktpass once on the KDC for each Vantage node and Unity server SPN that you created in [Determining the SPN for Each Node and Unity Server](#). Note that the procedure for the first database node is different from the procedure for all other nodes.

**Note:**

After creating the keys, you must copy a set of keytab files for each KDC to each Vantage system and Unity server. See [Installing the Kerberos Keys](#).

## Generating the Key for the First Node or for a Unity Server

Use the ktpass command to create the key for the first Teradata Vantage node in a system, or for a Unity server. The keytab file is created in the directory from which the command is issued unless you use -out keytab\_filename to specify another location.

**Note:**

Generate the keys for each Unity server individually. Key generation for additional Unity servers does not follow the same rules as generating keys for additional nodes in a database system.

```
ktpass -princ spn -mapuser [ node_name | unity_server_name ] -pass password
      -ptype KRB5_NT_PRINCIPAL -out keytab_filename
```

***spn***

The SPN for a Teradata Vantage node or Unity server, as defined in [Determining the SPN for Each Node and Unity Server](#).

***node\_name******unity\_server\_name***

The name of a Teradata Vantage node or Unity server created in step 4 of [Creating an Active Directory User for Each Node and Unity Server](#).

***password***

The Password for the user represented by the node or Unity server name. Use the password assigned to the name in step 6 of [Creating an Active Directory User for Each Node and Unity Server](#).

**KRB5\_NT\_PRINCIPAL**

The principal name type.

The example value, KRB5\_NT\_PRINCIPAL, is the same for all systems.

Specify the -ptype exactly as shown for all Kerberos setups.

***keytab\_filename***

The name of the keytab file to which the keys are written, for example, *domain\_name.sys\_name.keytab*, where:

- *domain\_name* is included to differentiate among the separate keytab files required for multiple domains, if present.
- *sys\_name* names of a Teradata Vantage system or Unity server in the domain.

**Note:**

If the Active Directory KDC serves more than one Vantage system or Unity server, you must create a keytab file for each one using a unique *sys\_name*.



**Note:**

The order in which the ktpass parameters appear is not important.

## Generating the Keys for Additional Nodes

The ktpass command structure for additional Teradata Vantage nodes, is similar to the structure for the first node, except for the added -in parameter, for example:

```
ktpass -princ SPN -mapuser node_name -pass password -ptype
KRB5_NT_PRINCIPAL -out keytab_filename -in keytab_filename
```

where the value for the -in parameter is the same keytab filename you specify for the -out parameter. See the description in [Generating the Key for the First Node or for a Unity Server](#).

## Example: ktpass Commands for a Two Node Setup

The following example shows the ktpass commands that set up a Teradata Vantage system that contains two nodes, node tdatsysa1-1 and tdatsysa1-2.

ktpass command for node tdatsysa1-1:

```
ktpass -princ TERADATA/tdatsysa1-1.corp.teradata.com@CORP.TERADATA.COM
-mapuser tdatsysa1-1 -pass a$2Tya3 -ptype KRB5_NT_PRINCIPAL
-out domain_name.sys_name.keytab
```

ktpass command for node tdatsysa1-2:

```
ktpass -princ TERADATA/tdatsysa1-2.corp.teradata.com@CORP.TERADATA.COM
-mapuser tdatsysa1-2 -pass a$2Tya3 -ptype KRB5_NT_PRINCIPAL -out
domain_name.sys_name.keytab -in domain_name.sys_name.keytab
```

**Note:**

If your system resides in multiple domains, you must generate separate keytab files with unique file names for the Active Directory KDC in each domain. Make a list of the files for use in [Installing the Kerberos Keys](#).

For systems with more than two nodes, you may find it more efficient to incorporate the ktpass commands into a script.

## Moving the Kerberos Keys to a Teradata Vantage System or Unity Server

After you generate Kerberos keys on the KDC(s), you must securely move copies of the set of KDC keytab files for each Teradata Vantage system from the KDC to a temporary location on each node in the corresponding Vantage system, and move Unity server KDC keytab files to the corresponding Unity servers.

### Note:

If a Vantage system or Unity sever resides in multiple domains, you must move the keytab files from the KDC in each domain to the Vantage system and the Unity server. Save the copies of the keytab files here: `/opt/teradata/tdat/tdgss/site/domain_name.sys_name.keytab`.

This is a temporary location to use when you install the keys to the permanent location in [Installing the Kerberos Keys](#). Each keytab file must have a unique file name.

## Setting Up Kerberos on a Linux MIT Kerberos KDC

You must perform the following steps to configure MIT Kerberos on Linux to support Teradata Vantage logons:

1. [Creating Teradata Vantage Node and Unity Server Principals](#)
2. [Checking Registration in UNIX DNS](#)
3. [Adding a Database Node or Unity Server to the Windows DNS](#)
4. [Creating the Kerberos Keys](#)
5. [Copying the Kerberos Keys From the KDC to the Principals](#)

If multiple Linux MIT Kerberos KDCs can authenticate Vantage users, duplicate this setup procedure on each KDC.

## Creating Teradata Vantage Node and Unity Server Principals

You must create a Kerberos principal and password for each node on each Teradata Vantage system, and for each Unity server (if used), that is served by the MIT Kerberos Linux KDC.

Use the `addprinc` command to create the principal and password, for example, for a node:

```
kadmin.local: addprinc TERADATA/principal_name.esrootdom.esdev.tdat

WARNING: no policy specified for TERADATA/
principal_name.esrootdom.esdev.tdat@UNIX.ESROOTDOM.ESDEV.TDAT; defaulting to
no policy
Enter password for principal
"TERADATA/principal_name.esrootdom.esdev.tdat@UNIX.ESROOTDOM.ESDEV.TDAT":
```

```
Re-enter password for principal
"TERADATA/principal_name.esrootdom.esdev.tdat@UNIX.ESROOTDOM.ESDEV.TDAT":
Principal "TERADATA/principal_name.esrootdom.esdev.tdat@
UNIX.ESROOTDOM.ESDEV.TDAT" created.
```

***principal\_name.esrootdom.esdev.tdat***

The FQDN of a Teradata Vantage node or Unity server.

*principal\_name* must use the naming conventions in step 4 of [Creating a Computer Component for Database Nodes and Unity Server](#).

UNIX.ESROOTDOM.ESDEV.TDAT is the Kerberos realm in which the Vantage node or Unity server principal(s) is being added.

The string TERADATA/*principal\_name*.esrootdom.esdev.tdat@UNIX.ESROOTDOM.ESDEV.TDAT, used to represent the principal, also constitutes the SPN for the principal. The SPN is used later in [Creating the Kerberos Keys](#) and [Installing the Kerberos Keys](#) to uniquely identify the keys.

**Note:**

When creating a Unity server principal, the service name is still TERADATA, for example:

```
kadmin.local: addprinc TERADATA/principal_name.esrootdom.esdev.tdat
```

## Password Changes for Unity Principals

If you change the passwords for the Unity server principal, for example, because the password change is required by site security policy, you must:

1. Regenerate the keys. See [Creating the Kerberos Keys](#).
2. Re-install them on the nodes. See [Installing the Kerberos Keys](#).

## Checking Registration in UNIX DNS

You should check to make sure that Teradata Vantage nodes and Unity servers are registered in DNS. The procedure is similar to [Checking Node and Unity Registration in the Windows DNS](#).

If any nodes or Unity servers are not yet registered in UNIX DNS, go to the next step, [Adding Database Nodes and Unity Servers to UNIX DNS](#).

## Adding Database Nodes and Unity Servers to UNIX DNS

On the Unix DNS server:

1. Add database nodes and Unity servers to DNS, including specification of the A, CNAME, and PTR records.

2. After completing changes to the files, test the registration as shown in [Checking Registration in UNIX DNS](#).

## Creating the Kerberos Keys

Use the `ktadd` command in `kadmin.local` to create the keytab file to contain the Teradata Vantage node and Unity server keys. For example, for a Vantage node:

```
kadmin.local: ktadd -k /etc/principal_name.keytab TERADATA/
principal_name.esrootdom.esdev.tdat@UNIX.ESROOTDOM.ESDEV.TDAT

Entry for principal TERADATA/
principal_name.esrootdom.esdev.tdat@UNIX.ESROOTDOM.ESDEV.TDAT with kvno
2, encryption type AES-256 CTS mode with 96-bit SHA-1 HMAC added to keytab
WRFILE:/etc/principal_name.keytab.
Entry for principal TERADATA/
principal_name.esrootdom.esdev.tdat@UNIX.ESROOTDOM.ESDEV.TDAT with kvno
2, encryption type AES-128 CTS mode with 96-bit SHA-1 HMAC added to keytab
WRFILE:/etc/principal_name.keytab.
Entry for principal TERADATA/
principal_name.esrootdom.esdev.tdat@UNIX.ESROOTDOM.ESDEV.TDAT with kvno
2, encryption type Triple DES cbc mode with HMAC/sha1 added to keytab
WRFILE:/etc/principal_name.keytab.
Entry for principal TERADATA/
principal_name.esrootdom.esdev.tdat@UNIX.ESROOTDOM.ESDEV.TDAT with kvno
2, encryption type ArcFour with HMAC/md5 added to keytab
WRFILE:/etc/principal_name.keytab.
```

### ***principal\_name.esrootdom.esdev.tdat***

The FQDN of a Teradata Vantage node or Unity server.

*principal\_name* must use the naming conventions in step 4 of [Creating a Computer Component for Database Nodes and Unity Server](#).

UNIX.ESROOTDOM.ESDEV.TDAT is the Kerberos realm in which the Vantage node or Unity server principal(s) is being added.

---

### **Note:**

When creating Kerberos keys for a Unity server principal, the service name is still TERADATA, for example:

```
kadmin.local: ktadd -k /etc/unity_server_name.keytab
TERADATA/unity_server_name.esrootdom.esdev.tdat@UNIX.ESROOTDOM.ESDEV.TDAT
```

---

## Copying the Kerberos Keys From the KDC to the Principals

After you generate Kerberos keys on the Linux MIT KDC(s), you must securely move copies of the set of keytab files for database nodes from the KDC to a temporary location on any node of the corresponding database system, and move copies of the set of keytab files for each Unity server (if used) to the corresponding server.

---

### Note:

If a database system or Unity server resides in multiple domains, make sure you move the keytab files from the KDC in each domain. Save the copies of the keytab files here: `/opt/teradata/tat/tgss/site/domain_name.sys_name.keytab`.

`domain_name.sys_name` is defined in [Generating the Key for the First Node or for a Unity Server](#).

---

This is a temporary location to use when you install the keys to the permanent location in [Installing the Kerberos Keys](#). Make sure that each keytab file has a unique file name.

## Configuring Teradata Vantage and Unity Servers for Kerberos Authentication

Execute these procedures on Teradata Vantage nodes and Unity servers to prepare for Kerberos authentication.

1. [Verifying that a Database Node or Unity Server Can Find the Name Server](#)
2. [Determining the Kerberos Key Installation Directory](#)
3. [Installing the Kerberos Keys](#)
4. [Synchronizing the Time and Date Within the Domain](#)
5. [Checking the Kerberos Setup](#)

## Verifying that a Database Node or Unity Server Can Find the Name Server

You can verify that a database node or Unity server can find the Name Server by running `nslookup` from the command prompt:

```
nslookup [node_name|unity_server_name]
```

where the specified name is not the name of the node or Unity server from which you run the `nslookup` command.

The `nslookup` fails if the specified node or Unity is unable to find the Name Server.

The `nslookup` output verifies that the `/etc/resolv.conf` file is set up properly, for example:

```
domain esrootdom.esdev.tdat
search esrootdom.esdev.tdat
nameserver 141.200.3.8
```

where esrootdom.esdev.tdat is the Kerberos domain.

## Setting Up the krb5.conf Kerberos Configuration File

The krb5.conf Kerberos configuration file requires a special setup on each database node, and on the Unity server, if used. The sample krb5.conf file is located by default in the /etc directory.

---

### Note:

MIT Kerberos contains several sections and tags that are not required for Teradata Vantage nodes or the Unity server, and are not shown in the configuration that follows. Teradata recommends that you do not use these options for Kerberos implementation unless it is absolutely necessary, and you have fully researched the effects.

---

For more detailed information on MIT Kerberos sections and tags, go to:

<http://web.mit.edu/Kerberos/krb5-1.5/krb5-1.5.3/doc/krb5-admin/krb5.conf.html>

### Example: krb5.conf

The example below shows the structure of the sample krb5.conf file. You must modify the example to conform to the requirements of your system, using the guidelines contained in the syntax table that follows the example.

```
[libdefaults]
    default_realm = default_kerberos_realm
    clockskew = allowable_skew

[realms]
    default_kerberos_realm = {
        kdc = kdchost1_fqdn
        kdc = kdchost2_fqdn
        kdc = kdchost3_fqdn
    }

[domain_realm]
    host_dnsdomain = kerberos_realm
    host_fqdn = kerberos_realm

[logging]
    kdc = FILE:/tmp/krb5kdc.log
```

```
[appdefaults]
pam = {
    ticket_lifetime = ticket_duration
    renew_lifetime = renew_duration
    forwardable = true/false
    proxiable = true/false
    retain_after_close = false
    minimum_uid = 0
    try_first_pass = true
}
```

Section, TagName, and TagValue	Description
[libdefaults]	The section that contains default values that the Kerberos library uses to authenticate a logon.
default_realm = <i>default_kerberos_realm</i>	<p>The realm that contains the Kerberos logon, including both the KDC host (Windows domain controller) and the Vantage nodes, for example: SUBDOMAIN.DOMAIN.COM</p> <p><b>Note:</b> The realm information must match the Windows domain name exactly, including case.</p>
clockskew = <i>allowable_skew</i>	<p>The maximum allowable difference, in seconds, for time synchronization between Vantage and the client domain. The maximum suggested value is +/- 300 (five minutes).</p> <p>You must enter this value as a whole, positive integer.</p>
[realms]	Subsections keyed by Kerberos realm names. Each subsection describes realm-specific information, including where to find the Kerberos servers for that realm.
default_kerberos_realm = {	See default_realm = <i>default_kerberos_realm</i> above.
kdc = <i>kdchost1_fqdn</i> kdc = <i>kdchost2_fqdn</i> kdc = <i>kdchost3_fqdn</i>	<p>Required. The KDC host is a domain controller for the Windows domain. The FQDN is similar to: hostname.subdomain.domain.com</p> <p><b>Note:</b> Only one KDC host may be required, but if you configured Vantage system nodes in multiple domains <a href="#">Working with Kerberos Setup on the KDC</a>, you need to define a KDC host for each domain.</p>
additional_kerberos_realm = }	<p>Required if realms other than the default Kerberos realms contain functioning KDC hosts, for example: ALTSUBDOMAIN.DOMAIN.COM</p>

Section, TagName, and TagValue	Description
	Specify the realm according to the rules for <i>default_kerberos_realm</i> shown above.
<i>kdc = additional_kdchost_fqdn</i>	<p>Required, if there is an additional KDC host. The FQDN of the additional KDC host, for example: additionalhostname.subdomain.domain.com</p> <p><b>Note:</b> The KDC host is an alternate domain controller for the Windows domain.</p>
[domain_realm]	The section that contains relationships that map domains and subdomains onto Kerberos realm names. This determines a host realm location by its FQDN.
<i>host_dnsdomain = kerberos_realm</i>	<p>Required. Maps the DNS domain containing one or more hosts, for example: .subdomain.domain.com to the Kerberos realm, for instance, SUBDOMAIN.DOMAIN.COM</p> <p>The leading dot in the <i>host_dnsdomain</i> expression indicates that the expression maps all hosts that reside in the domain to the Kerberos realm.</p> <p><b>Note:</b> Specify the DNS domain in lower case. The Kerberos realm is case sensitive, and must exactly match the Windows domain.</p>
<i>host_fqdn = kerberos_realm</i>	<p>Required. Maps a specific host FQDN (Teradata Vantage node), for example: subdomain.domain.com to the Kerberos realm, for example: SUBDOMAIN.DOMAIN.COM</p> <p>The lack of a leading dot in the <i>host_fqdn</i> expression indicates that the expression maps only the host with the exact specified FQDN to the Kerberos realm.</p> <p><b>Note:</b> The <i>host_fqdn</i> value is case sensitive. The <i>kerberos_realm</i> value is not case sensitive, but must exactly match the Windows domain.</p>
[logging]	The section that contains instructions for Kerberos logging.
<i>default = FILE:/tmp/krb5lib.log</i>	<p>Recommended. The location of the default Kerberos log on the Vantage node.</p> <p><b>Note:</b> The file location can be expressed as either:</p> <ul style="list-style-type: none"> <li>• FILE: appends each new log entry to the existing log file.</li> <li>• FILE= overwrites the previous log entry.</li> </ul>
[appdefaults]	Each tag in this section specifies an application or option. The tag value defines the behavior of the owning application.



Section, TagName, and TagValue	Description
pam = {	Identifies the start of a list of settings for the PAM application, which defines security policy parameters. Vantage installs PAM when it initially configures the system.  <b>Note:</b> Do not change any of the settings in this list.
ticket_lifetime = <i>ticket_duration</i>	
renew_lifetime = <i>renew_duration</i>	
forwardable = true /false	
proxiabile = true /false	
retain_after_close = false	
minimum_uid = 0	
try_first_pass = true	

## Determining the Kerberos Key Installation Directory

You must determine the file location for [Installing the Kerberos Keys](#) on database nodes and on each Unity server (if used), and then use the location when setting of the value of the TeradataKeyTab property. See [TeradataKeyTab](#).

### Note:

The value of TeradataKeyTab in the KRB5 mechanism affects all Kerberos authentication, including logons that use the SPNEGO mechanism.

The default value for the TeradataKeyTab property is /etc/teradata.keytab. Use this default location unless there is a reason to not do so. If you need to use a custom install location due to site policy requirements, you must also change the value of the TeradataKeyTab property to match the custom location.

### Note:

The Linux user under which Teradata Vantage and Unity run must have read access to the keytab file. If you specify a custom (non-default) file location, you must grant read access permission for the file to the Linux user.

## Installing the Kerberos Keys

The method you use to install Kerberos keys depends on the type of installation, for example:

- [Initial Installation of Kerberos Keys for the First KDC.](#)
- [Installing Kerberos Keys for Additional KDCs \(Merging Keys\).](#)
- [Replacing Existing Kerberos Keys Versus Merging Keys.](#)

**Note:**

You may find it useful to check the status of Kerberos keys on the Teradata Vantage system and Unity server (if used), to help determine the appropriate installation method, before installing a set of keys. See [Checking Nodes and Unity Servers for Existing Kerberos Keys \(Optional\)](#).

If Vantage users can be authenticated in more than one domain, you must install a set of keytab files for a KDC in each domain. If your site requires the installation of multiple keytab files, verify that each file has a unique keytab file name.

**Note:**

If you change the database node or Unity server passwords for the user accounts you created on the KDC(s), for example, because password change is required by site security policy, you must regenerate the keys and re-install them on the nodes.

## Checking Nodes and Unity Servers for Existing Kerberos Keys (Optional)

Any Kerberos keys that already exist in a node or Unity server keytab file could be overwritten (destroyed) when you install new keys. When replacing existing keys, overwriting is normal. However, if you want to retain and add to the existing keys, you must use the key merge procedure, which avoids overwriting.

You can use the `pcl` command to find and display any Kerberos keys that already exist on database nodes or a Unity server to help determine if you should use the merge procedure when installing new keys:

```
pcl -s klist -ke [keytab_file_name]
```

This example keytab file (standard location) shows a two-node system, with pre-existing keys in ***bold italics***:

```
13592:/ > pcl -s klist -ke /etc/teradata.keytab
All 2 node(s) have connected
<----- node_name2_bynet ----->
Keytab name: FILE:/etc/teradata.keytab
KVNO Principal
-----
14  TERADATA/L3592.esrootdom.esdev.tdat@ESROOTDOM.ESDEV.TDAT (DES cbc mode with RSA-MD5)
13  TERADATA/L3593.esrootdom.esdev.tdat@ESROOTDOM.ESDEV.TDAT (DES cbc mode with RSA-MD5)<----- node_name1_bynet ----->
```

```

Keytab name: FILE:/etc/teradata.keytab
KVNO Principal
-----
    14  TERADATA/L3592.esrootdom.esdev.tdat@ESROOTDOM.ESDEV.TDAT (DES cbc mode
with RSA-MD5)
    13  TERADATA/L3593.esrootdom.esdev.tdat@ESROOTDOM.ESDEV.TDAT (DES cbc mode
with RSA-MD5)-----

```

If no keys are present, the output appears without the key entries:

```

13592:/ > pcl -s klist -ke /etc/teradata.keytab
All 2 node(s) have connected
<----- node_name2_bynet ----->
Keytab name: FILE:/etc/teradata.keytab
KVNO Principal
-----
<----- node_name1_bynet ----->
Keytab name: FILE:/etc/teradata.keytab
KVNO Principal
-----

```

---

**Note:**

The key files are similar on a Unity server.

---

## Initial Installation of Kerberos Keys for the First KDC

This procedure copies the Kerberos keys for the first KDC from the temporary location used in [Moving the Kerberos Keys to a Teradata Vantage System or Unity Server](#) to the permanent location (/etc/teradata.keytab) on a Teradata Vantage system or on a Unity server.

On a single node Vantage system or a Unity server:

- Log on to the database node or Unity server:
  - From the database node console command prompt log on as teradata or another user with permission to run utilities.
  - From the Unity server log on as root.
- Copy the temporary keytab file from the temporary location shown in [Moving the Kerberos Keys to a Teradata Vantage System or Unity Server](#) to the permanent location chosen in [Determining the Kerberos Key Installation Directory](#), for example, the default permanent location:

- `cp /opt/teradata/tdat/tdgss/site/  
domain_name.sys_name.keytab /etc/teradata.keytab`

`domain_name.sys_name` is defined in [Generating the Key for the First Node or for a Unity Server](#).

---

**Note:**

If you use a custom location, be sure to specify the custom location as the `TeradataKeyTab` property value for the KRB5 mechanism.

---

3. Display a list of Kerberos keys to verify that all keys installed correctly:

```
klist -ke /etc/teradata.keytab
```

4. After verifying that all keys are installed correctly to the permanent location, delete the key file from the temporary location.

For multi-node Teradata Vantage systems:

1. From a database node console command prompt, log on to the Vantage node that has the temporary keytab file; log on as the user "teradata" or another user with permission to run utilities.
2. Copy the generated keytab file from the temporary location to /etc.
3. Distribute the keytab file to all nodes, using the `pcl` command. For example, send the file from the temporary location to /etc on the other nodes:

```
pcl -send <temporary_location>/teradata.keytab /etc/teradata.keytab
```

---

**Note:**

If you put the keytab file in a location other than /etc, be sure to specify the custom location as the `TeradataKeyTab` property value for the KRB5 mechanism.

---

4. Display a list of Kerberos keys to verify that all keys installed correctly:

```
pcl -s klist -ke /etc/teradata.keytab
```

5. After verifying that all keys are installed correctly to the permanent location, delete the key file from the temporary location.

## Installing Kerberos Keys for Additional KDCs (Merging Keys)

If one or more sets of Kerberos keys are already installed to the permanent keytab file location and you want to add another set of keys, for example, because you configured an additional KDC, you must install the additional keys so that they merge with the existing keys.

1. Run `ktutil` from the command prompt of the database node containing the existing keytab files, or from the Linux prompt of the Unity server with existing keys:

```
ktutil
```

**Note:**

For information on ktutil options, see the ktutil man page on any node or on the Unity server.

- At the ktutil prompt, enter the command to read the current keys:

```
rkt /etc/teradata.keytab
```

**Note:**

This procedure assumes that any existing keytab files are in the standard location. If an alternate location was used, it is shown in the value of the TeradataKeyTab property in the TdgssUserConfigFile.xml.

- Enter the command to read the new keys:

```
rkt /opt/teradata/tdat/tdgss/site/keytab_filename
```

where *keytab\_filename* is the name of a keytab file that you generated in [Running ktpass to Create the Kerberos Keys](#) or [Creating the Kerberos Keys](#), and stored on a database node or Unity server in [Moving the Kerberos Keys to a Teradata Vantage System or Unity Server](#).

**Note:**

If you are installing keys for more than one domain, rerun this step for each set of files, for example, *domain2.sys\_name.keytab*, *domain3.sys\_name.keytab*, and so on.

- List all keys to verify rkt has read all the new files:

```
list
```

- Save all keys:

```
wkt /etc/teradata.keytab
```

- Exit the command:

```
quit
```

- From the Teradata command prompt, distribute the merged keytab file to all nodes, using the pcl command. The new merged file, containing pre-existing and new keys, replaces the old file containing only pre-existing keys on all nodes. For example:

```
pcl -send /etc/teradata.keytab /etc/teradata.keytab
```

**Note:**

Step 7 is not required for a single node database system or Unity servers.

## Replacing Existing Kerberos Keys Versus Merging Keys

If you need to replace existing Kerberos keys with new keys, for example, when site security policy requires periodic key updates, you can overwrite the existing keys during installation.

1. Install new keys for the first KDC as shown in [Initial Installation of Kerberos Keys for the First KDC](#).
2. The installation overwrites all key sets in the file for all nodes to which you distribute the keys.

**Note:**

If you have new keys for additional KDCs, install the remaining key sets as shown in [Installing Kerberos Keys for Additional KDCs \(Merging Keys\)](#) to merge the additional key sets with the first replacement set installed in step 1 above.

## Synchronizing the Time and Date Within the Domain

Kerberos requires that the time and date are synchronized across the domain. The most efficient method is to use a time synchronization package, such as Network Time Protocol (NTP), to synchronize the nodes to the Domain Controller that acts as the Kerberos KDC. For further information on NTP, go to <http://www.ntp.org>.

## Synchronizing Time on Database Nodes and Unity Servers with Time on the KDC

1. Navigate to the /etc directory and find the sample NTP configuration file, ntp.conf. The sample configuration file is similar to:

```
## server 127.127.8.0 mode 5 prefer

##
## Undisciplined Local Clock. This is a fake driver intended for backup
## and when no outside source of synchronized time is available.
##
#server 127.127.1.0          # local clock (LCL)
#fudge 127.127.1.0 stratum 10 # LCL is unsynchronized

##
## Outside source of synchronized time
##
```

```
# server xx.xx.xx.xx    #IP address of server

##
## Miscellaneous stuff
##

#driftfile /var/lib/ntp/drift/ntp.drift
# path for drift file

#logfile /var/log/ntp
# alternate log file
# logconfig =syncstatus + sysevents
# logconfig =all

# statsdir /tmp/                # directory for statistics files
# filegen peerstats  file peerstats  type day enable
# filegen loopstats  file loopstats  type day enable
# filegen clockstats file clockstats type day enable

#
# Authentication stuff
#
# keys /etc/ntp.keys            # path for keys file
# trustedkey 1 2 3 4 5 6 14 15 # define trusted keys
# requestkey 15                # key (7) for accessing server variables
# controlkey 15                # key (6) for accessing server variables
```

2. Copy the sample NTP configuration and uncomment the lines that contain server xx.xx.xx.xx, driftfile, and logfile.
3. On the server line, substitute the IP address for the KDC to which you want to synchronize the Teradata Vantage or Unity server, and add additional server lines for any other domain controllers that act as additional KDCs.

You can use the following example to configure your file:

---

**Note:**

Make sure to enter data that is valid for your system in all uncommented lines.

---

```
## server 127.127.8.0  mode 5 prefer

##
## Undisciplined Local Clock. This is a fake driver intended for backup
## and when no outside source of synchronized time is available.
##
```

```

#server 127.127.1.0          # local clock (LCL)
#fudge 127.127.1.0 stratum 10 # LCL is unsynchronized

##
## Outside source of synchronized time
##
server 141.206.3.8 # IP address of server

##
## Miscellaneous stuff
##

driftfile /var/lib/ntp/drift/ntp.drift # path for drift file

logfile /var/log/ntp
# alternate log file
# logconfig =syncstatus + sysevents
# logconfig =all

# statsdir /tmp/          # directory for statistics files
# filegen peerstats file peerstats type day enable
# filegen loopstats file loopstats type day enable
# filegen clockstats file clockstats type day enable

#
# Authentication stuff
#
# keys /etc/ntp.keys      # path for keys file
# trustedkey 1 2 3 4 5 6 14 15 # define trusted keys
# requestkey 15          # key (7) for accessing server variables
# controlkey 15          # key (6) for accessing server variables

```

4. Save the NTP configuration file in /etc/ntp.conf.
5. To initiate the synchronization, enter the following:

```
node_name:/# /etc/init.d/ntp start
```

6. The synchronization process takes a few minutes to complete. To view the sequence as it takes place, enter:

```
ntpq -p
```

The system outputs something similar to:



```

ntpq -p
remote          refid      st t when poll reach  delay  offset  jitter
=====
*tuesday700.td.te .GPS.      1 u  896 1024 377   77.267   5.317   0.193
LOCAL(0)        LOCAL(0) 10 l   38   64 377    0.000    0.000   0.001

```

7. Check the log file to verify that all the identified directory servers are synchronized. The log file looks similar to:

```

Oct 16 12:42:55 node_name ntpd[12387]: ntpd 4.2.0a@1.1213-r Tue Nov  8
17:39:08 UTC 2005 (1)
Oct 16 12:42:55 node_name ntpd[12387]: precision = 1.000 usec
Oct 16 12:42:55 node_name ntpd[12387]: Listening on interface
wildcard, 0.0.0.0#123
Oct 16 12:42:55 node_name ntpd[12387]: Listening on
interface wildcard, ::#123
Oct 16 12:42:55 node_name ntpd[12387]: Listening on interface
lo, 127.0.0.1#123
Oct 16 12:42:55 node_name ntpd[12387]: Listening on interface
eth0, 141.206.28.199#123
Oct 16 12:42:55 node_name ntpd[12387]: kernel time sync status 0040

```

An entry appears for each synchronized directory server (KDC) and time server, identified by its IP address), as shown in the last line of log file.

## Checking the Kerberos Setup

After you synchronize each Teradata Vantage and Unity server (if used) with the related KDC(s), you can use any of several methods to verify that the configuration functions correctly.

## Using kinit to Test Communication with the Directory

Use the kinit utility to verify that the Teradata Vantage system or Unity server user in the directory can authenticate to the KDC. For example, for a Vantage node user:

1. From the command prompt, run the kinit utility:

```

node_name: >kinit domain_username
Password for domain_username@KERBEROS_REALM: <enter password>

```

2. From the command prompt, run klist to see if a ticket was issued:

```

node_name:>klist -c

```

The output is similar to:

```

Ticket cache: File:/temp/krb5cc_0
Default principal:node_name@KERBEROS_REALM

Valid starting    Expires    Service Principal
09/20/06 11:47:23 09/20/06 21:48:04 krbtgt/KERBEROS_REALM@KERBEROS_REALM
    renew until 9/21/06 11:47:23
Kerberos 4 ticket cache: /tmp/tkt()
klist: You have no tickets cached

```

**Note:**

Testing directory communication with a Unity server is similar.

## Using klist -e to Check the Credentials Cache and Encryption Type

You can also use klist -e to check the tickets in the credentials cache and the type of encryption that the credentials use:

```
node_name: >klist -e
```

The output is similar to:

```

Ticket cache: FILE:/tmp/krb5cc_0
Default principal: tester2@KERBEROS_REALM

Valid starting    Expires    Service principal
10/15/10 16:05:09 10/16/10 02:05:10 krbtgt/
ESROOTDOM.ESDEV.TDAT@ESROOTDOM.ESDEV.TDAT
    renew until 10/16/10 16:05:09, Etype (skey, tkt): ArcFour with
HMAC/md5, ArcFour with HMAC/md5

```

## Logging on to the Database to Check Kerberos Authentication

When you verify that the Kerberos setup is complete, log on from a Vantage client using BTEQ or another application and specify the KRB5 mechanism for all logon variations that your site allows, both directly to the database and through Unity (if used), to verify that Kerberos setup parameters are correct.

For information on allowable Kerberos logon variations, see the sections on Single Sign-on and Sign-on As in [Logging on to Teradata Vantage](#).

# Configuring Single Sign-On

Teradata supports Single Sign-On (SSO) so users do not have to enter credentials multiple times to access different applications such as the database.

In a non-browser Vantage environment, you can exchange the username and password with a token using an OAuth flow API. Vantage is based on the OpenID Connect (OIDC) protocol which uses JSON Web Tokens (JWT) delivered via OAuth 2.0. OAuth 2.0 provides resource access and sharing and OIDC provides user authentication. For on-premises Vantage systems, this functionality is provided by a central broker identity provider.

SSO authentication with the IdP can only be done in a browser from which an OpenID Connect is initiated and an authentication token (JWT) is made available to the applications and database.

## Single Sign-On Flow

JWT validation is done in TDGSS during connection establishment. TDGSS gets the JWT token and validates its signature.

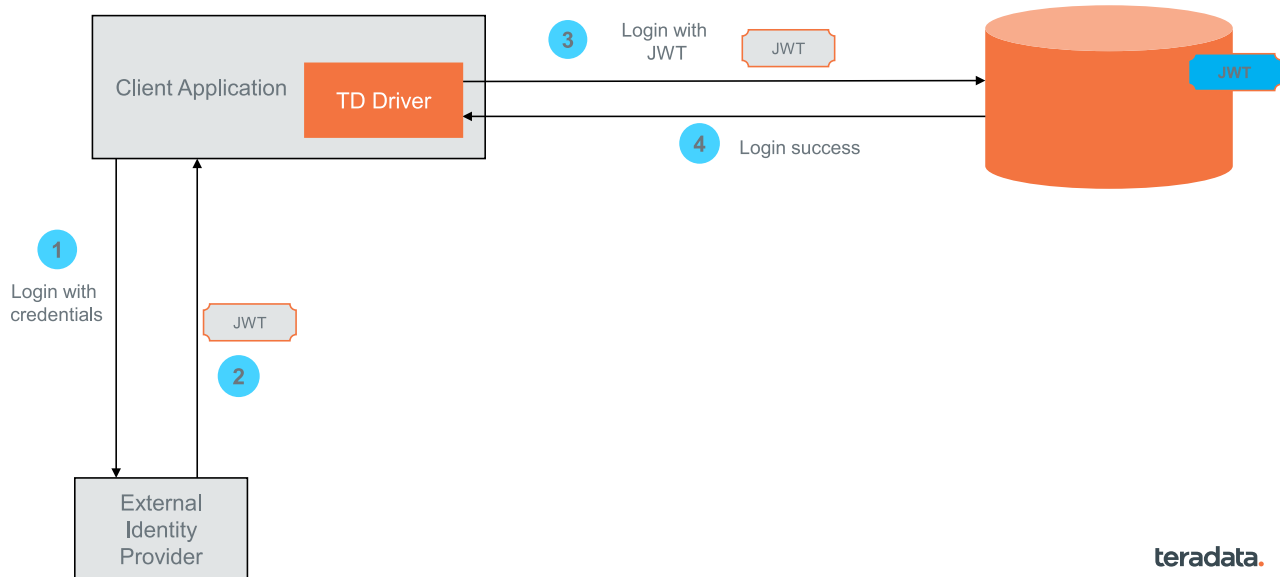
JWT (more precisely compact JWS) consists of three parts: header, payload, and signature. The payload contains a set of claims in a JSON object. The Issuer (“iss”) claim identifies the URI of the identity provider that issued the JWT.

There are two validation cases:

- **Local Validation:** TDGSS receives the JWT minted by an internal Central IdP.
- **Validation by Token Exchange:** TDGSS receives the JWT minted by an external IdP.

## Local Validation

## Token exchange



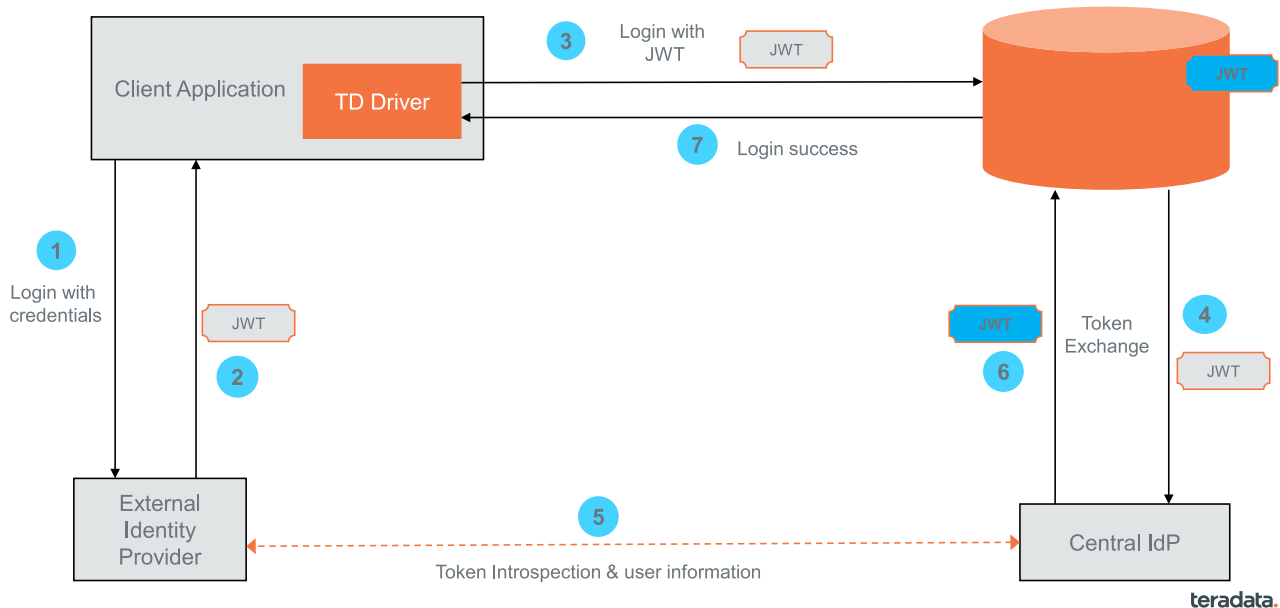
For local validation, the client application attempts to authenticate to the database as follows:

1. When the client authenticates, the Gateway sends the client a configuration response containing the ClientId (such as sso-dev) and the IdpUrl (such as <https://sso-idp-dev.iam.teradatacloud.io/.well-known/openid-configuration>). This information is defined in the TdgssUserConfigFile.xml in the <GlobalValues> section.
2. The client then requests a JWT token from the external IdP.
3. The client sends the JWT to the Gateway to log the session on.
4. The Gateway validates the token:
  - a. TDGSS peeks into the payload to get the issuer claim.
  - b. TDGSS gets the External IdP issuer claim from the TDGSS configuration.
  - c. TDGSS compares the External IdP issuer and the JWT “iss” claim.
5. **Local Validation:** If the two issuers match, TDGSS validates the connection locally. Gateway validates with a key that corresponds to the token. The key is retrieved from the JWK URI, which is published by IdP as OpenId Connect configuration (<https://<External IdP url>/.well-known/openid-configuration>).

To configure the JWT mechanism for local validation, see [Local Validation](#).

## Validation by Token Exchange

### Token exchange



For external authentication, TDGSS exchanges the JWT with Central IdP, and after the exchange TDGSS validates the user. This is done by identifying if the JWT comes from Central IdP or from an external identity provider. TDGSS uses the issuer “iss” claim in the JWT payload to identify the identity provider. The client application attempts to authenticate to the database as follows:

1. When the client authenticates, the Gateway sends the client a configuration response containing the ClientId (such as sso-dev) and the IdpUrl (such as <https://sso-idp-dev.iam.teradatacloud.io/.well-known/openid-configuration>). This information is defined in the TdgssUserConfigFile.xml in the <GlobalValues> section.
2. The client then requests a JWT token from the external IdP.
3. The client sends the JWT to the Gateway to log the session on.
4. The Gateway validates the token:
  - a. TDGSS peeks into the payload to get the issuer claim.
  - b. TDGSS gets the Central IdP issuer claim from the TDGSS configuration.
  - c. TDGSS compares the Central IdP issuer and the JWT “iss” claim.
5. **Validation by Token Exchange:** If the issuers don't match, a token exchange is done:
  - a. From the configuration, TDGSS gets the issuer of the configured external identity provider.
  - b. TDGSS compares the external IdP issuer and the “iss” claim from the JWT.
  - c. If they don't match, an error is generated and the user is not authenticated.
  - d. If they match, a REST API call is made to exchange the JWT from the Central IdP.

6. After exchanging the JWT from Central IdP, TDGSS validates the JWT.
7. The JWT is saved in the session control table to use it to authenticate to other Vantage services.

To configure the JWT mechanism for token exchanger validation, see [Validation by Token Exchanger](#).

The following table shows the required information to get an exchange token from the central IdP:

Parameter	Description
subject_issuer	The alias name with which external IdP (the access token issuer), is registered with the central IdP. This parameter is required only for the central IdP.
subject_token	A valid JWT access token issued by the external IdP.
client_id	The client id of the target application registered with the central IdP that intends to use the exchanged token. The mappers associated with client_id decide the content of the exchanged token.
client_secret	The client secret of the target application.
subject_token_type	Constant value: urn:ietf:params:oauth:token-type:access_token
grant_type	Constant value: urn:ietf:params:oauth:grant-type:token-exchange
token_endpoint	The /token endpoint of the central IdP. It is usually a URL in this form: https://<idp-host>/auth/realms/<realm name>/protocol/openid-connect/token

## Configuration for Browser Authentication

To set up browser authentication, you must configure TDGSS so the client is provided metadata from the Gateway, specifically the client needs IdpUrl and ClientId from the `<GlobalValues>` section of `TdgssUserConfigFile.xml`.

To configure TDGSS to provide the values:

1. Make a backup copy of the `/opt/teradata/tdat/tdgss/site/TdgssUserConfigFile.xml` and save it according to your site standard backup procedures.
2. Edit `TdgssUserConfigFile.xml`. Uncomment the `<GlobalValues>` section and add values for the `IdpUrl` and `ClientId` properties:

```
<TdgssConfigFile>
  <Header
    Version="1"
    ConfigFileType="User">
  </Header>
  <!--
    To enable, uncomment the GlobalValues section and fill in the
```

```

        IdpUrl and ClientId attributes. When backing down to an earlier
        version (e.g. 17.0), comment this entire section out.
    <GlobalValues>
        <IdpConfig
            IdpUrl=""
            ClientId=""
        />
    </GlobalValues>
    -->

```

Where the *<GlobalValues>* section properties are:

Property	Description
<i>IdpUrl</i>	Refers to the configured external identity provider. Example: IdpUrl="https://sso-idp-dev.iam.teradatacloud.io/.well-known/openid-configuration"
<i>ClientId</i>	The ID of the Gateway that is used during the token exchange. Example: ClientId="sso-dev"

3. If run\_tdgssconfig indicates that a TPA reset is required, run:

```
tpareset -f "use updated TDGSSCONFIG GDO"
```

## Configuration for OpenID Connect

### Local Validation

The JWT mechanism can be configured to dynamically update JSON Web Keys (JWKs).

When logging on to Teradata Vantage using JWT, an Identity Provider (IdP) signs the token using its private key and the Teradata server verifies the token's signature using the corresponding public key. Keys returned from IdPs are in JSON Web Key (JWK) format. Keys used to validate the token can be in in PEM or JWK format.

Key pairs are rotated by the IdP for various reasons, such as security policy or a compromised key. If a key is rotated by the identity provider the Teradata Gateway must update the public keys to validate the token issued with the new keys. This is done automatically if dynamic key rotation is enabled.

By default, this feature is disabled.

To enable dynamic key rotation:

1. Make a backup copy of the /opt/teradata/tdat/tdgss/site/TdgssUserConfigFile.xml and save it according to your site standard backup procedures.

2. Edit `TdgssUserConfigFile.xml` and uncomment `IdentityProvider`. Edit it so it looks similar to the following with your site information:

```
<Mechanism Name="JWT">
  <MechanismProperties
    JWTDynamicKey ="yes|no"
    JWTClientTlsCACertDir="ca_cert_dir"

    .../>

  <IdentityProvider
    Id="Keycloak"
    Url="https://Customer_IdP_URL"
    Type="keycloak"
  />
</Mechanism>
```

Set the following properties:

Property	Description
<i>JWTDynamicKey</i>	Set it to yes to enable dynamic key rotation.
<i>JWTClientTlsCACertDir</i>	Location of the CA certificates. Specify the full path to <code>site/ssl/cacerts</code> directory for this property. For example: <code>/opt/teradata/tdat/tdgss/site/ssl/cacerts/</code>

Set `<IdentityProvider>` section properties:

Property	Description
<i>Id</i>	Uniquely identifies the IdP in the configuration file.
<i>Url</i>	Url is the customer's IdP end point. From the URL, TDGSS can issue REST API calls to get the required URLs and other information, such as Issuer, JWK URI, and so on. Based on the Url, TDGSS decides whether to establish a TLS connection with the IdP. For a TLS connection <i>JWTClientTlsCACertDir</i> is the directory where all the CA certificates are configured.
<i>Type</i>	Type is the Identity Provider type. Example values are: Ping-Federate, keycloak, vantage-keycloak, azuread, okta, and auth0.

3. Place the CA certificates in the location specified in *JWTClientTlsCACertDir*. This directory is typically here: `/opt/teradata/tdat/tdgss/site/ssl/cacerts`.
4. Verify the configuration is correct:



- a. Run `tdgsstestcfg` to verify the new configuration is correct. It launches a test environment in a new shell that contains the updates to the configuration file.

```
/opt/teradata/tdgss/bin/tdgsstestcfg
```

- b. Test the configuration with the `tdgssauth` tool:

```
tdgssauth -m JWT -a token=JWT_from_IdP
```

Where *JWT\_from\_IdP* is the IdP you configured in `TdgssUserConfigFile.xml`.

- c. Exit the test shell:

```
exit
```

- d. Continue editing and testing until the configuration is correct.

## 5. Run:

```
/opt/teradata/tdgss/bin/run_tdgssconfig
```

6. If `run_tdgssconfig` indicates that a TPA reset is required, run:

```
tpareset -f "use updated TDGSSCONFIG GDO"
```

## Validation by Token Exchanger

You can configure the JWT mechanism to accept JWT logons from third-party applications. For example, a user logs into a web app from a browser. The web app federates the logon to the customer's IdP. Then, if the user connects to Teradata Vantage the web app provides the JWT token to the Teradata server to successfully complete the logon.

By default, this feature is disabled.

To enable the feature, two identity providers are configured:

- **Internal IdP** (called `ping-internal` in the example config file): Set *Type* to `Ping-Federate` and set *ValidateByTokenExchange* to `no` for the internal IdP.

If the JWT is minted by this IdP, validation can be done without exchanging a token. In this case, the JWT is coming from an internal application.

- **External IdP** (called `keycloak21` in the example config file): Set *Type* to `keycloak` and set *ValidateByTokenExchange* to `yes` for the external IdP.

If the JWT is minted by this IdP, first a token exchange is done, then the token is validated.

In the `TdgssUserConfigFile.xml`, the `<TokenExchanger>` section has *Ref* set to `ping-internal` so it is pointing to the `<IdentityProvider>` section for the internal IdP, the one with *Id* set to `ping-internal` in this case.

To enable the internal IdP to accept JWT tokens from an external IdP:

1. Make a backup copy of the /opt/teradata/tdat/tdgss/site/TdgssUserConfigFile.xml and save it according to your site standard backup procedures.
2. Edit TdgssUserConfigFile.xml and configure the internal IdP and external IdP:

```
<!-- JWT -->
<!-- To modify JWT mechanism configuration, uncomment this section and edit -->
<Mechanism Name="JWT">
  <MechanismProperties
    MechanismEnabled="yes"
    DefaultMechanism="no"

    JWTDynamicKey="yes"
    JWTTOKENExchange="yes"
    JWTClientTlsCACertDir="/etc/ssl/certs"
  />

  <TokenExchanger
    Ref="ping-internal"
    ClientId="account"
    ClientSecret="Y2I2OGZkZTctM2FjMC00OWQwLWIZMGUtODJjMGIxNTY2NzAy"
    ClientSecretProtected="yes"
  />

  <IdentityProvider
    Id="Keycloak21"
    Url="https://keycloak21/auth/realms/uda"
    Type="keycloak"
    ValidateByTokenExchange="yes"
  />

  <IdentityProvider
    Id="ping-internal"
    Url="https://auth.pingone.asia/0cea60dc-0279-4b55-98a1-
eca07904733a/as"
    Type="Ping-Federate"
    ValidateByTokenExchange="no"
  />

  <UserNameMapping
    Claim="given_name"
    Match="(\w+)"
    DatabaseName="{1}" />

  <UserNameMapping
    Claim="sub"
    Match="(\w+).*.com"
    DatabaseName="{1}" />

  <UserNameMapping
    Claim="preferred_username"
    Match="(\w+)@(\w+).com"
    DatabaseName="{1}" />

</Mechanism>
```

Where the *<TokenExchanger>* section properties are:

Property	Description
<i>Ref</i>	Refers to the internal identity provider configured. Ref is used to get the URL for the external Issuer of the JWT.

Property	Description
<i>ClientId</i>	The ID of the Gateway that is used during the token exchange.
<i>ClientSecret</i>	The Gateway Client secret password. If <i>ClientSecretProtected</i> is set to yes then <i>ClientSecret</i> is encrypted and saved.
<i>ClientSecretProtected</i>	Determines if the Gateway Client Secret is password protected. By default, this value is yes.

Where the *<IdentityProvider>* section properties are:

Property	Description
<i>ValidateByTokenExchange</i>	Flag to indicate if a JWT coming from this Identity Provider needs to do a token exchange for validation. Valid values are Yes or No. Set to No by default.
[Optional] <i>SubjectIssuer</i>	The alias name by which the external IdP (the access token issuer) is registered with the internal IdP. This parameter is optional if Ping-Federate is the internal IdP. <i>SubjectIssuer</i> is needed if Keycloak is the internal IdP.
<i>Id</i>	Uniquely identifies the IdP in the configuration file.
<i>Url</i>	URL of the IdP. From the URL, TDGSS can issue REST API calls to get the required URLs and other information, such as Issuer, JWK URI, and so on. This URL is not related to <i>IdpUrl</i> in the <i>&lt;GlobalValues&gt;</i> section. <b>Note:</b> It is required to use secure https URLs.
<i>Type</i>	The type of the Identity Provider. Example values are: Ping-Federate, keycloak, vantage-keycloak, azuread, okta, and auth0.

Different identity providers may supply usernames in different formats. *<UserNameMapping>* provides the expected patterns for usernames, so JWT can extract usernames from the payload.

The *<UserNameMapping>* section properties are:

Property	Description
<i>Claim</i>	Claims are a portion of the payload that contain information about the user, among other things.
<i>Match</i>	The pattern the claim is matched to. For example, if the claim contains a sub (subject), that sub is parsed by the pattern so that a valid database username can be extracted from it.

Property	Description
<i>DatabaseName</i>	The portion of the claim that is extracted and is used as the database logon username for the user.

For more information about user name mapping, see [User Name Mappings](#).

- Place the CA certificates in the location specified in *JWTClientTlsCACertDir*. This directory is typically here: `/opt/teradata/tdat/tdgss/site/ssl/cacerts`.
- Verify the configuration is correct:
  - Run `tdgsstestcfg` to verify the new configuration is correct. It launches a test environment in a new shell that contains the updates to the configuration file.

```
/opt/teradata/tdgss/bin/tdgsstestcfg
```

- Test the configuration with the `tdgssauth` tool:

```
tdgssauth -m JWT -a token=JWT_from_IdP
```

Where *JWT\_from\_IdP* is the IdP you configured in *TdgssUserConfigFile.xml*. You can run the command multiple times to test the configuration for each IdP configured.

- Exit the test shell:

```
exit
```

- Continue editing and testing until the configuration is correct.

- Run:

```
/opt/teradata/tdgss/bin/run_tdgssconfig
```

- If `run_tdgssconfig` indicates that a TPA reset is required, run:

```
tpareset -f "use updated TDGSSCONFIG GDO"
```

## Configuration of Static Keys

### Configuration of Static JSON Web Key

To manually configure keys:

- Make a backup copy of `/opt/teradata/tdat/tdgss/site/TdgssUserConfigFile.xml` and save it according to your site standard backup procedures.
- Edit *TdgssUserConfigFile.xml* and uncomment *IdentityProvider*. Edit it so it is similar to the following with your site information:

```

<Mechanism Name="JWT">
  <MechanismProperties
    MechanismEnabled="yes"
    DefaultMechanism="no"
    JWTDecryptionKeyFile=""
    JWTVerificationKeyFile=""
    JWTSkewTime="300"
    JWTKeyDirectory="/opt/teradata/tdat/tdgss/site/JWTKeyDir"

    .../>

    <IdentityProvider
      Id="Keycloak2"
      Url="https://Customer_IdP_URL"
      Type="keycloak"
    />
  </Mechanism>

```

3. Use the `getjwk` tool to add the JWK file to the directory specified in *JWTKeyDirectory*.

`getjwk` gets the JWK file from the identity provider and saves it in the specified directory. For example, run:

```

/opt/teradata/tdgss/bin/getgwk -d /opt/teradata/tdat/tdgss/site/JWTKeyDir -i
Keycloak2 -u http://sdw01827.labs.teradata.com:8080/auth/realms/TGTE

```

`getjwk` options are as follows:

Option	Description
-h or --help	Displays help information for the command.
-d or --dir	Required. Absolute Path of the directory to store the JWK file. The path should match the path in the configuration file shown in <i>JWTKeyDirectory</i> .
-i or --idp-id	Id of the Identity Provider configured the IdentityProvider section.
-u or --idp-url	IdentityProvider URL to do the service discovery.
-m or --max-timeout	Maximum time in seconds that the operation is allowed to take. Default value is 900.
-v or --verbose	Displays detailed output.
INFO	CA Certs path is fetched from the location specified in the <i>JWTClientTlsCACertDir</i> property.

Option	Description
	If <i>JWTClientTlsCACertDir</i> is not defined, the default location <i>/etc/ssl/certs</i> is used.

Result: After a successful execution there are two files in the directory specified in the *-d* option.

4. Verify the configuration is correct:

- a. Run `tdgsstestcfg` to verify the new configuration is correct. It launches a test environment in a new shell that contains the updates to the configuration file.

```
/opt/teradata/tdgss/bin/tdgsstestcfg
```

- b. Test the configuration with the `tdgssauth` tool:

```
tdgssauth -m JWT -a token=JWT_from_IdP
```

Where *JWT\_from\_IdP* is the IdP you configured in *TdgssUserConfigFile.xml*.

- c. Exit the test shell:

```
exit
```

- d. Continue editing and testing until the configuration is correct.

5. Run:

```
/opt/teradata/tdgss/bin/run_tdgssconfig
```

6. If `run_tdgssconfig` indicates that a TPA reset is required, run `tpareset` to activate the changes to the TDGSS configuration:

```
tpareset -f "use updated TDGSSCONFIG GDO"
```

7. Remove the old key from the directory.

## Configuration of Static Decryption and Verification Keys (Legacy)

The JSON Web Token (JWT) authentication mechanism enables single sign-on (SSO) to Teradata Vantage after the user successfully authenticates to Teradata UDA User Service. The UDA User Service authenticates users to various UDA applications and services, such as AppCenter and the Teradata® Query Service (REST services). JWT allows a user that has been authenticated to one of the applications or services to do a single sign-on to establish a session with Teradata Vantage.

Complete the following setup to enable the use of JWT authentication:

1. Get the decryption and verification keys from the UDA User Service by calling Teradata® Query Service (REST APIs). This can either be done through the service's built in Swagger UI browser interface or by using cURL commands. By default, the Swagger UI endpoints are configured to be blocked, so the recommended method is to use cURL.

The following commands can be used to authenticate and retrieve the keys. Do the following from a database node that has access to the UDA User Service:

- a. Authenticate as an Admin user and get a JWT:

```
curl -X POST --header 'Content-Type: application/json' --header 'Accept: application/json' -d '{ \
  "password": <PW>, \
  "username": <USERNAME> \
}' 'https://<server_name>:<port_num>/token'
```

- b. Get the decryption key:

```
curl -X GET --header 'Accept: text/plain' --header 'Authorization: Bearer <MY JWT TOKEN>' 'https://<server_name>:<port_num>/decryptionKey'
```

- c. Get the signature (verification) key:

```
curl -X GET --header 'Accept: text/plain' --header 'Authorization: Bearer <My JWT TOKEN>' 'https://<server_name>:<port_num>/signatureKey'
```

Where <server\_name> is the server running the UDA User Service and <port\_num> is the port number of the UDA User Service.

---

**Note:**

The port number (<port\_num>) is configurable. For the RPM version of the user service it is usually 8001. Replace <port\_num> in the example commands with the port number for your configuration.

---

- d. Save the key files to any file name and in any location. The decryption and verification key files should have a .pem extension and should contain a header and a footer.

For example, the decryption key is similar to this:

```
# cat decryption_key.pem
-----BEGIN RSA PRIVATE KEY-----
MIICXgIBAAKBgQCdx3v70097sQ4retwU54YFp+khWhqZ4qZ0ekUTz/9T9a36bbX6
2TnRTR/59tkITiu5mPwQCoAvL9DZNJWUUpzjugWjZGzBHLfKe3KTnhw6IXNnHb
PJV7b5vunaoDt+iur+MkKcgj0i+4G/mmXUk/rIFiLJQtWVr4Coj3jYj+NQIDAQAB
AoGBAIBnMyCJNgys2AJMl0Uv8mMx9kldQd7QlHSgeQ0ZrgpPG4p9tKb0F9ic8pQD
7zaSH4WI2kHXueAtAsNxxvWRkf17pzVjUMrIqwnivkNFTY4iPzJeRw/3KLxhlgfv
```

```

Q7l+CMprKnLusc19Dt9oR4+Ypm745yPJ+6ZnHJyvXELPU0rVAKEA6fhqcumIEOsJ
TU+Lo94xKngXHS4ms9dND0xZEhPImTXz4YPWCCwX11d6wII2Tz3k+LTQ/I/2rQy8
9DXpwBs4hwJBAKyikc9jvEEHfeUNNymjff5Bg9eRVnzPaq6QitTaXvT/zPgjE10Fh
GpADP7fUiUR4PtocZ47Q8co+jIMR8XEvTmMCQQC3SjmyLgq/HjGaVB+Tz0P6/js1
S+tb5eXjfy8j/0Wd60tWlt48ZraCp3BtkVSUfWt7/sLdqLZans4kDnxBV9HPAkEA
qNzoNW1AUneqjKdNovwbpjVBsJSUALvN8uJEUV9BrdEXh+oKGx8ppV6YMA/EKWZZ
TG3mWgtFx2dBeF/PxL/aMwJAWepYUPkM3MReuAp7oij4qIrOq9xPpH104+kBKMwp
yR6wvSLqDxSX3erkBq6Eh39BMchta3clU7PXck4pKrDf0A==
-----END RSA PRIVATE KEY-----

```

For example, the verification key is similar to this:

```

# cat verification_key.pem
-----BEGIN PUBLIC KEY-----
MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQCdx3v70097sQ4retwU54YFp+kh
WhqZ4qZ0ekUTz/9T9a36bbX62TnRTR/59tkITiu5mPwQCoAvL9DZJWUUpzjugWj
ZGzBH1CFfKe3KTnhw6IXNnHbPJV7b5vunaoDt+iur+MkKcgj0i+4G/mmXUk/rIFi
LJQtWVr4Coj3jYj+NQIDAQAB
-----END PUBLIC KEY-----

```

2. (For multi-node installations) Securely transfer the keys to the other database nodes. Check the permissions of the key files to make sure Teradata Vantage can access them. To transfer the keys, do the following:
  - a. Log on to the database node that contains the keys.
  - b. Move the keys to the other database nodes:

```

pcl -send <location>/<decryption_key_file_name>
<location>/<decryption_key_file_name>

pcl -send <location>/<verification_key_file_name>
<location>/<verification_key_file_name>

```

**Note:**

Store the decryption and verification key files in the same location on all the nodes.

3. Make a backup copy of /opt/teradata/tdat/tdgss/site/TdgssUserConfigFile.xml and save it according to your site standard backup procedures.
4. Edit the TdgssUserConfigFile.xml and uncomment the following section:

```

<!-- JWT -->
<!-- To modify JWT mechanism configuration, uncomment this section and edit
      <Mechanism Name="JWT">
        <MechanismProperties

```



```

        MechanismEnabled="yes"
        DefaultMechanism="no"

        JWTDecryptionKeyFile=""
        JWTVerificationKeyFile=""
        JWTSkewTime="300"
    />
</Mechanism>
(end of commented out section)-->

```

5. Optional. Set JWTDecryptionKeyFile to the absolute path to the file containing the decryption key.
6. Set JWTVerificationKeyFile to the absolute path to the file containing the verification key.
7. Optional. Edit and set JWTSkewTime. JWTSkewTime is the number of seconds a JWT will be still valid after its expiration.
8. Save the file.
9. Run the run\_tdgssconfig utility to update the TDGSSCONFIG GDO:

```
/opt/teradata/tdgss/bin/run_tdgssconfig
```

10. Run tdgssfixpaths to set the owner and permissions on the JWTDecryptionKeyFile and JWTVerificationKeyFile:

```
psh 'perl /opt/teradata/tdgss/bin/tdgssfixpaths'
```

11. You can edit mechanism properties that begin with JWT without performing a TPA reset. Other modifications may require a reset. run\_tdgssconfig indicates when you need to do a TPA reset. If indicated, run:

```
tpareset -f "use updated TDGSSCONFIG GDO"
```

## User Name Mappings

The UserNameMapping section of TdgssUserConfigFile.xml is used to parse out the database username. To obtain the database logon username, JWT uses the subject (sub) claim from the JWT payload as the username by default. But, the claim may be different for each identity provider or the claim may need to be parsed to select a portion of it to be used as the username.

For example, if a JWT claim has preferred\_username and the value is an email address like xys@company.com, the email address must be mapped to xys because an email address cannot be used as a database username. In the example, the UserNameMapping section contains a claim for preferred\_username: claim=preferred\_username. The claim is matched with the regular expression pattern in *Match*. If it matches, DatabaseName = \$1. In this example, \$1 is xys according to the *Match* pattern. So, xys is used as the database username.

```

<Mechanism Name="JWT">
  <MechanismProperties

    JWTDynamicKey="yes|no"
    JWTTokenExchange="yes|no"
    JWTClientTlsCACertDir="/opt/teradata/tdat/site/xyz/abc>"

  .../>

  <UserNameMapping
    Claim="preferred_username">
    Match="(\w+)@([\w+.]+)"
    DatabaseName="${1}" />

  <UserNameMapping
    Claim="sub">
    Match="(\w+)|(\w+)"
    DatabaseName="Auth_${2}" />

  <UserNameMapping
    Claim="sub"
    Match="(.+)"
    DatabaseName="${1}" />

  <UserNameMapping
    Claim="qlid"
    Match="(\w{2})(\d{6})"
    DatabaseName="${1}_${2}" />

</Mechanism>

```

Where the Match pattern is a POSIX regular expression.

## SSO Security Hardening

A JWT received from a client is validated using the JWK (JSON Web Key) from the JWK URI using REST API calls. For performance reasons JWK is cached, so that future validations are fast and avoid any further REST API calls. Some mechanism properties are added to JWT mechanism for security hardening.

### **JWTRestAPIMaxTimeAllowed**

The JWTRestAPIMaxTimeAllowed property specifies the maximum (in seconds) REST API call timeout.

The default setting is 20 seconds.

**JWTRestAPITimeLimit**

The JWTRestAPITimeLimit property specifies time (in seconds) between REST API calls. Too many REST API calls causes denial of service.

The default setting is 10 seconds.

**JWTKeyCacheRefreshTime**

The JWTKeyCacheRefreshTime property specifies the interval (in minutes) at which the key cache is purged, so the new key cache is refreshed.

The default setting is 1440 minutes (24 hours).

**JWTClientTlsCACertDir**

The JWTClientTlsCACertDir property specifies the location of the CA certificates. It specifies the full path to the site/ssl/cacerts directory.

There is no default, but it is typically here: /opt/teradata/tdat/tdgss/site/ssl/cacerts/.

**JWTClientUseTls**

The JWTClientUseTls property enforces TLS 1.2 or higher for REST API calls. This makes sure that the REST API always uses https and that peer and host verification is done.

The default setting is "Yes". The value "No" should not be used in production.

**JWTSkewTime**

The JWTSkewTime property specifies the maximum skew time (in seconds) allowed during JWT validation.

The default setting is 300 seconds (5 minutes).

**Related Information**

For more information about JWT, see <https://tools.ietf.org/html/rfc7519>.

For more information about configuring JWT, see [JWT Mechanism](#) and [JWT Support Properties](#).

For information about connectivity from clients, see:

- *Teradata JDBC Driver Reference*, available at <https://teradata-docs.s3.amazonaws.com/doc/connectivity/jdbc/reference/current/frameset.html>
- *.NET Data Provider for Teradata*, available at <https://teradata-docs.s3.amazonaws.com/doc/connectivity/tdnetdp/16.20/help/webframe.html>
- *ODBC Driver for Teradata® User Guide*, B035-2526
- *Teradata® Call-Level Interface Version 2 Reference for Mainframe-Attached Systems*, B035-2417
- *Teradata® Call-Level Interface Version 2 Reference for Workstation-Attached Systems*, B035-2418

# Creating Users and Granting Privileges

You must create users and define their database resources and privileges before you allow them to access the database. The process you must use depends upon how the users are authenticated and authorized. For information, see [Implementing User Authentication and Authorization](#)

## User Implementation Process

1. Satisfy the [Prerequisites](#).
2. Assign users to functional groups to help assess user setup needs. See [About Database User Functional Categories](#).
3. Use profiles to set user parameters, such as space allocations, password control settings, and security constraint assignments. See [Working with Database Profiles](#).
4. Review the database user types shown in [About Database User Types](#) and determine which of them apply to your authentication and authorization strategy. The method for creating users and defining their database privileges varies by user type.
5. Create the required users:
  - [Creating Permanent Database Users](#).
  - [Working with Directory Users](#).
  - [Working with Middle-Tier Application Users](#).
6. Review [About Teradata Vantage User Privileges](#) to understand which privileges you must grant and which are acquired indirectly.
7. Assign user privileges:
  - Directly to permanent database users. See [Granting Privileges to Permanent Database Users](#).
  - To roles, for permanent database users. See the topics beginning with [Using Roles to Manage Privileges](#).
  - To roles, for proxy users. See [Working with Roles for Proxy Users](#).
  - To external roles, for directory users. See [Using Roles for Directory Users](#).

---

**Note:**

When creating users and assigning user privileges, any SQL statements routed through Unity apply to all connected Vantage systems.

---

8. Review additional methods for restricting access based on the structure of the database and its component parts. See [Other Options for Restricting Database Access](#).

## Prerequisites

- Create administrative users and allocate user DBC space. See [Setting Up the Administrative Infrastructure](#).
- Define and implement a user management strategy, that specifies how the users are authenticated and authorized privileges in the database. See [Implementing User Authentication and Authorization](#).
- Create the databases, tables, views and other objects that make up the database, and which are the basis for user privileges. For more information see *Teradata Vantage™ - Database Administration*, B035-1093.

## About Database User Functional Categories

The database administrator and security administrator should work together to assess database user needs, based on typical job functions, before creating users and granting user privileges.

Functional Category	Description
General users	Database end-users who only need to read the data or to run pre-existing queries or macros to generate reports.
Update users	Privileged users who perform some general user functions, and who also need to insert, update, or delete data, and create new database objects.
Batch users	High-level users who typically perform batch-level functions such as: <ul style="list-style-type: none"> <li>• Load, update, and export operations, including creating and deleting staging tables.</li> <li>• Data backup, archive, and restore (BAR) operations.</li> </ul>
Database programmers	Users who design and create databases, tables, and views, as well as queries, macros and stored procedures for use by the user community. Programmers may require administrator privileges within a development database, while needing only limited privileges in the main production database.
Assistant administrators	Users who assist the principal administrator, user DBADMIN, created in <a href="#">Setting Up the Administrative Infrastructure</a> . Assistant administrators may have most or all of the same privileges granted to user DBADMIN, but typically have a much smaller permanent space allocation because they have a limited ownership of objects.

For information about the categories of users that exist in secure zones, see [Implementing Teradata Secure Zones](#).

## Assessing User Needs

To prepare for creating and provisioning database users:

1. Make a list of all users that require access to the database, and identify each one according to functional category. Minimize the number of user types to simplify user management.
2. Define user resource requirements for use in creating profiles:

- Examine user space requirements:
  - Users who create or own databases, tables, and other space-consuming objects require permanent storage space (perm space).
  - Users who submit SQL queries, macros, stored procedures or other executable requests require spool space to contain the temporary database structures used to execute the requests.
- Define user accounting requirements for resource accounting and prioritizing each user request. Then create the accounts, as shown in *Teradata Vantage™ - Database Administration*, B035-1093, and assign the accounts to users, either directly or through use of profiles. Each account can specify:
  - A priority level (low, medium, high, and rush)
  - An account identifier that specifies such things as department, group, and function
  - A date and time stamp
- Define the user default database (the database where the user most often works) to avoid specifying the database as part of each request.
- Define password control parameters. Consider your site security policy and decide whether or not all users can share the global default password parameters referenced in [Setting Up the Administrative Infrastructure](#), or if you need to set these parameters separately for groups of users.
- Determine whether users are subject to row level security constraints that should be assigned in profiles. See [Working with Constraint Assignments](#).

---

**Note:**

Users that log on through applications that pool sessions do not have access to personal profiles, and instead defer to the profile for the application user or trusted user.

---

3. Review the database objects (such as views, tables, macros, functions, and procedures) that users or user groups must access to do their job. Identify groups of users with common database privilege requirements and create roles to define the privileges for each group, rather than granting privileges to individual users.

## Working with Database Profiles

You can create profiles to define resource parameters for groups of users with similar needs, rather than defining them for individual users. Then specify the profile name in a CREATE or MODIFY USER statement to give users membership in the profile.

Profiles specify such parameters as:

- Spool space and temporary space allocations
- Password control settings
- Default database assignment
- User account assignments

- Row level security constraint assignments

## Privileges Required for Creating Profiles

Users that create profiles must have the CREATE PROFILE or PROFILE privilege. The creator of a profile automatically receives the DROP PROFILE privilege on the profile.

## Creating Profiles

Create database profiles using the CREATE PROFILE statement, documented in *Teradata Vantage™ - SQL Data Definition Language Syntax and Examples*, B035-1144.

## Default Values for the CREATE PROFILE Statement

If you do not specify a value for an optional parameter in the CREATE PROFILE statement, the profile uses the default value.

Many profile parameters also appear in the CREATE USER statement. Settings in a profile always take precedence over settings defined in a CREATE or MODIFY USER statement for the users that are members of the profile. However, if no value appears in the profile, it reverts to the corresponding value in the CREATE USER statement, if specified.

The following table describes the default parameter values the system uses if the profile does not contain a value.

Parameter	Result of Not Specifying a Profile Value
ACCOUNT	The system uses the default account specified for the user in a CREATE or MODIFY USER statement. If an account is not specified for the profile or the user, the system uses the account of the owner of the user.
DEFAULT DATABASE	The system uses the default database specified for the user in a CREATE or MODIFY USER statement. If a default database is not specified for the profile or the user, the system uses the username as the default. The user can also specify a default using SET SESSION DATABASE.
SPOOL	The system uses the spool limit defined for the user in a CREATE or MODIFY USER statement. If a spool space limit is not specified for the profile or the user, the system applies the SPOOL value for the immediate owner of the space in which the user is being created.
TEMPORARY	The system uses the temp space setting defined for the user in a CREATE or MODIFY USER statement. If a temp space allocation is not specified for the profile or the user, the user cannot create objects that require temporary space.

Parameter	Result of Not Specifying a Profile Value
PASSWORD	<p>The default password control settings are taken from the global defaults for the corresponding attributes in the DBC.SysSecDefaults table. See <a href="#">Viewing Current Password Control Settings</a>.</p> <p><b>Note:</b> PASSWORD controls apply only to users authenticated by Teradata Vantage. Externally authenticated users are not affected.</p>

## Precedence of Values for Profile Parameters

The values in a profile always take precedence over similar values defined for a user in a CREATE USER or MODIFY USER statement.

Profile members inherit any changed parameters. Parameters take effect at various times:

- SPOOL and TEMPORARY space allocations are imposed immediately. This affects the current session of any member who is logged on at the time the profile is modified.

---

**Note:**

If you create a profile with a 10 TB spool space limit and assign 10 users to the profile, each user can run queries in parallel and consume a total of 10 TB of spool (assuming you have 10 TB of space free).

---

- Changes to ACCOUNT and DEFAULT DATABASE do not take effect until the next login. However, if a member user submits a SET SESSION ACCOUNT statement at a point in the session after the profile is modified, the database uses the new values.
- Password control changes take effect at the next login by user members of the profile.

## About Assigning Profiles to Users

The user categories shown in the table below require differing strategies for assigning profiles.

---

**Note:**

For a description of each database user type, see the table in [About Database User Types](#).

---

Database User Type	Profile Assignment Method
Permanent database users	Specify the profile name in the CREATE USER or MODIFY USER statement for the user.
Directory users	Assign profiles to directory users in one of the following ways:



Database User Type	Profile Assignment Method
	<ul style="list-style-type: none"> <li>Map the directory user to one or more profile objects. For directory users mapped to more than one profile, the user must set the profile using <code>profile=profile_name</code> in the .logdata portion of the logon string. For information on mapping directory users to database profiles, see <a href="#">Mapping Directory Users to Vantage Profiles</a>.</li> <li>Map the directory user object to a Vantage user to provide the following profile assignment options: <ul style="list-style-type: none"> <li>The directory user inherits the profile assigned to the database user.</li> <li>If the database user does not have an assigned profile, the directory user inherits the default parameter values for the database user. See <a href="#">Default Values for the CREATE PROFILE Statement</a>.</li> <li>For directory users mapped to a Vantage profile and a database user, the mapped profile takes precedent by default.</li> </ul> For information mapping directory users to database users, see <a href="#">Mapping Directory Users to Database Users</a>.</li> </ul>
Application logon users or trusted users	Specify the profile name in the CREATE or MODIFY USER statement for the user name under which the application logs on to the database. The application user profile applies to all users that log on through the application.
Proxy users	<p>Proxy users who are also permanent database users, and for whom queries are sent to the database through a trusted user application, are subject to any row level security constraints that appear in the profile assigned to the corresponding permanent user. All other profile-based privileges are taken from the profile assigned to the trusted user application.</p> <p>For information on options for end users logging on through middle-tier applications, see <a href="#">Working with Middle-Tier Application Users</a>.</p> <p>For information on row level security constraints, see <a href="#">Implementing Row Level Security</a>.</p>

For information about the types of users that exist when you use secure zones, see [Implementing Teradata Secure Zones](#).

## Using a Profile to Set a Default Query Band

A default query band can be defined in a profile using the CREATE PROFILE statement. The profile query band can be used for tracking and controlling system usage, similar to the SET QUERY\_BAND statement. The profile query band is automatically set for the session at logon time.

To assign the profile to a user, specify the profile name in the CREATE USER or MODIFY USER statement for the user.

The SET QUERY\_BAND statement sets, updates, and removes the session and transaction query bands for the session. This statement does not affect the profile query band. The only way the profile query band changes for an existing session is if the profile for the session changes.

Use the MODIFY PROFILE statement to modify the profile query band. All profile query band changes to active sessions take effect at the beginning of the next request.

The profile query band is not saved in DBC.SessionTbl, so after a restart the session is initialized with the profile query band based on the current profile setting.

For information about the CREATE PROFILE and MODIFY PROFILE statements, see *Teradata Vantage™ - SQL Data Definition Language Syntax and Examples*, B035-1144.

## Dropping Profiles

Use the DROP PROFILE statement to delete a profile. For sessions that log on after the profile is dropped:

- Spool and temporary space settings revert to the default settings.
- Account and database settings default to the settings defined for the affected user.
- Any password attributes defined in the profile default to the global password settings.
- The system does not enter a NULL for *profile\_name* in the user definition. Therefore, if you recreate a profile with the same name as a dropped profile, the user is subject to the profile parameters at the next logon.

## About Database User Types

A Teradata Vantage user:

- Can log on to Teradata Vantage, establish logon sessions, and perform actions.
- Is defined as an object, much like a database, and if the user object is defined to have perm space, it can contain other database objects. A user owns the objects contained within its perm space.

The method used to define a user depends on how the user is managed.

Vantage User Type	Description
Permanent user	Define permanent users in Vantage using a CREATE USER statement and manage them from within Vantage. See <a href="#">Creating Permanent Database Users</a> .
Directory-based user	Create users in the directory and map them to Vantage objects. See <a href="#">Working with Directory Users</a> .
Auto provisioned user	If auto provisioning is configured for your system, unmapped users can automatically obtain a Teradata Vantage user identity when they first log on to the system. To auto provision a Vantage account the user must have an identity in the directory and be mapped to a Vantage object, such as an external role or profile, but not yet be mapped to a Vantage user.  During auto provisioning a user object is automatically generated and granted LOGON ... WITH NULL PASSWORD privileges. Auto provisioned users must always authenticate externally.  Unlike the pseudo-user, EXTUSER, auto provisioned users have permanent, individual identities in Vantage. This allows them to create and own database objects, and use global temporary tables and volatile tables.

Vantage User Type	Description
	Because auto provisioned users have a Vantage identity, they can be individually subjected to access logging and workload management rules, and can use administrative tools, such as Teradata Viewpoint. See <a href="#">About Auto Provisioned Directory Users</a> .
Application logon user	An application logon user is a permanent username under which a middle-tier application server logs on to Vantage. Define application logon users similarly to other permanent users, using the CREATE USER statement. See <a href="#">Working with Middle-Tier Application Users</a> .
Application end user	Assume the identity and database privileges of the logon user for the application through which they log on.
Trusted user	A trusted user is a middle-tier application that is specially configured to allow end users (proxy users) to log on as individuals. Define trusted users by entering the permanent username under which the trusted user application logs on to Vantage in a GRANT CONNECT THROUGH statement. See <a href="#">Working with Middle-Tier Application Users</a> .
Proxy user	A proxy user is an end user that logs on to Vantage through a trusted user application. The system identifies and authorizes the user as an individual. Proxy users can be either permanent users or other end users unknown to Vantage. Define proxy users in Vantage using a GRANT CONNECT THROUGH statement, which also identifies the trusted user application through which the user can log on. For information on creating proxy users, see <a href="#">Working with Middle-Tier Application Users</a> .

## Creating Permanent Database Users

### Privileges Required for Creating Database Users

You must explicitly grant the CREATE USER or USER privilege to any user that requires it, except user DBC, which has these privileges by default. The creator of a user automatically receives the DROP USER privilege on the user.

### Creating a Database User

Create database users using the CREATE USER statement, documented in *Teradata Vantage™ - SQL Data Definition Language Syntax and Examples*, B035-1144.

---

#### Note:

For information on assigning row level security constraints to users, see [Working with Constraint Assignments](#).

---

## CREATE USER Default Values

The following table summarizes default values associated with the CREATE USER statement. If you do not define these elements in the CREATE USER statement, the system supplies the default values described in the table.

**Note:**

PERM space and PASSWORD clauses have no default values. You must specify values for both at user creation.

Syntax Element	Default Value
FROM <i>database</i>	The default database of the creating user.
TEMPORARY	If a profile is: <ul style="list-style-type: none"> <li>Assigned to the user, and includes a TEMPORARY value, the default is the value defined in the profile.</li> <li>Assigned but does not include a TEMPORARY value, the default is the same TEMPORARY value as the immediate owner of the space in which the user is created.</li> <li>Not assigned, the default is the same TEMPORARY value as the immediate owner of the space in which the user is created.</li> </ul>
SPOOL	If the profile is: <ul style="list-style-type: none"> <li>Assigned to the user, and includes a SPOOL value, the default is the value defined in the profile.</li> <li>Assigned but does not include a SPOOL value, the default is the SPOOL value for the immediate owner of the space in which the user is created.</li> <li>Not assigned, the default is the SPOOL value for the immediate owner of the space in which the user is created.</li> </ul>
DEFAULT DATABASE	The user name. The user can submit a SET SESSION DATABASE statement to change the default for the current session, as long as the appropriate privileges on that database are granted to the user.
ACCOUNT	If a profile is: <ul style="list-style-type: none"> <li>Assigned to the user, but the profile contains more than one account, the default is the first account in the string. The user can submit a SET SESSION ACCOUNT statement to set the default for the current session to one of the other accounts.</li> <li>Assigned to a user and the profile contains only one account, the default is the account defined in the profile.</li> <li>Assigned to a user, but the profile does not include an account, the default is the account identifier of the immediate owner of the user.</li> <li>Not assigned to the user, the default is the account identifier of the immediate owner of the user.</li> </ul>

Syntax Element	Default Value
	<b>Note:</b> If a user definition specifies an account but not a profile, the default account is the account string in the user definition. If you assign a profile to a user, and the profile specifies an account string, the profile account is in effect for AMPUsage and DBQL for that user. However, the system uses the value found in the DBC.DatabaseV view for space accounting.
DEFAULT ROLE	Null. The database does not use a role for privilege validation unless the user submits a SET ROLE statement for roles that you explicitly grant to the user, including nested roles. See <a href="#">Assigning the Default Role</a> .
STARTUP	Null (no startup string).
PROFILE	Null. No profile exists for the user.

## Dropping the Default Database for a User

When you drop the default database for a user, the system defaults to the username space as the session database, and does not return an error or warning. The user can use the SET SESSION DATABASE statement to set the default database during the session.

## Dropping Permanent Database Users

The effects of dropping a permanent database user can be very complex. Review the DROP USER statement in *Teradata Vantage™ - SQL Data Definition Language Detailed Topics*, B035-1184 before you drop a user.

## Working with Directory Users

Administrators have the following options to manage database users in a supported directory.

- Directory authentication with Teradata Vantage authorization.
- Directory authentication and authorization.

Both of these options require that you complete some setup tasks in the database, in addition to the directory administration tasks.

For information on...	See...
options for managing database users in a directory	<a href="#">Directory Management of Database Users</a> .
using database external roles to assign privileges to directory users	<a href="#">Using Roles for Directory Users</a> .

## About Auto Provisioned Directory Users

Auto provisioning allows directory principals to get a Teradata Vantage logon automatically, without a DBA creating a database account. To use auto provisioning, the database system must be enabled to allow auto provisioning and the directory principal must be a member of a Vantage external role or profile. The directory principal must not be mapped to a database user object.

At the initial logon, a database user identity is created for the auto provisioned user. The database user account is given a NULL password. Attributes, such as SPOOL space, are allocated according to the profile to which the directory principal is mapped or set to zero if the directory principal is not a member of a profile.

The privileges given to the auto provisioned account are determined by the external role to which the directory user is assigned. If an auto provisioned directory user is assigned to an external role and is also granted a role in the database, the user is allowed to have the privileges of both roles; however, the user is externally authenticated, so only external roles are active for the session. A granted role must be explicitly enabled. If the directory principal is not assigned to a role, the user inherits privileges from EXTERNAL\_AP (a system user).

In subsequent logons, the user must authenticate to an authenticating mechanism, such as the directory or Kerberos. Auto provisioned users are given authorization by the directory.

---

### Note:

By default, auto provisioning is disabled. When it is disabled, external directory users who are not mapped to permanent database users are logged on as EXTUSER.

---

There are several advantages in using auto provisioning instead of EXTUSER.

- Auto provisioning removes privileges limitations that EXTUSER is subject to; for example, EXTUSER has no USER right, no WITH GRANT OPTION, and no per-DSA-user grant/revoke.
- Auto provisioning allows assignment of SPOOL and TEMP space on a per user basis.
- Auto provisioned users can be identified by tools such as Viewpoint and TASM.
- Auto provisioned users can be individually logged.

### Prerequisites for Auto Provisioning

A supported directory server must be running and configured for authorization.

The LDAP, Kerberos (KRB5), or SPNEGO authentication mechanisms in TDGSS must be configured to authorize users. This means TDGSS must be configured on Vantage nodes and Unity servers with MechanismEnabled = "yes" and AuthorizationSupported = "yes".

External authentication must be enabled in the database and on the gateway.

The AutoProvision DBSControl parameter must be enabled. Run dbscontrol and enter `m g 81 T`.

Profiles and external roles must exist in the database. Matching profile and role objects must exist in the directory.

The directory principals to whom you want to provide auto provisioning must be assigned to roles or profiles in the directory.

---

**Note:**

Directory principals must not be mapped to a database user object.

---

## Setting Up Auto Provisioning

If your directory, users, roles, profiles, and external authentication need to be set up, follow the steps in [Directory Database User Implementation Process](#) to configure everything, including auto provisioning.

If your directory, directory users, directory and database roles and profiles, and external authentication are already set up, perform the following steps to enable and use auto provisioning.

1. Enable the AutoProvision parameter in DBSControl.

```
dbcontrol m g 81 T
```

2. To provide auto provisioning to selected directory principals assign them to database objects (roles or profiles) in the directory. On their first logon attempt a database account is created for these users.
  - See the examples for mapping users to external roles in [Mapping Directory Users to Vantage External Roles](#).
  - For information about mapping to profiles see [About Assigning Profiles to Users](#) and see the example in [Mapping Directory Users to Vantage Profiles](#).

## Working with Middle-Tier Application Users

Middle-tier applications are situated between Teradata Vantage clients and the Teradata Vantage Advanced SQL Engine. A middle-tier application server that accesses Vantage must log on as a permanent database user to establish a session pool before end users can access the database through the application.

---

**Note:**

Users who access the database through a middle-tier application are subject to the security policies assigned to the application logon user, rather than the policies assigned to them as individuals. For details, see [Network Security Policy](#).

---

## Setting Up a Permanent Application Logon User

You can set up each application as a single user, for example:

1. Create a permanent database user (logon user) for the application, for example, App1User.
2. Create a profile and one or more roles for the application logon user.

**Note:**

Use the PASSWORD option in the profile to set the PasswordExpire parameter to 0 for the application, to prevent the application password from expiring, which could cause a disruption in service for application end user

After the application logs on and initiates a session pool, all application end users share the logon user identity and privileges.

## Setting Up Trusted User Applications and Proxy Users (Recommended)

In addition to setting up a logon user for each application, you can optionally set up the application as a trusted user and then define proxy users, so that application end users log on to the database with individual identities and role privileges.

1. Create a permanent database user for the application, for example, App1User.
2. Create a profile for the application logon user.
  - a. Use the PASSWORD option in the profile to set the PasswordExpire parameter to 0 for the application, to prevent the application password from expiring, which could cause a disruption in service for application end users.
  - b. If you want row level security constraints to apply generally to all application end users, you can assign RLS constraints to the application user profile. See [Assigning Security Constraints in a CREATE PROFILE Statement](#).
3. Create roles for defining database privilege variations among application (proxy) users.
4. For proxy users that are also permanent database users, you can optionally modify the permanent user profile to assign row level security constraints, if used. See [Assigning Security Constraints in a CREATE PROFILE Statement](#).
5. Grant the CTCONTROL privilege to the user who sets up trusted user applications, for example:

```
GRANT CTCONTROL ON trusted_user_name TO user_name ;
```

**Note:**

The administrator that issues the GRANT CONNECT THROUGH privilege TO a permanent database user must have the DROP USER privilege on the permanent user.

6. Use the GRANT CONNECT THROUGH statement to define a trusted user and associate the proxy users and roles, for example:
  - For users already defined in the database (permanent users):



```
GRANT CONNECT THROUGH trusted_user_name
  TO PERMANENT perm_user_name [, perm_user_name]
  WITH ROLE role_name [, role_name] | WITHOUT ROLE;
```

- For application users not known to the database:

```
GRANT CONNECT THROUGH trusted_user_name
  TO app_user_name [, app_user_name]
  WITH ROLE role_name [, role_name];
```

- To restrict use of SET QUERY\_BAND statements through a trusted user application to those statements included in a trusted request:

```
GRANT CONNECT THROUGH trusted_user_name [WITH TRUST_ONLY] ;
```

Syntax Element	Description
<i>trusted_user_name</i>	The permanent username that the application uses to log on to Teradata Vantage and establish a session pool. This user must exist in the database before it is referenced in a GRANT CONNECT THROUGH statement.
<i>perm_user_name</i>	The name of a permanent database user being defined as a proxy user. You must precede the <i>perm_user_name</i> with a TO PERMANENT clause to identify the user as a permanent user. A user must exist in the database before you can reference it in a GRANT CONNECT THROUGH statement. You can specify up to 25 perm users in each GRANT CONNECT THROUGH statement. There is no limit to the number of perm users to which you can grant proxy logon privileges for a <i>trusted_user_name</i> .
<i>app_user_name</i>	The name of an application end user being defined as a proxy user. You must precede the <i>app_user_name</i> with a TO clause to identify the user as <i>not</i> being a permanent database user. Users associated with <i>app_user_names</i> do not exist in the database, but the names must follow Teradata Vantage object naming conventions. You can specify up to 25 app users in each GRANT CONNECT THROUGH statement. There is no limit to the number of app users to which you can grant proxy logon privileges for a <i>trusted_user_name</i> .
WITH ROLE <i>role_name</i>	Lists the role names available to the proxy user(s) contained in the GRANT CONNECT THROUGH statement. You must specify at least one role name in a GRANT CONNECT THROUGH statement that assigns proxy user status to application end users. The <i>role_name</i> must identify a role that exists in the database. ALL, NONE, and NULL are not valid role names. You can specify up to 15 role names in each GRANT CONNECT THROUGH statement, and you can specify up to 15 role names for each proxy user/trusted user name pair. If the CONNECT THROUGH privilege for a particular trusted user already exists for the <i>perm_user_name</i> or <i>app_user_name</i> , the database adds any new roles that you specify to the existing roles. If you exceed the limit of 15 roles for a user, the statement aborts.

Syntax Element	Description
	<p>Use the REVOKE CONNECT THROUGH statement to remove a role from the proxy user for a trusted user.</p> <p>For information on specifying roles in a proxy connect and the effects of the specifications on proxy user privileges, see <a href="#">Using the SET QUERY BAND Statement to Enable Session Proxy Roles</a>.</p>
WITHOUT ROLE	<p>If the <i>proxy_user_name</i> is a permanent database user, the proxy user inherits the access privileges, including roles, of the permanent user.</p> <p><b>Note:</b></p> <p>Do not use WITHOUT ROLE for GRANT CONNECT THROUGH statements that specify an <i>app_user_name</i>.</p>
WITH TRUST_ONLY	<p>Instructs the database to honor SET QUERY_BAND statements that set or update a proxy user only if they are part of a trusted request. This prevents end users from changing their database privileges.</p> <p>If you specify the WITH TRUST_ONLY option, you cannot create proxy users in the same request.</p> <p><b>Note:</b></p> <p>To use this feature, you must set up the query band to tag requests as trusted or not trusted.</p>

- You must set up the application to send a SET QUERY\_BAND statement to Teradata Vantage to initiate each proxy user session, identify the proxy user, and optionally specify the operant proxy user role for the session or transaction.

See [Using the SET QUERY BAND Statement to Enable Session Proxy Roles](#).

## About Teradata Vantage User Privileges

Database users can access only those database objects and functions allowed by the database privileges available to them. Users acquire some privileges automatically, but you must grant, give, or assign others. Privileges can include the right to manage privileges for other users.

Database privileges are also known as a rights, access rights, or permissions.

If a user does not have the required privileges for a database request, the database denies the action, returns an error message to the user, and logs a violation in the access log.

Some row level security restrictions do not generate errors or log violations. For information, see [About Access Logging for Row-Level Security](#).

## Database Privilege Types

All database privileges are either explicit or implicit.

Privilege	Description
<b>Implicit Privileges</b>	
Ownership	<p>Teradata Vantage™ grants implicit privileges on a database object to the owner of the space that contains the object.</p> <p>See <a href="#">Ownership Privileges</a>.</p>
<b>Explicit Privileges</b>	
Automatic	<p>When a user creates a database object, SQL Engine automatically grants privileges to:</p> <ul style="list-style-type: none"> <li>• The creator of the object</li> <li>• A newly created user or database</li> </ul> <p>See <a href="#">Automatic Privileges</a>.</p>
GRANT	<p>You can GRANT privileges:</p> <ul style="list-style-type: none"> <li>• Directly to a user or database</li> <li>• To a role, then GRANT membership in the role to one or more users</li> <li>• To an external role, then map the role to one or more groups of directory users</li> </ul> <p>See <a href="#">Working with User Privileges in Teradata Vantage</a>.</p>
Inherited	<p>Privileges that a user acquires indirectly:</p> <ul style="list-style-type: none"> <li>• All users automatically have the privileges of PUBLIC, a role-like collection of default privileges. You can also grant or revoke privileges for PUBLIC. See <a href="#">About System-Generated Users</a>.</li> <li>• A user inherits all the privileges granted to any roles of which the user is a member. See <a href="#">Using Roles to Manage Privileges</a>.</li> <li>• Directory users inherit the privileges of the database users and external roles to which they are mapped. See <a href="#">Using Roles for Directory Users</a>.</li> </ul>
Assigned	<p>Security constraints define user access to table rows protected by a corresponding security constraint column.</p> <p>You can assign the security constraints in a CONSTRAINT object to a:</p> <ul style="list-style-type: none"> <li>• User, by specifying the CONSTRAINT object in a: <ul style="list-style-type: none"> <li>◦ CREATE USER or MODIFY USER statement</li> <li>◦ CREATE PROFILE or MODIFY PROFILE statement, and then assigning the profile to the user</li> </ul> </li> </ul> <p>See <a href="#">About Assigning Security Constraints</a>.</p> <p><b>Note:</b></p> <p>Constraint OVERRIDE privileges, which allow a user to bypass row level security protection, are granted using the GRANT OVERRIDE CONSTRAINT statement.</p> <p>See <a href="#">Granting SQL DML OVERRIDE Privileges</a>.</p> <ul style="list-style-type: none"> <li>• Table, by defining a constraint column that is named for the CONSTRAINT object in a CREATE TABLE or ALTER TABLE statement. See <a href="#">Working with Security Constraint Columns</a>.</li> </ul>

## Ownership Privileges

The system implicitly grants ownership of any object to the owner of the space that contains the object. The owning user is the parent and any users contained in the owner space are child users. In turn, a child becomes the parent of any new users it creates. All owners and parents within the ownership hierarchy implicitly possess certain privileges on all lower-level objects contained in the space they own.

Implicit privileges for an owner/parent are similar to the privileges a creator automatically receives on a created object, as listed in [Privilege Dictionary](#), except that the system does not insert rows for implicit privileges in the DBC.AccessRights table as it does for a creator.

Ownership privileges normally include the discretionary privilege to grant full access on any owned object to other users, unless the object is protected by row level security, in which case user access to the object is limited by security constraint assignments. Owners do not have the privilege to administer security constraints unless they are granted the CONSTRAINT DEFINITION and CONSTRAINT ASSIGNMENT privileges. See [Implementing Row Level Security](#).

Ownership is subject to these additional rules:

- You cannot revoke ownership privileges.
- Privileges implicitly available to an owner are not all inclusive, but an owner/parent may grant itself additional privileges on any objects that its child users own.
- A user does not own itself, and therefore does not have implicit privileges on itself. Created users do receive some automatic privileges. See [Automatic Privileges](#).
- Although the DBC.AccessRights table does not list ownership privileges, these privileges are subject to access logging, if it is enabled. For information on access logging, see [Monitoring Database Access](#).

Site security policy must take into account the ownership hierarchy and resulting implicit privileges in setting guidelines for creating databases and users.

For further information on creating databases, see *Teradata Vantage™ - Database Administration*, B035-1093.

## Giving Ownership

You can use the GIVE statement to transfer ownership of a database or user to a new owner, who may or may not be part of the current parent-child hierarchy. The GIVE statement transfers to the recipient the specified database or user space, and also all of the databases, users, and objects owned by that database or user.

## Related Information

For information on ownership of users, databases, and objects created by directory-based users, see [About Directory User Characteristics](#).

## Explicit Privileges

Use the GRANT or REVOKE statement to explicitly provide or retract one or more privileges for a database object. See [Working with User Privileges in Teradata Vantage](#).

## Automatic Privileges

Type of Automatic Privilege	Explanation
User DBC	The initial user in the system, user DBC, automatically has all database privileges. User DBC uses these privileges to create the administrative users and explicitly grant privileges to them. Administrative users can pass these privileges down to the users, roles, and databases they create.
The creator (and in some cases, the modifier) of a database, user, or object	When a user with the CREATE DATABASE or the CREATE USER privilege creates a new database or user, Vantage automatically grants that user creator privileges on the created object and any space that it owns. Creator privileges differ from ownership privileges because the FROM <i>dbname</i> option of the CREATE DATABASE or CREATE USER statement allows use of space owned by another user. For a list of associated privileges, see <a href="#">Privilege Dictionary</a> . For CREATE options, see <i>Teradata Vantage™ - SQL Data Definition Language Syntax and Examples</i> , B035-1144.
A newly created user or database	The database automatically grants a set of privileges to newly created users within their defined user space. An owner or creator can revoke the automatic privileges for a newly created user. For a list of associated privileges, see <a href="#">Privilege Dictionary</a> .

## Checking Privileges in Data Dictionary Views

You can query the Data Dictionary to return information about privileges.

View	Description
DBC.UserRightsV[X]	The privileges granted to each user
DBC.AllRoleRightsV	The privileges granted to each role
DBC.RoleMembersV	Lists each role and its members

For more information on these views, see *Teradata Vantage™ - Data Dictionary*, B035-1092.

## Working with User Privileges in Teradata Vantage

You can grant database privileges directly to users, or grant privileges to roles and then assign role membership to users.

## About Privileges for User Types

You can assign privileges to database users according to the user type.

**Note:**

See also [About Database User Types](#).

User Type	Method for granting privileges
Permanent user	<ul style="list-style-type: none"> <li>Grant privileges directly to the user.</li> <li>Create roles and grant privileges to them. Then grant membership in one or more roles to each user (recommended).</li> </ul>
Directory-based user	<ul style="list-style-type: none"> <li>Map each directory user to one or more database users that already have database privileges.</li> <li>You can optionally create external roles and grant privileges to them. Then map each directory user to one or more of the external roles.</li> </ul> <p><b>Note:</b> The system registers objects created by a directory user in the data dictionary, with the mapped permanent user as the owner and creator.</p>
Auto provisioned user	<ul style="list-style-type: none"> <li>Set the AutoProvision DBSControl flag to true.</li> <li>In the directory create an external role or profile for auto provisioned users.</li> <li>Create matching roles and profiles in the database.</li> <li>Grant privileges to the external roles, if created.</li> <li>Map the directory users to the external role or profile.</li> </ul> <p><b>Note:</b> The privileges given to the auto provisioned account are determined by the external role to which the directory user is assigned. If an auto provisioned directory user is assigned to an external role and is also granted a role in the database, the user is allowed to have the privileges of both roles. However, the user is externally authenticated, so only external roles are active for the session. A granted role must be explicitly enabled. If the directory principal is not assigned to a role, the user inherits privileges from EXTERNAL_AP (a system user).</p>
Proxy user	<ul style="list-style-type: none"> <li>For proxy users that are either permanent database users or users unknown to the database, you can specify one or more roles in the GRANT CONNECT THROUGH statement that defines the proxy.</li> <li>For proxy users that are also permanent database users: <ul style="list-style-type: none"> <li>You can specify WITHOUT ROLE to use the privileges granted to the permanent user</li> <li>You can assign row level security constraints to the permanent user or the user profile. Proxy user sessions use the profile constraints, if assigned. If no constraints are assigned in the profile, the session uses the user constraints. The user can also use the SET SESSION CONSTRAINT command to access any assigned security constraints.</li> </ul> </li> </ul>

## Related Information

For information on...	See...
middle-tier application user requirements	<a href="#">Working with Middle-Tier Application Users.</a>
assigning security constraints to permanent users and profiles	<a href="#">Working with Constraint Assignments.</a>
how the system processes constraint assignments for proxy users	<a href="#">Session Constraint Values for Trusted User Applications and Proxy Users.</a>

## Working with GRANT and REVOKE Statements

The GRANT statement has the following variations:

This Statement...	Grants or Revokes...
GRANT/REVOKE (SQL Form)	database privileges to a user or role.
GRANT/REVOKE (Monitor Form)	monitoring privileges for a user or role. See <i>Teradata Vantage™ - SQL Data Control Language</i> , B035-1149.
GRANT LOGON/ REVOKE LOGON	logon privileges to a user. For usage, see <a href="#">About Logon Privileges</a> .
GRANT ROLE/REVOKE ROLE	membership in a role for one or more users or other roles.
GRANT CONNECT THROUGH/ REVOKE CONNECT THROUGH	<ul style="list-style-type: none"> <li>Trusted user status to a middle-tier application.</li> <li>CONNECT THROUGH privileges to proxy user(s).</li> <li>Use of roles to the specified proxy users.</li> </ul>

For GRANT and REVOKE statement syntax and options, see *Teradata Vantage™ - SQL Data Control Language*, B035-1149.

## Privileges Required to Use GRANT and REVOKE

The owner of an object has the implicit privilege to grant/revoke any privilege on that object to/from any user, unless the object is a view, macro, or stored procedure. Owners of these objects require the same grantor privileges required for non-owners.

Non-owners must be granted a privilege on an object, WITH GRANT OPTION, or for roles WITH ADMIN OPTION, to grant/revoke privileges to/from another user.

## Granting Privileges to Permanent Database Users

Grant privileges directly to users with the GRANT statement, documented in *Teradata Vantage™ - SQL Data Control Language*, B035-1149.

You must grant some privileges directly to users:

- For privileges that you cannot grant to a role, see [Limitations on Using Roles](#).
- For a few users with truly unique needs, it may be more efficient to grant required privileges directly to the individual users.

Where possible, avoid granting privileges directly to users. Instead, grant privileges to roles and then grant role membership to users with similar database access needs. See [Using Roles to Manage Privileges](#).

## Granting Privileges to PUBLIC

By default, all permanent database users have privileges of PUBLIC, a system-generated user. For information on PUBLIC, see [Default PUBLIC Privileges](#).

## Granting Privileges in a Unity Environment

When users log on through Unity, a session may address any one of the connected Teradata Vantage systems, so user privileges should be the same on all database systems. If some users can also log on directly to a database system, for example, to perform administrative duties or to access data that is not shared with other systems, you may need to grant special privileges to a few users for that system.

For information about Unity, see *Teradata® Unity™ Installation, Configuration, and Upgrade Guide for Customers*, B035-2523 and *Teradata® Unity™ User Guide*, B035-2520.

## Granting Privileges for BAR Users

For third-party backup and restore solutions compatible with Teradata Vantage, the system checks access privileges separately for each object accessed by a BAR operation. This allows for the possibility that some BAR tasks assumed to be complete are actually incomplete due to the BAR user lacking the necessary access privileges. Make sure that BAR users have the necessary privileges on all objects in BAR jobs they run.

For BAR jobs using Teradata Data Stream Architecture (DSA) the system checks all access privileges for a job and prevents the job from running if the user does not have the necessary access privileges on all objects in the job.

## Using Roles to Manage Privileges

Role privileges add to any privileges you grant directly to users.



---

**Note:**

Security constraint privileges and overrides are assigned rather than granted. See [Assigning Security Constraints in a CREATE PROFILE Statement](#).

---

Granting privileges to roles and then granting role membership to users offers these advantages:

- Standardizes privileges for users with a similar job description
- Reduces the time required to assign the privileges, compared with granting privileges to individual users
- Reduces the time the system takes to check user privileges at logon

You can grant one or more roles to one or more users or roles, therefore:

- A role can have many members.
  - A user or role can be a member of more than one role.
- 

**Note:**

The database allows only a single level of role nesting, that is, a role that has a member role cannot also be a member of another role. Members of the grantee role (the top level role) also have all the privileges in the nested role

---

When you grant a privilege to an existing role, it immediately affects any role member for which the role is currently active in a session.

## Limitations on Using Roles

You cannot grant certain privileges to a role:

- CREATE ROLE
- DROP ROLE
- CREATE PROFILE
- DROP PROFILE
- CREATE USER
- DROP USER
- CTCONTROL
- OVERRIDE privileges
- WITH GRANT OPTION (membership in a role cannot confer the ability to grant any of the privileges it contains to other users or roles)

**Note:**

Instead of WITH GRANT OPTION, you should use WITH ADMIN OPTION for roles. A user granted WITH ADMIN OPTION on a role can:

- Drop the role
- Grant the role to other users and roles
- Grant the role to another user with the WITH ADMIN OPTION
- Revoke the role from a grantee

WITH ADMIN OPTION does not provide the ability to grant or revoke privileges to or from the role or to any members of the role.

## Creating Roles

Use the CREATE ROLE statement to create a roles. When you create a role, you automatically receive creator privileges, including the DROP ROLE and WITH ADMIN OPTION privileges. Implicit ownership privileges do not apply.

**Note:**

Creator privileges do not give you the privilege to assign a default role to a user. These parameters are specified in the DEFAULT ROLE clause of CREATE/MODIFY USER, for which you need the CREATE USER or DROP USER privilege, respectively.

### Example: Create a Role and Grant it SELECT Privilege

Create a role for general users, grant it SELECT privileges on the entire Accounting database, and grant role membership to the general users Alan, Betty, David, and Ellen:

```
CREATE ROLE AcctUserRole;
GRANT SELECT ON Accounting TO AcctUserRole;
GRANT AcctUserRole TO Alan, Betty, David, Ellen;
```

### Example: Create a Role and Grant Membership to a User

Create a role for updating the accounts payable table, grant it data change privileges on the Accounting.AccPay view (an updatable view), and then grant role membership to a single high-level accounting user, Charles:

```
CREATE ROLE AcctUpdateRole;
GRANT SELECT, UPDATE, INSERT, DELETE ON Accounting.AccPay TO AcctUpdateRole;
GRANT AcctEndUserRole, AcctUpdateRole TO Charles WITH ADMIN OPTION;
```

**Note:**

WITH ADMIN OPTION allows Charles to grant, revoke, or drop either of the specified roles.

**Example: Grant CREATE ROLE to Administrators**

Grant CREATE ROLE privileges to administrators to perform role administration tasks:

```
GRANT CREATE ROLE TO Admin2, Admin3;
```

Assistant administrators Admin2 and Admin3 can now submit the following commands:

```
CREATE ROLE Role3;
GRANT Role1, Role3 TO Francis;
REVOKE Role1 FROM David;
DROP ROLE Role2;
```

**Related Information**

Information on...	Is available in...
using external roles for directory users	<a href="#">Using Roles for Directory Users</a> .
CREATE ROLE syntax and options	<i>Teradata Vantage™ - SQL Data Definition Language Syntax and Examples</i> , B035-1144.
usage notes for CREATE ROLE	<i>Teradata Vantage™ - SQL Data Definition Language Detailed Topics</i> , B035-1184.

**Dropping a Role**

If you drop a role while a member user is logged on, the user immediately loses the privileges granted to that role.

The user must use the SET ROLE statement to reactivate some or all of the remaining roles within a session.

**Assigning the Default Role**

For users with role assignments, you must set the default role in a CREATE USER or MODIFY statement. See [Creating a Database User](#).

**Dropping the Default Role**

If you drop the default role for a user, the system no longer defaults to a role when the user logs on, and the user must use the SET ROLE statement to activate a role.

The system records DROP statements in the Data Dictionary. However, a DROP ROLE statement does not affect the user rows in DBC.Dbase, so the system does not reset the corresponding default role setting for affected users. When an affected user next logs on, the system does not return an error or warning if the role does not exist.

You can reset the default role for a user with the MODIFY USER statement.

## Related Information

Information on...	Is available in...
the syntax to CREATE, DROP, or SET a role, or to assign a default role as part of a CREATE USER statement	<i>Teradata Vantage™ - SQL Data Definition Language Syntax and Examples</i> , B035-1144.
the syntax required to GRANT or REVOKE a role	<i>Teradata Vantage™ - SQL Data Control Language</i> , B035-1149.
roles for directory users	<a href="#">Using Roles for Directory Users</a> .

## Working with Roles for Proxy Users

Proxy users have role-based privileges defined in the GRANT CONNECT THROUGH statement that creates them. See [Setting Up Trusted User Applications and Proxy Users \(Recommended\)](#).

For further information on the use of the SET QUERY\_BAND statement for proxy user sessions, see *Teradata Vantage™ - SQL Data Definition Language Detailed Topics*, B035-1184.

## Using the SET QUERY BAND Statement to Enable Session Proxy Roles

You must set up the trusted user application to issue a SET QUERY\_BAND statement similar to the following when proxy users log on to Teradata Vantage:

```
SET QUERY_BAND = 'PROXYUSER=appluser;PROXYROLE=<role>;' FOR SESSION;
SEL cola, colb FROM database1.table1;
SEL cola, colb FROM database2.table2;
...
SET QUERY_BAND = NONE FOR SESSION;
```

The system examines both the SET QUERY\_BAND statement and the role specification in the originating GRANT CONNECT THROUGH statement to determine the operant role.

SET QUERY_ BAND statement	GRANT CONNECT THROUGH Statement	Role Used by the Trusted Session
PROXYROLE is omitted	WITHOUT ROLE	The default role for the matching permanent database user.
	WITH ROLE <role(s)>	All roles specified for the proxy user in the GRANT CONNECT THROUGH statement for the user.
PROXYROLE = ALL	WITHOUT ROLE	All roles granted to the matching permanent database user.
	WITH ROLE <role(s)>	All roles specified for the proxy user in the GRANT CONNECT THROUGH statement for the user.
PROXYROLE = <i>rolename</i>	WITHOUT ROLE	The named role, if the matching permanent database user is a member of the named role.
	WITH ROLE <role(s)>	The named role, if the role is specified in the GRANT CONNECT THROUGH statement for the proxy user.
PROXYROLE = NONE or PROXYROLE = NULL	WITHOUT ROLE	The current role is set to NULL. The proxy user inherits the default role for the matching permanent database user.
	WITH ROLE <role(s)>	The system returns an error. The SET QUERY_BAND statement must specify a role from the GRANT CONNECT THROUGH statement.

For further information on the use of the SET QUERY\_BAND statement for proxy user sessions, see *Teradata Vantage™ - SQL Data Definition Language Syntax and Examples*, B035-1144 and *Teradata Vantage™ - SQL Data Definition Language Detailed Topics*, B035-1184.

## Using Roles for Directory Users

The use of roles by directory users depends on the setting of the AuthorizationSupported property:

- When the AuthorizationSupported property is set to no, directory users can log on using a username that matches a database username. The directory user has access to all roles in which the matching database username is a member.
- When the AuthorizationSupported property is set to yes, directory users are authorized privileges according to the roles (and users) to which they are mapped in the directory.

## Implementing Roles for Directory Authorization of Database Privileges

1. Create external roles as shown in [Creating and Dropping External Roles](#).
2. Review directory user management options and select a user provisioning strategy. See [Directory Management of Database Users](#).

3. Create one or more directory role objects with names that match Teradata Vantage external roles and map the roles to directory group objects.

For information, see [Provisioning Directory Users with Teradata Schema Extensions](#) or [Using Native Directory Schema to Provision Directory Users](#).

---

**Note:**

Since roles are assigned by mapping instead of role grants, assignments cannot include WITH ADMIN OPTION.

---

**Note:**

Additional considerations apply when configuring directory authorization in a Unity environment. See *Teradata® Unity™ Installation, Configuration, and Upgrade Guide for Customers*, B035-2523 and *Teradata® Unity™ User Guide*, B035-2520.

---

## Creating and Dropping External Roles

You can specify EXTERNAL ROLE in the standard CREATE/DROP ROLE syntax to create external roles for directory users. The user that executes a CREATE/DROP EXTERNAL ROLE statement must have CREATE ROLE and DROP ROLE privileges. For example:

```
CREATE EXTERNAL ROLE  ext_role_name;
```

or

```
DROP EXTERNAL ROLE  ext_role_name;
```

---

**Note:**

If you drop a database role while including EXTERNAL in the syntax, or dropping an external role without including the EXTERNAL term, the system returns an error, for example:

```
DROP EXTERNAL ROLE dbrole;
Failure 5933: Role being dropped is not an external role

DROP ROLE extrole;
Failure 5934: Role being dropped is an external role
```

---

The system records external roles in the data dictionary, along with database roles, but when you map an external role to a directory user, the system does not insert a row in DBC.RoleGrants.

The method for granting privileges to an external role is similar to granting privileges to a database role. See [Creating Roles](#).

A user can occupy a maximum of 50 roles. If the maximum is exceeded, an error is reported.

## About Default Session Roles for Directory Users

For directory users, determination of the default role is more complex:

- All external roles mapped to the directory user are enabled.
- If the directory user is not mapped to any roles, but mapped to a permanent database user, the default role for the mapped permanent user is enabled.
- If the directory user is not mapped to either database roles or users, the directory user has PUBLIC privileges.
- If you drop an external role while directory users are logged on, the logged on users with that role immediately lose the privileges granted to the role.

## Effects of Dropping an External Role on Directory User Role Privileges

If you drop an external role, logged-on users to which that role is mapped immediately cease lose the privileges defined in the role.

## Effects of Changing Directory Role Assignments

Unlike database role grants and revokes, if you map or unmap external roles in the directory, it does not affect a session in progress. Only sessions that log on after you change a directory role are affected by the change.

## Switching Roles During a Session

Each permanent database user has a default role that is active for every user session. Users have access to other roles to which they have membership using the following requests:

- SET ROLE *role\_name* activates a specific role
- SET ROLE ALL. activates all roles

Directory users, including auto provisioned directory users, can also use the SET ROLE *role\_name* or SET ROLE ALL request to enable their mapped permanent user roles in addition to their external roles.

After changing the active role for a session, directory users can revert back to the initial set of directory-assigned roles by issuing a new request, SET ROLE EXTERNAL.

When a directory user has a large number of role mappings, the user can enhance performance by selecting a single role, using the SET ROLE *role\_name* request, so the system does not have to authorize all the role privileges for the user.

## Other Options for Restricting Database Access

Granting database privileges may not always be the best way to restrict database access. Teradata Vantage also provides the following optional methods for limiting user access.

### Restricting User Access Using Views

Database tables may contain information that is needed by the general user population, as well as other information to which access must be highly restricted.

You can create:

- Views to omit columns containing proprietary data, while including other columns.
- Multiple views for the same table, where:
  - Many users have SELECT privileges on a view of the entire table
  - Some users have INSERT, UPDATE, or DELETE privileges on a view that includes only part of the table

#### Related Information

For information on...	See...
creating views	<i>Teradata Vantage™ - SQL Data Definition Language Syntax and Examples</i> , B035-1144.
using views	<i>Teradata Vantage™ - Database Administration</i> , B035-1093.

### Restricting User Access by Column in a Table or View

Restricting user access by column is more precise than assigning user privileges on views, and it avoids the unnecessary proliferation of views.

You can define column level access restrictions by specifying a comma-separated list of columns after INSERT, REFERENCES, SELECT, or UPDATE when assigning privileges on a view or table:

- In a GRANT statement, to allow only the specified privilege on a limited number of columns in a table or view.
- In a REVOKE statement, to limit broader privileges previously granted on a table or view.

For further information, see *Teradata Vantage™ - SQL Data Control Language*, B035-1149.

### Restricting User Access by Row in a Table or View

You can create access label categories, and levels within each category, and then assign them:

- To users or user profiles to determine user access privileges
- To tables to define row level access requirements

See [Implementing Row Level Security](#).



## Restricting User Access with Macros and Stored Procedures

Macros and stored procedures are prefabricated routines that perform specific job functions in the database. You can grant the EXECUTE privilege on a macro or stored procedure to a user, and limit the actions the user can perform on a database object to those actions allowed by the routine.

Macros are useful for generating reports or repetitive data entry tasks, where the user only needs limited access to the database. You can construct macros to:

- Allow the user to provide values for certain columns, for example name, address, and phone number in a personnel table, without being able to select other columns.
- Gather data for reports without allowing the user to interact with the data.

You can construct stored procedures to perform complex tasks in the database, while the executing user has only the EXECUTE privilege on the stored procedure.

### Related Information

For information on...	See...
creating macros and stored procedures	<ul style="list-style-type: none"> <li>• <i>Teradata Vantage™ - SQL Data Definition Language Syntax and Examples</i>, B035-1144.</li> <li>• <i>Teradata Vantage™ - SQL Data Definition Language Detailed Topics</i>, B035-1184.</li> </ul>
using macros and stored procedures	<i>Teradata Vantage™ - Database Administration</i> , B035-1093.
additional privileges required by users who execute macros or stored procedures on row level security tables	<ul style="list-style-type: none"> <li>• <a href="#">Macro Requirements</a>.</li> <li>• <a href="#">Stored Procedure Requirements</a>.</li> </ul>

# Managing Database Passwords

Teradata Vantage imposes certain password format requirements for database passwords, and provides several options for managing password content and use.

## Prerequisites

The following password related procedures assume that the PasswordExpire control is set so that the temporary passwords assigned to newly created users expire at first logon, and periodically thereafter, as specified in [Creating the Security Administrator User](#).

## Password Management Process

1. Review password format requirements and inform all database users about the requirements, so they can successfully create a private password when the system prompts them. See [Working with Password Formatting](#).
2. Learn how to manage common password problems, for example, forgotten passwords and password lockouts. See [Managing Common Password Problems](#).
3. Evaluate password storage and retrieval options, and set up storage on Teradata Teradata Vantage clients, if needed. See [Using Teradata Wallet to Store and Retrieve Logon Elements](#).
4. Evaluate password control default values and options to develop a password control strategy. See [Working with Password Controls](#).
5. Optionally set password control parameters. See [Setting Password Controls](#).

## Working with Password Formatting

Each CREATE USER statement must specify a user password. The initial password assignment is temporary and set to expire at first logon, after which users create their own password.

Passwords must conform to default system password format requirements and any password controls that affect password structure. See [About Password Controls](#).

The system validates each password against default format requirements and settable format controls. An error message results whenever a password violates a password format rule.

## Password Formatting and Object Name Validation

The system validates password character content using the same rules that validate database object names.

For details about object naming, see *Teradata Vantage™ - SQL Fundamentals*, B035-1141.

Optionally, you can set the DBS Control NameValidationRule field to impose additional object name character restrictions. These restrictions also apply to passwords. Changing the NameValidationRule field does not affect existing passwords. The change only affects passwords that are subsequently created or changed.

For information about setting the DBS Control NameValidationRule field, see *Teradata Vantage™ - Database Utilities*, B035-1102.

## Password Sharing Among Character Sets

The system converts all passwords to UNICODE to make them sharable among all supported character sets, and then stores them in encrypted form. However, if a character in a password does not exist in the session (client) character set, the user cannot log on.

## Format Rules for Object Naming

There is no difference between the password rules for Japanese and non-Japanese systems. The system converts each password it receives from the session character set to UNICODE. Password format rules are enforced based on the UNICODE equivalent of the submitted password.

### Note:

Setting the DBS Control NameValidationRule field imposes additional object name character restrictions. For more information, see *Teradata Vantage™ - Database Utilities*, B035-1102.

In addition to the format rules that follow, password content is subject to the rules defined by setting password control options. See [Working with Password Controls](#).

The following table summarizes object naming rules.

Parameter	Description
Object name length	A maximum of 127 characters when expressed in UNICODE normalization form D.
Characters allowed in object names not enclosed in quotation marks	<p>An object name not enclosed in quotation marks must be composed of an identifier-start character followed by a sequence of identifier-start or identifier extend characters, up to the maximum object name length limit.</p> <p><b>Note:</b></p> <p>Characters in object names not enclosed in quotation marks must also be in the session character set.</p> <p>Identifier start characters must be contained in the session character set and belong to one of the following Unicode General Category classes:</p> <ul style="list-style-type: none"> <li>• Upper-case letters [Lu]</li> <li>• Lower-case letters [Ll]</li> <li>• Title-case letters [Lt]</li> <li>• Modifier letters [Lm]</li> <li>• Other letters ([Lo])</li> </ul>

Parameter	Description
	<ul style="list-style-type: none"> <li>• Letter numbers [NI]</li> </ul> <p>Or be one of the following characters:</p> <ul style="list-style-type: none"> <li>• NUMBER SIGN (U+0023)</li> <li>• DOLLAR SIGN (U+0024)</li> <li>• LOW LINE (U+005F)</li> <li>• INVERTED EXCLAMATION MARK (U+001A)</li> <li>• OVERLINE (U+203E)</li> <li>• EURO SIGN (U+20AC)</li> <li>• KATAKANA-HIRAGANA VOICED SOUND MARK (U+309B)</li> <li>• KATAKANA-HIRAGANA SEMI-VOICED SOUND MARK (U+309C)</li> <li>• FULLWIDTH NUMBER SIGN (U+FF03)</li> <li>• FULLWIDTH DOLLAR SIGN (U+FF04)</li> <li>• FULLWIDTH LOW LINE (U+FF3F)</li> </ul> <p>Identifier-extender characters must be in the session character set and belong to one of the following Unicode General Category classes:</p> <ul style="list-style-type: none"> <li>• Non-spacing marks [Mn]</li> <li>• Spacing combining marks [Mc]</li> <li>• Decimal numbers [Nd]</li> <li>• Connector punctuations [Pc]</li> <li>• Formatting codes [Cf]</li> </ul> <p><b>Note:</b> The MIDDLE DOT character (U+00B7) is also a valid identifier-extender character.</p>
Characters allowed only in object names that are enclosed in quotation marks	<p>A quoted string is required for object names that:</p> <ul style="list-style-type: none"> <li>• Have an identifier-extender character as the first character.</li> <li>• Include the white space character, SPACE (U+0020)</li> <li>• Are Teradata Vantage keywords</li> </ul> <p>In addition, object names that contain any character from the following classes must be enclosed in quotation marks, unless the character explicitly appears in the list of allowed characters:</p> <ul style="list-style-type: none"> <li>• Other, Control [Cc]</li> <li>• Other, Not Assigned [Cn]</li> </ul> <p><b>Note:</b> No characters in this category appear in UNICODE character repertoire.</p> <ul style="list-style-type: none"> <li>• Other, Private Use [Co]</li> <li>• Other, Surrogate [Cs]</li> <li>• Letter, Cased [LC]</li> <li>• Mark, Enclosing [Me]</li> <li>• Number, Other [No]</li> <li>• Punctuation, Dash [Pd]</li> <li>• Punctuation, Close [Pe]</li> <li>• Punctuation, Final quote [Pf] (may behave like Ps or Pe depending on usage)</li> <li>• Punctuation, Initial quote [Pi] (may behave like Ps or Pe depending on usage)</li> </ul>

Parameter	Description
	<ul style="list-style-type: none"> <li>• Punctuation, Other [Po]</li> <li>• Punctuation, Open [Ps]</li> <li>• Symbol, Currency [Sc]</li> <li>• Symbol, Modifier [Sk]</li> <li>• Symbol, Math [Sm]</li> <li>• Symbol, Other [So]</li> <li>• Separator, Line [Zl]</li> <li>• Separator, Paragraph [Zp]</li> <li>• Separator, Space [Zs]</li> </ul> <p><b>Note:</b> When used to enclose an object name, the beginning and ending quotation marks must be represented as a sequence of two QUOTATION MARK characters (U+0022). Each set of two quotation marks is counted as one character when calculating the name size limit.</p>
Disallowed characters	<p>The following characters cannot appear in an object name:</p> <ul style="list-style-type: none"> <li>• NULL (U+0000)</li> <li>• SUBSTITUTE character (U+001A)</li> <li>• REPLACEMENT CHARACTER (U+FFFD)</li> <li>• Compatibility ideographs (U+FA6C, U+FA6F, U+FAD0, FAD1, FAD5, FAD6, and FAD7)</li> </ul> <p><b>Note:</b> The setting of the DBS Control NameValidationRule field may define additional character restrictions. See <i>Utilities</i>.</p>
Other considerations	<p>These additional restrictions apply:</p> <ul style="list-style-type: none"> <li>• An object name consisting entirely of white spaces is not allowed.</li> <li>• A trailing white space is not considered part of an object name.</li> <li>• You can use the NameValidationRule field to restrict object name allowable characters to a subset of those normally allowed. For more information, see <i>Teradata Vantage™ - Database Utilities</i>, B035-1102.</li> </ul>

## Managing Common Password Problems

### Releasing Password Lockouts

Users may accidentally exceed the limit for erroneous logon attempts if they repeatedly specify a password incorrectly. This causes the system to lock the user from attempting further logons.

You can use the MODIFY USER or MODIFY PROFILE statement with the RELEASE PASSWORD LOCK option to release the password lock for a user.

```
MODIFY USER username AS RELEASE PASSWORD LOCK;
```

## Resetting Forgotten Passwords

If a user forgets his or her password, you can replace the forgotten password using the MODIFY USER statement:

```
MODIFY USER JDoe AS  
PASSWORD = mysecret;
```

The example MODIFY USER statement replaces the forgotten password with mysecret.

When the MODIFY USER statement resets the forgotten password to a temporary password, the system does not update the DBC.OldPasswords table or verify the temporary password against password reuse rules.

---

### Note:

Do not lose the password for user DBC. The system can lock out user DBC if it exceeds the MaxLogonAttempts setting, and only user DBC can modify the user DBC password. If this lockout occurs, contact Teradata Support Center to unlock DBC.

---

## Resetting the Password Expiration Interval

You should set passwords to expire as part of basic system setup, as described in [Creating the Security Administrator User](#). If you are unsure whether or not passwords are set to expire on your system, execute one of the following SQL statements to check the value:

- ```
SELECT ExpirePassword FROM DBC.SecurityDefaultsV;
```

This global value affects all users. If the value is 0, you can reset it to a non-zero value. Teradata recommends that you set the password expiration interval to a value between 90 and 270 days. Reset the value as follows:

```
UPDATE DBC.SysSecDefaults  
SET ExpirePassword=90;
```

---

### Note:

You must restart the database for this change to take effect.

---

- ```
SELECT profile_name, ExpirePassword FROM DBC.ProfileInfoV;
```

This value affects only users that are members of the profile. If the value for EXPIRE is 0 or NULL, MODIFY the profile to specify a non-zero value, as follows:

```
MODIFY PROFILE profile_name AS PASSWORD = (EXPIRE = 90);
```

If you change the expiration value in either a profile or the system table, the system causes all affected user passwords to expire.

## Tracking Changes to Passwords

The DBC.DBase table stores the date and time a password was last changed by a user.

Query the DBC.UsersV view, which references the DBC.DBase table and other system tables, selecting the columns PasswordLastModDate and PasswordLastModTime, to see the latest activities against passwords, for example:

```
SELECT UserName, PasswordLastModDate, PasswordLastModTime FROM DBC.Users;
*** Query completed. 6 total rows found. 3 columns returned.
*** Total elapsed time was 1 second.
```

UserName	PasswordLastModDate	PasswordLastModTime
-----	-----	-----
DBC	07/06/25	12:15:27
TDPUSER	07/06/25	12:24:25
SystemFe	07/06/25	12:24:31
SysAdmin	07/06/25	12:24:31
Crashdumps	07/06/25	12:29:01
Sys_Calendar	07/06/25	12:29:0

## Working with Password Controls

### About Password Controls

Teradata recommends that all sites enforce strong password controls. Teradata Vantage offers several options for controlling password format and usage.

The DBC.SysSecDefaults table contains a set of global controls that restrict the usage and content of passwords for all users, as shown in the following table.

Field Name	Description
<b>Usage Controls</b>	
<a href="#">ExpirePassword</a>	The number of days that must elapse before a password expires.
<a href="#">MaxLogonAttempts</a>	The number of erroneous logons the system allows before it locks the user out of the database.
<a href="#">LockedUserExpire (Password Lockout Time)</a>	The number of minutes elapsed before the system unlocks a locked user.

Field Name	Description
<a href="#">PasswordReuse</a>	The number of days that must elapse before a user can reuse an expired password.
<b>Format Controls</b>	
<a href="#">PasswordMinChar</a>	Sets the minimum number of characters required in a password.
<a href="#">PasswordMaxChar</a>	Sets the maximum number of characters allowed in a password.
<a href="#">PasswordDigits</a>	Determines whether ASCII digits are: <ul style="list-style-type: none"> <li>• Allowed in a password</li> <li>• Not allowed in a password</li> <li>• Required in a password</li> </ul>
<a href="#">PasswordSpecChar</a>	Indicates whether ASCII special characters are: <ul style="list-style-type: none"> <li>• Allowed in a password</li> <li>• Not allowed in a password</li> <li>• Required in a password</li> </ul> You can place tight restrictions on passwords to require that: <ul style="list-style-type: none"> <li>• Passwords must contain at least one ASCII alpha character</li> <li>• Passwords must contain a mixture of ASCII upper/lower case letters</li> <li>• No password can contain a database username</li> </ul>
<a href="#">PasswordRestrictWords</a>	Determines whether passwords are rejected if they contain words in the Restricted Words list.

**Note:**

The system creates the DBC.SysSecDefaults table, and provides default values for the password control parameters, during system initialization.

## Password Control Activation Options

All password controls have default values, which take effect automatically upon database system setup. Check the default values to see if they meet your site needs.

If you need to use a non-default password control value, you can reset the value:

- Globally, in DBC.SysSecDefaults
- For groups of users, in a profile

For more information, see [Setting Password Controls](#).

Changes to password control values take effect when passwords are created or changed following the change to the value.



## Password Control Recommendations

The following list represents a best-practice approach to password control for sites that need to maintain high security standards, for example, Common Criteria Compliance. See [Setting Up a System for Common Criteria Compliance](#).

- Passwords must be at least 8 characters in length and not exceed 30 characters (unless a longer password is required).
- Passwords must use a combination of alpha, numeric, and special characters.
- The username must be locked after 3 unsuccessful logons and stay locked for 5 minutes.
- The username cannot be part of the password.
- Passwords should expire at least every 90 days.
- Restrict reuse of passwords for at least 270 days.

## ExpirePassword

The ExpirePassword parameter specifies the number of days a password is valid. Teradata Vantage automatically adds this value to the password change date value maintained in the database row for each created user. Then it compares the result to the current date to determine if the password is still valid.

## Default Setting

The default setting for the ExpirePassword parameter is 0, that is, passwords do not expire.

## Allowable Values

The range of allowable values (in days) for ExpirePassword is 0 through 32767.

If you enter a negative value, the system accepts the UPDATE, but the value reverts to the default zero value at the next restart and writes an error message to the event log:

```
3698: SysSecDefaults table has a negative value in an integer column.
```

## Strategy

Consider the value of data in the database and the likelihood of a security breach, when you determine a suitable password lifetime for your system. The United States Department of Defense recommends a maximum password lifetime of no more than one year. Administrators commonly set this parameter at 3 to 6 months (90 to 180 days).

To set a temporary password, you must assign a non-zero value to ExpirePassword.

## Example: UPDATE Statement to Set Duration of Password

You can use this statement to reset the duration of password acceptance to 30 days:

```
UPDATE DBC.SysSecDefaults SET ExpirePassword = 30 ;
```

## Expired Password Effects on a Session

When a password expires, the system:

- Allows an existing session for the user with the expired password to continue.
- Allows the user to log on one session after the user password expires, if the logon string contains the correct expired password. The system limits the session to the MODIFY USER statement, to allow the user to establish a new password. After the user modifies the password, the system permits normal SQL Engine activity in the session.
- Prevents access to the database when a user makes a second attempt to logon with an expired password.
- Provides the following methods to update an expired password, depending on the application through which the user logs on:
  - Some applications prompt the user for a new password.
  - For applications that do not support password prompts, most allow the user to submit a MODIFY USER statement to create a new password.
  - For utilities such as MultiLoad that do not support MODIFY USER, the user must log off and log on again through a utility such as BTEQ that does support the statement.

## Using MODIFY USER to Replace an Expired Password

Use the following SQL statement to establish a new password:

```
MODIFY USER username AS PASSWORD = passwordname;
```

The password is immediately valid for the number of days indicated in the ExpirePassword field of the DBC.SysSecDefaults table.

For details, see the MODIFY USER syntax in *Teradata Vantage™ - SQL Data Definition Language Syntax and Examples*, B035-1144.

## MaxLogonAttempts

The MaxLogonAttempts parameter restricts the number of successive logon attempts allowed to a user who submits an incorrect password.

If the number of erroneous logon attempts reaches the value specified in DBC.SysSecDefaults.MaxLogonAttempts, the system locks out the User ID from further logon attempts, for the duration of the password lockout time, as described in [LockedUserExpire \(Password Lockout Time\)](#). Further erroneous attempts during the password lockout time generate Event log entries that contain User Locked as the event type.

## Default Value

The default value of the MaxLogonAttempts parameter is 0, that is, allow unlimited failed logon attempts.

Enter the number of allowable failed logon attempts into the DBC.SysSecDefaults table MaxLogonAttempts column.

If you enter a negative value, the system accepts the UPDATE, but the value reverts to the default zero value at the next restart, and writes an error message to the event log:

3698: SysSecDefaults table has a negative value in an integer column

---

### Note:

If user DBC exceeds the maximum allowable logon attempts, user DBC could be locked out. Then the only way user DBC can log on is through the TSTSQL console. Contact your Teradata support representative for more information about TSTSQL console.

---

## Allowable Values

The range of allowable values for MaxLogonAttempts is 0 through 127.

A negative value is not allowed and returns an error message.

## Example: UPDATE Statement to Set Maximum Failed Logon Attempts

You can use this statement to set the maximum number of failed logon attempts to 3:

```
UPDATE DBC.SysSecDefaults SET MaxLogonAttempts = 3 ;
```

The system does not accept a null value. A zero value prevents locking, and a negative value causes the system to write an error message to the event log at the next restart.

You must have the DROP USER privilege to execute the MODIFY USER statement.

For a discussion of the GRANT statement, see [Working with User Privileges in Teradata Vantage](#).

## Strategy

Consider the following factors when you specify the number of failed logon attempts to allow before locking out the user:

- If you allow a greater number of failed attempts, you increase exposure to unauthorized access by means of sophisticated, repetitive logon methods.
- Users make keyboard mistakes and forget passwords. If you allow too few failed logons, you may increase the number of legitimate user calls to the security administrator requesting password lock release.
- If your security policy allows shorter passwords, you can set the allowable number of failed logons lower. If your security policy requires longer passwords, you should set the allowable number of failed logons higher.
- If your database contains valuable or sensitive data, you may justify allowing fewer logon failures (for example, 2, or in extreme cases, 1). For less sensitive data, you may choose to allow more attempts (for example, 3 or 4, or no lockout at all).

## Rescuing Locked-Out Users

To break through the password lockout before the allotted time is elapsed, use the MODIFY USER statement to release the lock on the user, as shown in [Releasing Password Lockouts](#).

## PasswordReuse

The PasswordReuse parameter defines a time span during which a user cannot reuse a previously used password.

When a user changes a password, Teradata Vantage records the old password and the current date. The next time the user attempts to change the password, Vantage compares the new password to the current password.

- If they are equal, the system rejects the new password.
- If they are not equal, the system searches the list of old (expired) user passwords for a password equal to the proposed password. If the system finds a match and the number of days between the last password modify date and the current date is less than the PasswordReuse setting, Teradata Vantage does not allow the password change.

The system tracks expired passwords for the duration of the PasswordReuse setting.

## Default Value

The default value for the PasswordReuse parameter is zero, that is, allow immediate reuse.

## Allowable Values

The range of allowable values (in days) for PasswordReuse is zero through 32767. A zero value allows an immediate reuse of the password.

If you enter a negative value, the system accepts the UPDATE, but the value reverts to the default zero value at the next restart and writes an error message to the event log:

```
3698: SysSecDefaults table has a negative value in an integer column
```

## Strategy

The United States Department of Defense recommends that you do not allow reuse of passwords for at least 6 months after they expire, or for as long as the password lifetime, whichever is greater.

## Example: UPDATE Statement to Set PasswordReuse Lock

You can use this statement to set the PasswordReuse lock duration to 60 days:

```
UPDATE DBC.SysSecDefaults SET PasswordReuse = 60 ;
```

## Password History

To aid in researching password reuse status, Teradata Vantage saves all previously used passwords in the DBC.OldPasswords table. When users successfully change their password, the system:

- Writes a row containing the current password to DBC.OldPasswords.
- Deletes old password rows for the user with a date earlier than the current date minus the PasswordReuse time span from DBC.OldPassword.

### Note:

If you reset a user password, the system does not enforce any PasswordReuse restriction that would normally apply to that password. PasswordReuse restrictions only apply when users reset their own passwords.

The DBC.OldPassword table contains the following information.

Column	Description
UserName	Identity of the user to which the password was assigned.
PasswordDate	Date the password was changed for the user.

Column	Description
EncryptionFlag	Identifies whether the password is encrypted by DES or SHA-256.
PasswordSalt	SHA Standard seed needed to encrypt the password.
EncryptedPassword	Encrypted password string.
EncryptedPasswordLength	<ul style="list-style-type: none"> <li>• DES encrypted passwords = 8 bytes</li> <li>• SHA-256 passwords = 32 bytes</li> </ul>

## LockedUserExpire (Password Lockout Time)

The LockedUserExpire parameter sets the length of time the system locks out a user for exceeding the maximum number of logon attempts (MaxLogonAttempts).

### Default Value

The default value for LockedUserExpire is zero, that is, do not lock the user for submitting an erroneous password.

### Allowable Values

The acceptable range of values for the LockedUserExpire parameter is zero through 32,000 minutes (about 23 days). You can specify an indefinite lockout by entering a value of -1.

### Strategy

Consider the following when specifying the password lockout time:

- Users sometimes make mistakes entering passwords. Do not lock users out for long periods if the MaxLogonAttempts parameter allows only one or two erroneous attempts.
- A common security threat initiates erroneous logons to deny service to legitimate users. If the lockout time is long, such a process could quickly lock out all users.
- You can specify an indefinite lockout by entering a value of -1.
- You can reset the value to zero to immediately release the lock. This action disables the password lockout time feature until the value is reset.

## Example: UPDATE Statement to Set Lock Duration

You can use this statement to set the duration of user lock to 2 minutes:

```
UPDATE DBC.SysSecDefaults SET LockedUserExpire = 2 ;
```

## PasswordMinChar

The PasswordMinChar parameter sets the minimum required characters for a valid password.

### Default Value

The default value of PasswordMinChar is 1, that is, the password must have at least one character.

### Allowable Values

The valid range of values for PasswordMinChar is between 1 and PasswordMaxChar.

The value cannot be less than the minimum number of characters implied by other rules. For instance, if at least 1 digit, 1 special character, and a mixture of upper and lower case letters is required by other option settings, then the PasswordMinChar setting must be at least 4.

If you enter a zero or a negative value, the system replaces the value with the system default value and writes an error message to the event log at the next restart.

### Strategy

The United States Department of Defense recommends that passwords consist of 8 or more characters. When you specify a password length, consider a longer password if security is critical. However, keep in mind that because it is more difficult to remember a longer password, the user is more likely to write it down rather than memorize it, and users should not write down passwords.

### Example: UPDATE Statement to Set the Minimum Number of Characters in a Password

You can use this statement to set the minimum number of characters in a password:

```
UPDATE DBC.SysSecDefaults SET PasswordMinChar = 8 ;
```

## PasswordMaxChar

The PasswordMaxChar parameter sets the maximum number of characters allowed in a valid password.

## Default Value

The default value of PasswordMaxChar is 127 characters.

## Allowable Values

The valid range of values for PasswordMaxChar is 1-127 characters.

The lower end of the PasswordMaxChar range is subject to the PasswordMinChar limit.

The value cannot violate other password control rules, for example, if at least 1 digit, 1 special character, and a mixture of upper and lower case letters is required by other option settings, the PasswordMaxChar setting must be at least 4.

If you enter a zero or a negative value, the system replaces it with the system default value and writes an error message to the event log at the next restart.

## Strategy

When you specify the maximum password length, keep in mind that some users may try to create a password of maximum length. Users are more likely to write down long, hard-to memorize passwords, which poses a security risk.

## Example: UPDATE Statement to Set the Maximum Number of Characters in a Password

You can use this statement to set the maximum number of characters in a password:

```
UPDATE DBC.SysSecDefaults SET PasswordMaxChar = 12 ;
```

## PasswordDigits

The PasswordDigits parameter determines how you can use the digits 0 through 9, in a password.

### Default Value

The default value for the PasswordDigits parameter is Y, that is, digits are allowed in a password.

### Allowable Values

The acceptable values for the PasswordDigits parameter are:



- Y = Digits are allowed
- N = Digits are not allowed
- R = At least one digit is required

**Note:**

The PasswordDigits value is not case sensitive.

If you enter a value other than Y, N, or R, the system reverts to the default value and writes an error message to the event log:

```
3699: SysSecDefaults table has an invalid character value in a char column
```

**Note:**

You cannot use the digits zero through 9 as the first character in a password unless the entire password is enclosed in double quotation marks, for example:

```
password="78password"
```

**Strategy**

Many passwords are relatively easy for an intruder to guess, especially if some of the letters are known. You should force users to create passwords with one or more digits, to enhance password security.

**Example: UPDATE Statement to Set the Option for Digits in a Password**

You can use this statement to set the option for digits in a password:

```
UPDATE DBC.SysSecDefaults SET PasswordDigits = 'R' ;
```

**PasswordSpecChar**

The PasswordSpecChar parameter determines how you can use ASCII special characters in a password, with these options:

- Special characters are allowed/not allowed/required
- Passwords must contain at least one alpha character
- No password can contain the database username
- Passwords must contain a mixture of upper/lower case letters
- Upper includes all UNICODE characters in General Category Class Lu.
- Lower includes all UNICODE characters in General Category Class Ll.
- Alpha includes all UNICODE characters that are in either Upper (Lu) or Lower (Ll)

- Special indicates characters that are neither Alpha nor any of the characters 0 through 9 (U+0030 – U+0039). By this definition, un-cased letters (General Category Class Lo) are considered Special (for example, Arabic, Chinese or Hebrew characters). Likewise numeric digits (General Category Class Nd) other than the characters 0 through 9 (U+0030 – U+0039) are classified as Special.

For consistency in handling characters, the system checks PasswordSpecChar based on the NFC representation of the password string. For example, the character 'é' (U+00E9) is a Special, a Lower, and an Alpha character.

## Default Value

The default value of the PasswordSpecChar parameter is Y, special characters are allowed but not required.

## Allowable Values

Teradata Vantage has simplified the setup of special character options by allowing a single character to represent each valid combination of the four options.

- Special characters are allowed in a password
- Username is allowed in the password string
- Alpha characters are allowed but not required
- Mixed upper and lower case characters are allowed but not required

The values are not case sensitive.

Use the following rule key when you choose the combination of special character options you want to use:

Status Indicator	Description
Y	Allowed but not required
N	Not Allowed
R	Required

If you enter a value other than Y, N, or R, the system reverts to the default value and writes an error message to the event log:

```
3699: SysSecDefaults table has an invalid character value in a char column
```

Specify the SpecChar single-letter option code with the set of status indicators that represents how you want to enforce the four SpecChar options.

SPECCHAR OptionCode	Included Special Character Option Rules			
	UserName	Mixture of Upper and Lower Case letters	At least One Alpha Character	Special Characters
N,n	Y	Y	Y	N
Y,y	Y	Y	Y	Y
A,a	Y	Y	Y	R
B,b	Y	Y	R	N
C,c	Y	Y	R	Y
D,d	Y	Y	R	R
E,e	Y	R	R	N
F,f	Y	R	R	Y
G,g	Y	R	R	R
H,h	N	Y	Y	N
I,i	N	Y	Y	Y
J,j	N	Y	Y	R
K,k	N	Y	R	N
L,l	N	Y	R	Y
M,m	N	Y	R	R
O,o	N	R	R	N
P,p	N	R	R	Y
R,r	N	R	R	R

## Strategy

You can force users to create passwords with one or more of the special character options to enhance password security. When deciding how elaborate the password special character requirements should be for your system, consider that password special character requirements may make user passwords harder to remember and to type in at login.

## Example: UPDATE Statement to Require Special Characters in a Password

The following statement requires mixed case, at least one alpha character, and one special character in passwords:

```
UPDATE DBC.SysSecDefaults SET PasswordSpecChar = 'R' ;
```

## PasswordRestrictWords

The PasswordRestrictWords parameter determines whether a password is subject to the content restrictions defined in the RestrictedWords list. The system contains a default set of Restricted Words, and you can add others. See [Password Restricted Words](#).

The system trims special characters from a password before comparing it to the list of password restricted words. The definition of special characters is determined as follows:

- Upper includes all UNICODE characters in General Category Class Lu.
- Lower includes all UNICODE characters in General Category Class Ll.
- Alpha includes all UNICODE characters that are in either Upper (Lu) or Lower (Ll)
- Special indicates characters that are neither Alpha nor any of the characters 0 through 9 (U+0030 – U+0039). By this definition, un-cased letters (General Category Class Lo) are considered Special (for example, Arabic, Chinese or Hebrew characters). Likewise numeric digits (General Category Class Nd) other than the characters 0 through 9 (U+0030 – U+0039) are classified as Special.

A character with non-zero combining class is trimmed only if the character immediately preceding it is also trimmed. (Combining characters are characters that have a non-zero combining class and non-combining characters are those with a combining class of zero.)

For example, an accent is a combining character. If the character A is followed by an accent mark or two, the accent marks are not trimmed. Alternatively, if a multiplication sign is followed by two accents, the accents and the multiplication sign are trimmed.

## Default Value

The default value for the PasswordRestrictWords parameter is N, that is, do not restrict any words from being contained in a password string.

## Allowable Values

- Y = Words in the Restricted Words list are not allowed in a password string.
- N = Words in the Restricted Words list have no effect on the content of a password string.
- The values are not case sensitive.

## Example: UPDATE Statement Option to Restrict Words in a Password

You can use this statement to set the option to restrict words in a password:

```
UPDATE DBC.SysSecDefaults SET PasswordRestrictWords = 'Y' ;
```

If you enter a value other than Y or N, the system reverts to the default value and writes an error message to the event log:

```
3699: SysSecDefaults table has an invalid character value in a char column
```

## Strategy

Many passwords are relatively easy for an intruder to guess, especially if they contain common words or names. To enhance password security, users should create passwords that do not use common words or names.

## Adding and Removing Restricted Words

You can add words to the Restricted Words list:

```
INSERT into PasswordRestrictions values ('newrestrictedword')
```

### Note:

Although the default Restricted Words list is composed of English words, you can add words in any supported character set.

You can also remove words from the Restricted Words list:

```
DELETE from PasswordRestrictions WHERE (RestrictedWords='wordtobedeleted')
```

## Determining if a Password Contains a Restricted Word

1. Remove all ASCII numbers and ASCII special characters from the beginning and end of the password string.
2. Use the resulting string (the stripped password) in the query:

```
SELECT*FROM DBC.PasswordRestrictions
WHERE UPPER(RestrictWords)=UPPER('StrippedPassword')
```

## Setting Password Controls

### Using UPDATE to Manage Global Password Controls

You can set global password control options in the SysSecDefaults table with an UPDATE statement, for example:

```
UPDATE DBC.SysSecDefaults SET PasswordMinChar = 8 ;
```

### Using CREATE or MODIFY PROFILE to Set Password Controls for Users

You can use a CREATE or MODIFY PROFILE statement to override global password control parameters defined in SysSecDefaults, and apply a password control only to those users assigned to the profile. For information about the CREATE PROFILE and MODIFY PROFILE statements, documented in *Teradata Vantage™ - SQL Data Definition Language Syntax and Examples*, B035-1144.

### Effects of Profile-Based Password Controls

Password controls take effect differently when defined in a profile than when you modify the controls directly in the DBC.SysSecDefault table.

Password Control Parameter	Effects of Password Controls
EXPIRE	Applies to first level user to which the profile is assigned, and is effective immediately.
MINCHAR and MAXCHAR	Does not apply to first level user to which the profile is assigned, rather it applies only to the children of that user. Controls take effect at the first child user logon after the child is created or modified.  Use an UPDATE statement to apply password controls to individual primary users that CREATE/MODIFY the password controls of other users.
DIGITS	
REUSE	
SPECCHAR	
RESTRICTWORDS	
MAXLOGONATTEMPTS	Applies to first level users to which the profile is assigned, at the next logon attempt.
LOCKEDUSEREXPIRE	

## Related Information

For more information on . . .	See . . .
Administrative procedures for using profiles	<i>Teradata Vantage™ - Database Administration</i> , B035-1093.
Detailed syntax for creating, dropping, or modifying profiles and assigning them to users	<i>Teradata Vantage™ - SQL Data Definition Language Syntax and Examples</i> , B035-1144.
Profiles for directory-based users	<a href="#">About Assigning Profiles to Users</a> .

## Comparing Global and Profile-Based Password Controls

The security administrator can reset the password control parameters shown in the SysSecDefaults table in the following ways.

### Note:

Password controls do not affect externally authenticated users because the database does not validate their passwords.

Reset Method	Description	Effects
Change a parameter value using an UPDATE statement	Resets the control globally. <b>Note:</b> This is a global reset that causes the system to undo all default control values for affected parameters.	<ul style="list-style-type: none"> <li>Global password controls apply to all users defined in the database.</li> <li>DBC.SysSecDefaults table only at database startup. You must restart the database to enable changed values.</li> </ul> <b>Note:</b> Password changes can take place when: <ul style="list-style-type: none"> <li>You modify an existing user definition, or a user modifies his or her own user definition, using a MODIFY USER statement.</li> <li>A password expires and the user creates a new password.</li> </ul>
CREATE/MODIFY PROFILE	Resets global password control parameter(s) only for members of the profile. <b>Note:</b> Changes take effect at the next user logon.	Profile-based password controls apply only to the children of profile member users. To assign password controls to administrative users, you must assign a profile that contains the controls to a super administrator, who can then pass the controls down to other administrative users. Also see <a href="#">Effects of Profile-Based Password Controls</a> .

## Using Password Controls With Multibyte Characters

Teradata Vantage automatically converts all passwords to Unicode prior to applying password controls. Therefore, all character sets are subject to password controls.

## Viewing Current Password Control Settings

You can use a SELECT statement to check the current global default password control settings:

```
SELECT * FROM DBC.SecurityDefaultsV
```



# Logging on to Teradata Vantage

The following topics describe how to set up and manage user logons to Teradata Vantage. For information about optional logon controls, see [Using Logon Controls](#).

## Prerequisites

Implementing user logons to the database assumes that the following tasks are complete.

- A user authentication and authorization strategy is in place. See [Implementing User Authentication and Authorization](#).
- Password management policy is in place and the necessary password controls are enabled. See [Managing Database Passwords](#).
- Vantage clients are configured to connect to the database.
- Users understand the logon conventions for the Vantage client applications they use. See the user guide for each application.

## Logon Implementation Process

1. Review the default logon privileges. See [About Default Logon Privileges](#).
2. Review the descriptions of [Using Logon Elements](#) and [About Network Logons](#) and inform users how to:
  - Configure logon defaults as part of the setting up ODBC-based client applications.
  - Create proper logon syntax for CLIV2 applications such as BTEQ.
  - Assign logon attributes values for applications such as Teradata Parallel Transporter.
3. Log on with a stored password. See [Using Teradata Wallet to Store and Retrieve Logon Elements](#).
4. Complete any setup required for special logon environments. See:
  - [Logging On from Mainframe Systems](#)
  - [Logging On from Teradata Vantage Nodes](#)
  - [Logging On through Unity](#)
  - [Logging On from .NET Clients](#)
  - If a client is authenticating via Kerberos, the Kerberos installation has to be completed following the Kerberos vendor instructions. After the installation, configure Kerberos on the client system by following the guidelines in [Installing and Configuring Kerberos](#).
5. Determine logon error handling. See [Using Logon Error Handling Options](#).

## About Default Logon Privileges

All users that you create in the database automatically have database logon privileges, and can logon from any configured, connected client using Teradata authentication (TD2 mechanism).

Also see [About Logon Privileges](#).

## Using Logon Elements

To access the database, a user must submit a logon request. The request must specify some or all of the following elements, depending on system configuration and security policy:

- Authentication mechanism
- Tdpid
- Username
- Password
- Account string
- Domain or Realm

---

**Note:**

Teradata Vantage may provide some of these elements by default, or transfer them from other sources, depending on the system setup options you have exercised.

---

When a user submits a logon string, the mechanism (user-specified, or the default) authenticates the user and establishes a session based on user privileges. If the logon is successful, Teradata Vantage associates the username with a unique session number, under which the session runs until the user logs off.

## About Logon String Format Requirements

The logon string contains several elements that are subject to formatting requirements.

### User Name and Account String Format

Teradata Vantage logons may include a user name and account string, which are subject to validation of such parameters as length and allowable characters. The system validates name formatting when you initially specify the name, for example, when you create the user or account, not during logon.

For information on name formatting requirements, see *Teradata Vantage™ - SQL Fundamentals*, B035-1141.

### Password Format

The system checks the format of a database password when the password is created or updated in the database. For Teradata Vantage password format rules, see [Working with Password Formatting](#).

Format requirements for external authentication passwords depends on whether they are specified in the .logon or .logdata statement. See [Formatting of Logon Elements in .logon and .logdata Statements](#).

## Formatting of Logon Elements in .logon and .logdata Statements

In general, you can specify the user credentials in either the .logon or .logdata statement, however, certain logon specifications require use of the .logdata statement. For details, see [Working with Logon Syntax Elements](#).

## Object Names Acquired from External Authenticating Agents

User names forwarded to the database by outside authenticating agents are subject to Teradata Vantage object name format requirements, such as allowable characters and length limits, which otherwise are validated by the database. For example:

- For Single Sign-on or Sign-on As, Kerberos forwards a domain username to the database.
- After directory authentication, the authenticating agent may forward a directory username to the database for logging and auditing.
- A proxy user for a trusted session, if the proxy is not a permanent database user.

Be sure that object names that may be automatically forwarded to Teradata Vantage conform to object name size requirements. See *Teradata Vantage™ - SQL Fundamentals*, B035-1141.

## Determining the Authentication Mechanism for a Logon

TD2 is automatically set as the default mechanism on the database. If a session logon does not specify a mechanism the system uses TD2.

Users can specify an authentication mechanism, which overrides the default.

- In the logon string. See [About Network Logons](#).
- As part of a job script, for example, when using Teradata Parallel Transporter.
- In a connection field, for example, when making a connection to the Vantage Advanced SQL Engine through an ODBC application.

## Changing the Default Mechanism

You can designate a default authentication mechanism other than TD2 on a client (for information, see the appropriate TTU reference or user guide for your specific client), the Unity server (if used), or on the database, using the DefaultMechanism property. See [DefaultMechanism](#).

The system determines the default mechanism according to the following hierarchy:

1. Client default
2. Unity server (if used) default

3. Teradata Vantage system default, defined in the TDGSS configuration file

---

**Note:**

If the mechanism is disabled in the configuration file for the client, the Unity server (if used), or the database to which the user is connecting, the logon fails, regardless of how the mechanism was determined.

---

## About Mechanism Security Policy

You can define a security policy to limit the mechanisms available to a user, or to force the use of a specific mechanism. See [Configuring a Security Mechanism Policy](#). If a user logs on with a mechanism that violates applicable security policy, the logon fails.

## Specifying a Tdpid

The tdpid identifies a Teradata Vantage system, or Unity server (if used), to which the logon connects. Tdpids are assigned as part of setting up network connections between the Vantage system and its clients. For details, see the *Teradata Tools and Utilities Installation Guide* for the client operating system or the application user guide.

## Specifying the User Name

### Database User Name

A database user name for a logon can take the following forms:

- A simple user name, for example, username
- A user name with distinguishing information, for example, username@domain

---

**Note:**

User names such as username@domain must be created enclosed in double quotation marks. However, the quotation marks are optional when specifying the user name in a logon string.

---

For information on creating users, see [Creating Users and Granting Privileges](#).

### Directory User Name

Directory users must specify their directory user names and passwords.

- If the AuthorizationSupported property for the authentication mechanism is set to no, the directory username must match a database username.

- If the AuthorizationSupported property for the authentication mechanism is set to yes, the directory username does not have to match a database username.

See [Working with Directory User Management Options](#).

A username can contain up to 128 characters. If a directory user name exceeds this limit, the AuditTrailID (used in monitoring user access to the database) truncates the name. The AuditTrailID is logged in DBC.SessionTbl.

## Using Appended Domain Name

---

### Note:

Teradata strongly recommends that you do not use the Append Domain Name feature. Allowing different users to have the same username, even if they are in different domains, is not compatible with a strong security policy. If you are already appending domain names to distinguish identical usernames, discontinue the practice as soon as possible for better security. Reassign these users unique usernames.

---

Appending a domain name to a username ensures that every logon name is unique across all domains for users that are authenticated externally. You can configure the database to append the domain name for external authentication for mechanisms that provide domain information, including the following:

- KRB5
- SPNEGO

To check on whether the Append Domain Name feature is already set up, do the following:

1. Query the Append Domain Name value of the Gateway Control GDO -d option to determine what name the system uses to identify the user.
  - If Append Domain is set to no, the system uses the username contained in the logon.
  - If Append Domain is set to yes, the name the system uses depends on the mechanism:
    - If the mechanism does not provide a domain name, the system uses username.
    - If the mechanism provides a domain name, the system uses username@domain.
2. To change the current value, toggle it with the -F option for the gtwcontrol command:

```
gtwcontrol -F
```

For further information about the gtwcontrol utility, see *Teradata Vantage™ - Database Utilities*, B035-1102.

3. The database accepts appended domain names only if the corresponding usernames are defined in the database as username@domain, for example, for user "joe" in domain "domain1", you must define the user similarly to:

```
CREATE USER "joe@domain1" AS PERM=10000000, PASSWORD=pw1234;
GRANT LOGON ON ALL TO "joe@domain" WITH NULL PASSWORD;
```

**Note:**

Use this special format only for users that require an appended domain name.

## Specifying a Password

The system authenticates a user when it determines that the logon username and password match. The password is stored in an encrypted form in the Teradata Vantage Advanced SQL Engine, and in protected form in a Teradata wallet, if used.

## Submitting a Password for External Authentication

Users that are externally authenticated submit their external username and password. These passwords are not subject to Teradata Vantage password controls.

**Note:**

For mainframe-attached systems, a security exit in the TDP must acknowledge that the logon string for this username is valid without a password.

## Related Information

For information on...	See...
password format requirements and controls	<ul style="list-style-type: none"> <li>• <a href="#">Working with Password Formatting.</a></li> <li>• <a href="#">About Password Controls.</a></li> </ul>
using Teradata Wallet to recover a stored password	<a href="#">Using Teradata Wallet to Store and Retrieve Logon Elements.</a>
password protection	<a href="#">About Password Encryption.</a>
external authentication of users	<a href="#">About External Authentication.</a>
security exits	<i>Teradata® Director Program Reference, B035-2416.</i>

## Specifying an Account String

Users can specify an account string in a logon string to help track system resource usage, and to set session priorities in a busy system. If the logon does not include an account string, the system assigns a default value.

You can assign accounts to users in a CREATE/MODIFY USER statement. Each username can have one or more associated account strings. The first account string you assign is the default.

For information on...	See...
creating and using account strings	<ul style="list-style-type: none"> <li>• <a href="#">Creating Permanent Database Users</a>.</li> <li>• <i>Teradata Vantage™ - Database Administration</i>, B035-1093.</li> </ul>
account string syntax and options	CREATE USER information in <i>Teradata Vantage™ - SQL Data Definition Language Syntax and Examples</i> , B035-1144.

## Specifying a Domain or Realm

Logons using the LDAP mechanism must include the name of either the domain or realm (depending on the directory), where both of the following are true:

- The site elects to use SASL/DIGEST-MD5 authentication.

### Note:

The DIGEST-MD5 authentication protocol used by LDAP is deprecated. Teradata strongly recommends you use simple binding with TLS protection, and stop using DIGEST-MD5.

- The authenticating directory server offers more than one SASL realm.

You can use the .logdata statement to specify a domain or realm in the form:

```
realm=realM_name
```

If the logon string does not include a domain/realm value, and a value is required, the system defaults to the value stored in the LdapServerRealm property of the LDAP mechanism. If the LdapServerRealm property value is not correct, you can change the value in the configuration file or require that users enter the correct value as part of the logon. If the system defaults to an incorrect LdapServerRealm property value, or if the user submits an invalid value as part of the logon string, the system returns an error message.

Information on...	Is available in...
setting LdapServerRealm	<a href="#">LdapServerRealm [Deprecated]</a> .
specifying a domain/realm in a logon string	<a href="#">About Network Logons</a> .

## About Network Logons

You can logon from a network-attached Teradata client using one of the following methods.

Logon Type	Description
Command line	Used for BTEQ interactive mode. Enter logon elements interactively
ODBC connection	Used by ODBC applications. Set default logon parameters as part of configuring the ODBC driver connection. The system prompts for the user password.
Script	Used by scripted applications, for example, BTEQ and Teradata Parallel Transporter. Enter logon elements as script components, for example, Teradata Parallel Transporter scripts submit logon elements as operator attribute values.  <b>Note:</b> Logon syntax element names may vary among applications.

For mainframe logons, see [Logging On from Mainframe Systems](#).

## Using Logons with Different Authentication Methods

Content requirements and options for the logon string vary depending on the user authentication and authorization method. The following authentication methods are available to users that log on to Teradata Vantage.

- [Logging on Using Teradata Authentication and Authorization](#)
- [Logging on Using LDAP Authentication and Authorization](#)
- [Logging on Using Sign-on As](#)
  - with Teradata authorization
  - with directory authorization
- [Logging on Using Single Sign-on with Kerberos](#)
  - with Teradata authorization
  - with directory authorization
- [Logging on Using Teradata Negotiating \(TDNEGO\)](#)

External (non-Teradata) authentication requires completion of the set up tasks shown in [About External Authentication Controls](#) and [About External Authentication Requirements](#).

## Working with Logon Syntax Elements

Logon string information comprises three syntax elements. In interactive command line environments, such as Teradata Basic Query Utility (BTEQ), logon elements appear as discrete elements.

Syntax Element	Description
.logmech	Specifies the authentication mechanism.



Syntax Element	Description
	If a .logmech statement is not present, the logon uses the default security mechanism.
.logdata	<p>Optionally specifies user credentials for external authentication mechanisms.</p> <p><b>Note:</b> Use of the .logdata statement is required:</p> <ul style="list-style-type: none"> <li>• If the user is authorized in a directory and must specify user=<i>username</i>, profile=<i>profile_name</i>, or realm=<i>realm_name</i> to differentiate among user mappings.</li> <li>• For certain expressions of the user name and password, for example: <i>diruser@@password</i>.</li> </ul> <p>For details, see the topics for specific logon methods that follow this one.</p>
.logon	<p>Specifies the tdpid and user credentials for all authentication methods. For TD2 authentication, logon can also include the user account.</p> <p><b>Note:</b> Not usable for external authentication, where a user=<i>username</i>, profile=<i>profile_name</i>, or realm=<i>realm_name</i> specification is required. Instead, use .logdata.</p>

**Note:**

For applications that do not use a command line interface, logon element syntax requirements and options still apply, but you must configure the elements as defaults, or specify them in a connection GUI, depending the application you use.

Also see [Working with Logon Variations by Application](#).

## Logging on Using Teradata Authentication and Authorization

The system defaults to Teradata authentication and authorization.

### Example: TD2 Logon

```
[ .logmech TD2 ]
.logon tdpid/username,password[, "account" ]
```

**TD2**

[Required if TD is not default.] The Teradata authentication mechanism.

Default: TD2 unless set to another value.

***tdpid***

The logon connection.

If logging on directly to a database system, use one of the following:

- Database system *tdpid*
- Database host group *tdpid* (required if using host groups)

If logging on through Unity, use one of the following:

- Unity server *tdpid*
- Unity host group *tdpid* (required if the Unity server is set up for host groups)

***username***

The name of a user defined in the database.

***password***

The password for *username*.

***account***

[Optional] The account string.

## Logging on Using LDAP Authentication and Authorization

Directory users must specify the LDAP mechanism at logon. A configured LDAP-compliant directory performs authentication and authorization of the user.

For information on setup options for directory users, see [Directory Management of Database Users](#).

## LDAP Authorization Processing

The directory authenticates the directory user and then authorizes user privileges in the database according to these rules:

- The directory can only authorize privileges that are defined in the database.
- Directory users mapped to a Teradata Vantage user (including EXTUSER) inherit the database user privileges.
- If the directory user is mapped to database profiles or external roles, those privileges take precedence over any privileges inherited from mapping to a database user.

## LDAP Logon Format Examples

The LDAP mechanism supports two logon formats, based on the placement of user credentials in either the `.logdata` or `.logon` statement.

## Example: Logon with User Credentials in the .Logdata Statement

```
.logmech ldap
.logdata user_credentials[authorization_qualifier]
.logon tdpid/[,,"account"]
```

### Note:

The user credentials must be specified in the .logdata statement if there is an authorization qualifier or if required for the user credential format.

### Note:

Separate multiple .logdata specifications with white space.

## Example: Logon with User Credentials in the .Logon Statement

```
.logmech ldap
.logon tdpid/user_credentials[,,"account"]
```

## Explanation of LDAP Logon Format Examples

Syntax Element	Description
.logmech ldap	Specifies the authentication mechanism. Required unless LDAP is set as the default mechanism. LDAP is the only mechanism that supports directory authentication.
user_credentials	Specifies the directory username and password, using a format that is valid for the specifying statement. <b>Note:</b> You can specify user credentials in either the .logdata or .logon statement, except when you specify an authorization qualifier, which requires you to use the .logdata statement.  Valid credential formats for the .logdata statement: <ul style="list-style-type: none"> <li>• authcid= <i>diruser</i> password= <i>dirpassword</i></li> <li>• <i>diruser</i> @@<i>dirpassword</i></li> <li>• <i>diruser</i> password= <i>dirpassword</i></li> </ul> Valid credential formats for the .logon statement: <ul style="list-style-type: none"> <li>• <i>diruser,dirpassword</i></li> </ul> If the directory service is Active Directory, or when an identity map or identity search is configured, you can also specify: <ul style="list-style-type: none"> <li>• <i>diruser</i> password= <i>dirpassword</i></li> </ul>

Syntax Element	Description
	<p>For information on configuring identity map and identity search, see <a href="#">Optimizing Directory Searches</a>.</p> <p><b>Ensuring Correct Interpretation of UPNs</b></p> <p>For the logon <i>diruser,dirpassword</i>, if the <i>user</i> specification is “a@b” or a/b” or “a\b”, set LdapCredentialsUPN to interpret the user specification. See <a href="#">LdapCredentialsUPN</a>.</p> <ul style="list-style-type: none"> <li>If the LdapCredentialsUPN property is absent or set to yes (the default), the system treats the user specification as a UPN, which must conform to the rules of IETF 1964.</li> </ul> <p><b>Note:</b></p> <p>When LdapCredentialsUPN is set to yes, the UPN must appear in the logon as: “a\@b” or “a\b” or “a\\b”, where the added backslash (\) character shows the system how to handle the following character.</p> <ul style="list-style-type: none"> <li>If the CredentialsUPN property is set to no, the system disregards the special characters and considers the user specification to be an Authcid.</li> </ul>
<i>authorization_qualifier</i>	<p>Specifies authorization parameters. Required when:</p> <ul style="list-style-type: none"> <li>The directory user is mapped to multiple user or profile objects</li> <li>LDAP is set to use SASL/DIGEST-MD5 binding (the default), the directory offers more than one realm, and the value of the LdapServerRealm property is set to "" (the default).</li> </ul> <p><b>Note:</b></p> <p>The DIGEST-MD5 authentication protocol used by LDAP is deprecated. Teradata strongly recommends you use simple binding with TLS protection, and stop using DIGEST-MD5.</p> <p><b>Directory user mapped to multiple database users:</b></p> <p>If the directory user is mapped to more than one database user, specify one of the users in the form <code>user=database_username</code>.</p> <p><b>Directory user mapped to multiple profiles:</b></p> <ul style="list-style-type: none"> <li>If a directory user is mapped to more than one profile, specify <code>profile=profile_name</code> in the <code>.logdata</code> statement to identify the session profile.</li> <li>If the directory user is mapped to one or more database users, and also to a profile, the session defers to the separately mapped profile rather than the profile for the mapped database user.</li> </ul> <p><b>Directory offers multiple realms:</b></p> <p>Specify the realm as it appears in the directory, normally the fully qualified DNS name of the directory, for example:</p> <p><code>realm=directory_FQDNSName</code></p> <p>The system processes realm information as follows:</p> <ul style="list-style-type: none"> <li>If the logon does not specify a realm, and the LdapServerRealm property value does not yield a valid realm, the logon fails.</li> <li>If the directory does not offer a realm contained in the <code>.logdata</code> statement, the logon fails.</li> <li>If the <code>.logdata</code> statement specifies a realm when it is needed, the logon succeeds if it is a valid realm specification.</li> </ul>
<i>tdpid</i>	<p>Required. The <code>tdpid</code> identifies the Teradata Vantage system, Unity server, or host group to which the logon, if successful, connects.</p>

Syntax Element	Description
, ,	<p>If the logon specifies an account, and the directory username and directory password appear in the .logdata statement, the , , must precede the account specification, with these exceptions:</p> <ul style="list-style-type: none"> <li>• If the user credentials appear in the .logon statement, only a single comma is required.</li> <li>• If the .logon does not specify an account, no commas are required.</li> </ul>
"account"	<p>Optional. The account string must be enclosed in double quotation marks. For information on accounts, see <i>Teradata Vantage™ - Database Administration</i>, B093-1093.</p>

## Common Errors with LDAP UPN Logons

Example of Format Error	Reason not Allowed
diruser@@dirpassword authcid=diruser	Duplicate user information in logon string.
diruser@@dirpassword password=dirpassword	Duplicate password information
user=teradatauser diruser@@dirpassword	UPN token must be the first token

## Logging on Using Sign-on As

In a sign-on as logon, the user is authenticated by an external agent that does not also provide authorization. There are two variations of sign-on as logons, based on how the mechanism authorizes database privileges.

- [Using Sign-on As with Teradata Authorization](#)
- [Using Sign-on As with Directory Authorization](#)

## Using Sign-on As with Teradata Authorization

After the external agent authenticates the user, it passes the external user name to the database for authorization, based on the access privileges available to the matching database username.

- Enable external authentication in the database. See [About External Authentication Controls](#).
- At logon, the user must specify a mechanism that corresponds to the agent that does the authentication, from among the following mechanisms:
  - KRB5
  - SPNEGO (not available for ODBC-based applications)
  - LDAP

**Note:**

Sign-on As using Kerberos authentication (KRB5 or SPNEGO mechanism) is usable only from Windows clients.

- Set the AuthorizationSupported property for the authenticating mechanism to no.

**Note:**

This setting ignores any directory mappings that may exist for the user.

- The logon username must match a Teradata Vantage username that has WITH NULL PASSWORD privileges. See [About External Authentication Requirements](#).

## Using Sign-on As with Directory Authorization

Once the external agent has authenticated the user, it passes the username to the directory for authorization of user access privileges, based on mappings to the matching directory user.

- Enable external authentication in the database. See [About External Authentication Controls](#).
- At logon, the user must specify the authenticating mechanism from among the following:
  - KRB5
  - SPNEGO (not available for ODBC-based applications)

**Note:**

Sign-On As using Kerberos authentication (KRB5 or SPNEGO mechanism) is usable only from Windows clients.

For a description of logons where LDAP does both authentication and authorization, see [Logging on Using LDAP Authentication and Authorization](#).

- Configure the authentication mechanism:
  - Set the AuthorizationSupported property for the authenticating mechanism to yes. The KRB5 and SPNEGO mechanisms set AuthorizationSupported to no by default.
  - The mechanism must contain the LDAP properties and values shown in [Option 3: Non-LDAP External Authentication with Directory Authorization](#).
- The logon username must match a username in the authorizing directory, and the matching directory user must be mapped to one or more Teradata Vantage objects, as shown in [Provisioning Directory Users with Teradata Schema Extensions](#) or [Using Native Directory Schema to Provision Directory Users](#).

## Sign-on As Logon Format Examples

The examples that follow are from a BTEQ script. Interactive logons prompt the user to enter logon information. For more detailed information on logon for a specific client application, see the user's guide for that application.

### Example: Sign-on As Using the .logdata Statement

```
.logmech mech_name
.logdata user_credentials [authorization_qualifier]
.logon tdpid/[,,“account”]
```

#### Note:

The user credentials must be specified in the .logdata statement if there is an authorization qualifier, or if required for the user credential format.

### Example: Sign-on As Using the .logon Statement

```
.logmech mech_name
.logon tdpid/user_credentials[,“account”]
```

## Explanation of Sign-on As Examples

Syntax Element	Description
<i>mech_name</i>	Specifies the authentication mechanism. For Teradata authorization: <ul style="list-style-type: none"> <li>• KRB5</li> <li>• SPNEGO</li> <li>• LDAP</li> </ul> For directory authorization: <ul style="list-style-type: none"> <li>• KRB5</li> <li>• SPNEGO</li> </ul>
<i>user_credentials</i>	Specifies the username and password for the logon, and must conform to the following rules: <ul style="list-style-type: none"> <li>• For Teradata authorization, the username is a network or directory username for which there is a matching database user.</li> <li>• For directory authorization, the username is a network username for which there is a matching directory user.</li> <li>• The password is always a network password.</li> </ul>

Syntax Element	Description
	<p>Valid formats for user credentials:</p> <p>In the .logdata statement for KRB5 and SPNEGO:  <code>diruser@@dirpassword</code></p> <p><b>Note:</b></p> <p>Sign-On As using Kerberos authentication (SPNEGO mechanism) is usable only from Windows clients.</p> <p>In the .logdata statement for LDAP:</p> <ul style="list-style-type: none"> <li>• <code>authcid= diruser password= dirpassword</code></li> <li>• <code>diruser@@dirpassword</code></li> <li>• <code>diruser password= dirpassword</code></li> </ul> <p>In a .logon statement for KRB 5 and SPNEGO:  <code>domain_username, domain_password</code></p> <p>In a .logon statement for LDAP:  <code>dir_username, dirpassword</code></p> <p><b>Ensuring Correct Interpretation of UPNs</b></p> <p>For the logon <code>diruser, dirpassword</code>, the <code>user</code> specification can be “a@b” or a/b” or “a\b”. Set <code>LdapCredentialsUPN</code> to interpret the user specification. See <a href="#">LdapCredentialsUPN</a>.</p> <ul style="list-style-type: none"> <li>• If the <code>LdapCredentialsUPN</code> property is absent or set to yes (the default), the system treats the user specification as a UPN, which must conform to the rules of IETF 1964.</li> </ul> <p><b>Note:</b></p> <p>When <code>LdapCredentialsUPN</code> is set to yes, the UPN must appear in the logon as: “a\@b” or “a\b” or “a\b”, where the added backslash (\) character informs the system how to handle the following character.</p> <ul style="list-style-type: none"> <li>• If the <code>CredentialsUPN</code> property is set to no, the system disregards the special characters and considers the user specification to be an Authcid.</li> </ul>
<code>authorization_qualifier</code>	<p>Required if the user is authorized by the directory (<code>AuthorizationSupported=yes</code>) and one or more of the following is true:</p> <ul style="list-style-type: none"> <li>• The directory user is mapped to multiple user or profile objects</li> <li>• LDAP is set to use SASL/DIGEST-MD5 binding (the default), the directory offers more than one realm, and the value of the <code>LdapServerRealm</code> property is to "" (the default).</li> </ul> <p><b>Note:</b></p> <p>The DIGEST-MD5 authentication protocol used by LDAP is deprecated. Teradata strongly recommends you use simple binding with TLS protection, and stop using DIGEST-MD5.</p> <p><b>Directory user mapped to multiple database users:</b></p> <p>If the directory user is mapped to more than one database user, specify the user with the database privileges needed for the session in the form:</p> <p><code>user= database_username</code></p>



Syntax Element	Description
	<p><b>Note:</b> The database username can be either an individual database user or EXTUSER.</p> <p><b>Directory user mapped to multiple profiles:</b></p> <ul style="list-style-type: none"> <li>• If a directory user is mapped to more than one profile, specify <code>profile=profile_name</code> in the <code>.logdata</code> statement to identify the session profile.</li> <li>• If the directory user is mapped to one or more database users and also to a profile, the session defers to the separately mapped profile instead of the profile belonging to the mapped database user.</li> </ul> <p><b>Directory offers multiple realms (LDAP authentication only):</b> Specify the realm as it appears in the directory, normally the fully qualified DNS name of the directory, for example: <code>realm=directory_FQDNSName</code> The system processes realm information as follows:</p> <ul style="list-style-type: none"> <li>• If the logon does not specify a realm, and the <code>LdapServerRealm</code> property value does not yield a valid realm, the logon fails.</li> <li>• If the directory does not offer a realm contained in the <code>.logdata</code> statement, the logon fails.</li> <li>• If the <code>.logdata</code> statement specifies a realm when it is needed, the logon succeeds if it is a valid realm specification.</li> </ul>
<code>tdpid</code>	Required. The <code>tdpid</code> identifies the Teradata Vantage system, Unity server, or host group to which the logon, if successful, connects.
<code>, ,</code>	<p>If the logon specifies an account, and the directory username and directory password appear in the <code>.logdata</code> statement, the <code>, ,</code> must precede the account specification, with these exceptions:</p> <ul style="list-style-type: none"> <li>• If the user credentials appear in the <code>.logon</code> statement, only a single comma is required.</li> <li>• If the <code>.logon</code> does not specify an account, no commas are required.</li> </ul>
<code>"account"</code>	Optional. The account string specification must be enclosed in double quotation marks. For information on accounts, see <i>Teradata Vantage™ - Database Administration</i> , B035-1093.

## Logging on Using Single Sign-on with Kerberos

Single sign-on allows users that are logged on to the Kerberos realm to subsequently logon to Teradata Vantage without submitting a Teradata Vantage username and password.

There are two variations of single sign-on, which are based on the authorization method.

- [Using Single Sign-on with Teradata Authorization](#)
- [Using Single Sign-on with Directory Authorization](#)

---

**Note:**

If a logon to Teradata Vantage does not include a username and password, the system assumes it is a single sign-on, and defaults to the KRB5 mechanism.

---

**.NET Enabled Clients**

References to .NET Data Provider for Teradata (or simply .NET) are to the current release of the .NET Data Provider for Teradata, unless otherwise noted.

If TTU 16.20 Feature Update 1 or higher .NET clients use SPNEGO from Linux and Mac OS to access a Teradata system, the Teradata system must be Teradata Database Release 16.10 or above or a Teradata Vantage system.

## Using Single Sign-on with Teradata Authorization

After the external agent authenticates the user, it passes the network user name to the database, which authorizes the access privileges available to the matching database username.

**Requirements**

- If the logon is from a .NET enabled client, users cannot use the default KRB5 authentication mechanism, and instead must specify the SPNEGO mechanism.
- Set the AuthorizationSupported property for the authentication mechanism to no (the default) on all database nodes.
- The logon username (a domain username) must match a Teradata Vantage username.

## Using Single Sign-on with Directory Authorization

After the external agent authenticates the user, it passes the username to the directory for authorization of user privileges, based on mappings to the matching directory username.

Observe the following when using Single Sign-on logons with directory authorization:

- If the logon is from a .NET enabled client, users cannot use the default KRB5 authentication mechanism, and instead must specify the SPNEGO mechanism.
- The authentication mechanism must be configured as follows:
  - The AuthorizationSupported property for the authenticating mechanism must be set to yes. The KRB5 and SPNEGO mechanisms are set to no by default and must be reconfigured to yes to support directory authorization.
  - All supporting mechanisms contain required LDAP properties and values, which you must configure. See [Option 3: Non-LDAP External Authentication with Directory Authorization](#).
- The logon username must match a username in the authorizing directory and the matching directory user must be mapped to one or more Teradata Vantage objects. See [Provisioning Directory Users with Teradata Schema Extensions](#) and [Using Native Directory Schema to Provision Directory Users](#).

## Single Sign-on Examples

### Example: Single Sign-on with Teradata Authorization

```
.logmech mech_name.logon tdpid/[,,"account"]
```

### Example: Single Sign-on with Directory Authorization

```
.logmech mech_name
.logdata [authorization_qualifier]
.logon tdpid/[,,"account"]
```

## Explanation of Single Sign-on Examples

The following explains logon terms used in the Single Sign-on example.

Syntax Element	Description
<i>mech_name</i>	<p>Required only if KRB5 is not used. Specify the SPNEGO mechanism for Kerberos authentication from a .NET client.</p> <p><b>Note:</b></p> <p>If no mechanism and no user credentials are specified, the system assumes a single sign-on and authenticates with Kerberos.</p>
<i>authorization_qualifier</i>	<p>Required if users are authorized by a directory, that is, the KRB5 mechanism has AuthorizationSupported=yes:</p> <ul style="list-style-type: none"> <li>The directory user is mapped to multiple user or profile objects (for all mechanisms).</li> <li>LDAP uses SASL/DIGEST-MD5 binding (the default), the directory offers more than one realm, and the value of the LdapServerRealm property is to the default "" (for the LDAP mechanism only).</li> </ul> <p><b>Note:</b></p> <p>The DIGEST-MD5 authentication protocol used by LDAP is deprecated. Teradata strongly recommends you use simple binding with TLS protection, and stop using DIGEST-MD5.</p> <p><b>If the matching directory user is mapped to multiple database users:</b></p> <p>If the directory user is mapped to more than one database user, specify the user with the database privileges needed for the session in the form:</p> <p>user= database_username</p> <p><b>Note:</b></p> <p>The database username can be either a database user or EXTUSER.</p> <p><b>If the matching directory user is mapped to multiple profiles:</b></p>

Syntax Element	Description
	<ul style="list-style-type: none"> <li>If a directory user is mapped to multiple profiles, specify <code>profile=profile_name</code> to identify the session profile.</li> <li>If the directory user is mapped to one or more database users, and also to a profile, the session defers to the separately mapped profile instead of the profile belonging to the mapped database user.</li> </ul> <p><b>If the directory offers multiple realms:</b> Specify the realm as it appears in the directory, normally the fully qualified DNS name of the directory, for example: <code>realm=directory_FQDNSName</code> The system processes realm information as follows:</p>
<code>tdpid</code>	Required. The <code>tdpid</code> identifies the Teradata Vantage system, Unity server, or host group to which the logon, if successful, connects.
<code>, ,</code>	User credentials are not required for single sign-on. The <code>, ,</code> is required as a place holder for the user credentials only if an account string is specified. Otherwise commas are not needed.
<code>"account"</code>	Optional. The account string must be enclosed in double quotation marks. For information on accounts, see <i>Teradata Vantage™ - Database Administration</i> , B035-1093.

## Logging on Using Teradata Negotiating (TDNEGO)

The TDGSS model allows a user running a client application to pick from a set of TDGSS mechanisms to establish a security context between the client and the Server Gateway. Teradata Negotiating (TDNEGO) automatically negotiates the sign-on mechanism so the user or client application does not have to specify it.

### TDNEGO Benefits

TDNEGO negotiates the appropriate security mechanism for the user if the user is not aware of which mechanisms are available on the server or is not aware of restrictions on mechanism usage.

### TDNEGO Negotiation

TDNEGO is the only negotiating mechanism that Teradata Vantage supports. You can configure TDNEGO to choose from one or more logon authentication mechanisms for client access.

#### Note:

SPNEGO can be negotiated by TDNEGO for clients using Kerberos with the .NET framework.

TDGSS also allows you to set a non-negotiated mechanism, which is TD2 by default. The non-negotiated mechanism is tried if protocol negotiations fail. See information about the [DefaultMechanism](#) setting in the `TdgssUserConfigFile.xml`.

If the database server and the client cannot find a common mechanism, the logon request is denied, and the logon fails.

In most cases, negotiations are initiated by the client logon request. If the client does not specify an authentication protocol, TDNEGO uses the default mechanism of the database server.

If you need to modify the TDNEGO configuration, the best practice is to modify the configuration on the database server. If you must modify the TDNEGO configuration on the client, install the Teradata GSS Administrative Package on the client. For more information, see [Teradata GSS Administrative Package](#).

## TDNEGO Use Cases

TDNEGO automatically determines the correct mechanism to use in the following situations:

- If a user is not sure which security mechanism is required by mechanism policy in the external directory the user can select TDNEGO.
- If a user has different passwords for Kerberos and LDAP and is not sure which password goes with which mechanism the user can select TDNEGO.
- If a user wants to do a single sign-on (SSO) but does not know which mechanisms support SSO the user can select TDNEGO without providing credentials and TDNEGO automatically uses the appropriate mechanism.
- If a client application does not support mechanism selection, the site can set the default negotiating mechanism to TDNEGO at either the client or at the server.

## TDNEGO Supported Mechanisms

TDNEGO supports the following mechanisms.

- TD2
- LDAP
- KRB5
- JWT (JSON Web Token)
- SPNEGO (provides KRB5 for .NET clients)

## TDNEGO Supported Clients

TDNEGO supports the following clients.

- CLI
- ODBC
- JDBC
- .NET

## Unity Support for TDNEGO

Unity does not support TDNEGO. Teradata recommends disabling TDNEGO on Unity servers. See [Disabling TDNEGO](#).

## TDNEGO Usage Constraints

TDNEGO results in a mechanism other than TDNEGO being used, so the following applies:

- A user must not be restricted to using only TDNEGO in the network security policy, because TDNEGO always selects another mechanism; the user must be allowed to use the selected mechanism, or else the logon is not allowed.
- It is allowed, but not required, to add TDNEGO to the list of mechanisms a user is allowed to use; however, is recommended that TDNEGO not be specified as an allowed mechanism in the directory.
- Concerning QOP and enforced network security policy, note that QOP is not supported by all mechanisms. TDNEGO is one of the mechanisms that does not support QOP. However, any QOP restrictions in the security policy for the mechanism selected by TDNEGO do apply. For example, if TDNEGO selects TD2, and the security policy requires the user to use high level encryption, then that will be enforced.

## TDNEGO Compatibility with TTU Client Tools

You must have TDNEGO installed on both the Teradata Tools and Utilities (TTU) client and the database server:

- Both the client and database must be release 15.10 or higher. All versions of the Teradata Vantage Advanced SQL Engine support TDNEGO.
- Clients using the .NET framework must be at TTU release 16.0 or higher to use SPNEGO, which provides Kerberos authentication to .NET clients.
  - For .NET client access to Teradata Database 15.10, TDNEGO must be manually configured to accept SPNEGO as a negotiated authentication mechanism. For more information, see [Changing the Configuration on Teradata Vantage Nodes](#) and [SPNEGO Mechanism Offered by TDNEGO on Teradata Database 15.10 for TTU 16.0 .NET Clients](#).

## Configuring TDNEGO Properties

TDNEGO is enabled by default on both the client and server and can be used by the client explicitly requesting it.

TDNEGO is not set as a default negotiating mechanism.

To set TDNEGO as the default negotiating mechanism modify TdgssUserConfigFile.xml. Do the following:

- Review [TDNEGO Mechanism Properties](#).
- Perform the steps in [Changing the Configuration on Teradata Vantage Nodes](#).

## TDNEGO Mechanism Properties

The following TDGSS mechanism properties are not available and not used with TDNEGO.

- MechanismIgnoresQop
- VerifyDHKey
- DHKeyP and DHKeyG
- DHKeyP2048 and DHKeyG2048

The following TDGSS mechanism properties are set as shown in the table for the TDNEGO mechanism and may or may not be modified.

Do Not Modify These Properties	These Properties May Be Modified
<ul style="list-style-type: none"> <li>• AuthenticationSupported="yes"</li> <li>• AuthorizationSupported="yes"</li> <li>• SingleSignOnSupported="yes"</li> <li>• GenerateCredentialsFromLogon="yes"</li> <li>• NegotiationSupported="yes"</li> <li>• MutualAuthentication="yes"</li> <li>• ReplayDetection="yes"</li> <li>• OutOfSequenceDetection="yes"</li> <li>• ConfidentialityDesired="yes"</li> <li>• IntegrityDesired="yes"</li> </ul>	<ul style="list-style-type: none"> <li>• MechanismEnabled="yes"</li> <li>• DefaultMechanism="no"</li> <li>• DefaultNegotiatingMechanism="no"</li> <li>• MechanismRank="10"</li> </ul>

TDNEGO requires a complete LdapConfig section at the server, if mechanism policy is used as a selection criterion. If you are not using mechanism policy to restrict the mechanisms available for negotiation, the LdapConfig settings are not required.

The NegotiationSupported property is boolean and signifies that a mechanism is actually a pseudo mechanism that negotiates between the client and the server to find an appropriate mechanism to use. All mechanisms that fit this description will have NegotiationSupported set to yes; TDNEGO has this property set to yes. If the mechanism has NegotiationSupported set to yes it must also have at least one NegotiatedMechanism ObjectID set, otherwise an error is reported by the tdgssconfig executable. The NegotiationSupported property should not be modified.

The DefaultNegotiatingMechanism mechanism property is boolean and is similar to the DefaultMechanism property. DefaultNegotiatingMechanism is used to specify a default negotiating mechanism from among those mechanisms whose NegotiationSupported mechanism property is yes. DefaultNegotiatingMechanism is used to determine the default mechanism when both the client and the server support mechanism negotiation. The default setting for DefaultNegotiatingMechanism is no.

If both client and server support negotiation, but no common negotiating mechanism exists between them, the existing DefaultMechanism property is used to select a default mechanism.

## Changing the Configuration on Teradata Vantage Nodes

1. On the Teradata Vantage node with the lowest ID number, go to the directory that contains TdgssUserConfigFile.xml.

```
cd /opt/teradata/tdat/tdgss/site
```

2. Make a backup copy of TdgssUserConfigFile.xml and save it according to your site standard backup procedures.
3. Edit the TdgssUserConfigFile.xml using a text editor, such as vi.
4. Uncomment the TDNEGO mechanism section, if not already done, and edit the properties as desired.

---

### Note:

Do not set the DefaultMechanism property to “yes” in the TDNEGO section. If DefaultMechanism is set to yes, older clients requesting the default mechanism will get a failed login.

---

- If you are performing an upgrade, rather than a new installation, the TdgssUserConfigFile.xml in /opt/teradata/tdat/tdgss/site directory is preserved so the TDNEGO mechanism may not be present in the file. Copy the TDNEGO mechanism from the TdgssUserConfigFile.xml in the /opt/teradata/tdat/tdgss/<version>/etc directory. Paste the mechanism into /opt/teradata/tdat/tdgss/site/TdgssUserConfigFile.xml file. Edit it as needed.
  - (Optional) If there are TTU 16.0 or higher .NET clients that use SPNEGO as a TDNEGO negotiated mechanism to access Teradata Database 15.10, see the following for configuration details: [SPNEGO Mechanism Offered by TDNEGO on Teradata Database 15.10 for TTU 16.0 .NET Clients](#).
  - (Optional) Edit the DefaultNegotiatingMechanism. For example, to set TDNEGO as the DefaultNegotiatingMechanism update the property to “yes” in the TDNEGO mechanism.
5. Run the run\_tdgssconfig utility to update the TDGSSCONFIG GDO.

```
/opt/teradata/tdgss/bin/run_tdgssconfig
```

6. Run tpareset to activate the changes to the TDGSS configuration.

```
tpareset -f “use updated TDGSSCONFIG GDO”
```

---

### Note:

You only need to perform this procedure once from any node in a Teradata Database or Teradata Vantage system. Running the run\_tdgssconfig tool distributes the change to all database nodes.

---



## SPNEGO Mechanism Offered by TDNEGO on Teradata Database 15.10 for TTU 16.0 .NET Clients

TDNEGO offers TD2 and LDAP as negotiated mechanisms for TTU 16.0 or higher .NET clients to access Teradata Database 15.10 or higher and Teradata Vantage. TDNEGO offers SPNEGO as a negotiated mechanism for TTU 16.0 or higher .NET clients to access Teradata Database 16.0 or higher and Teradata Vantage.

TTU 16.0 or higher .NET clients that want to use SPNEGO as a TDNEGO negotiated mechanism to access Teradata Database 15.10 must add SPNEGO as a negotiated mechanism to the TDNEGO mechanism on the Teradata Database 15.10 server.

---

### Note:

Teradata Database 16.0 or higher and Teradata Vantage are already configured to offer SPNEGO to .NET clients. CLI, ODBC, and JDBC do not support SPNEGO, so the only time the following configuration needs to be done is with a Teradata Database 15.10 server and TTU 16.0 .NET clients that want to use SPNEGO.

---

1. On the Teradata Database 15.10 server, edit `TdgssUserConfigFile.xml`, add the highlighted line below to the TDNEGO mechanism, and uncomment the TDNEGO section (if not already done):

```
<!-- TDNEGO: Teradata Negotiated Method -->

    <!-- To modify TDNEGO configuration, uncomment this section and edit
    <Mechanism Name="TDNEGO"
    <MechanismProperties

        MechanismEnabled="yes"
        DefaultMechanism="no"
        DefaultNegotiatingMechanism="no"
        MechanismRank="10"

    />

    <!-- Mechanisms offered for negotiation: KRB5, SPNEGO, ldap, TD2 -->
    <NegotiatedMechanism ObjectId="1.2.840.113554.1.2.2" Enable="yes"/>
    <NegotiatedMechanism ObjectId="1.3.6.1.5.5.2" Enable="yes"/>
    <NegotiatedMechanism ObjectId="1.3.6.1.4.1.191.1.1012.1.20"
Enable="yes"/>
    <NegotiatedMechanism ObjectId="1.3.6.1.4.1.191.1.1012.1.1.9"
Enable="yes"/>
</Mechanism>
```

```
(end of commented out section) -->
```

2. Complete the configuration steps (run\_tdgssconfig and tpareset) shown in [Changing the Configuration on Teradata Vantage Nodes](#).

## Removing a NegotiatedMechanism from TDNEGO

To remove a NegotiatedMechanism:

1. Edit TdgssUserConfigFile.xml and change Enable="yes" to Enable="no". For example:

```
<NegotiatedMechanism ObjectId="1.3.6.1.4.1.191.1.1012.1.1.9" Enable="no"/>
```

### Note:

Do not remove all of the NegotiatedMechanisms from TDNEGO. TDNEGO has NegotiationSupported set to yes, so it must have at least one NegotiatedMechanism ObjectId set; otherwise, an error is reported by the run\_tdgssconfig executable.

If you want to disable TDNEGO, see [Disabling TDNEGO](#).

2. Run the run\_tdgssconfig command to enable the change.
3. Run tpareset (on Teradata Vantage SQL Engine nodes only).

For specific steps on editing the configuration, see: [Changing the Configuration on Teradata Vantage Nodes](#).

## Disabling TDNEGO

To disable TDNEGO:

1. Edit TdgssUserConfigFile.xml and set the following TDNEGO mechanism properties to no.
  - MechanismEnabled
  - DefaultMechanism
  - DefaultNegotiatingMechanism
2. Run the run\_tdgssconfig command to enable the change.
3. Run tpareset (on Teradata Vantage SQL Engine nodes only).

For specific steps on editing the configuration, see: [Changing the Configuration on Teradata Vantage Nodes](#).

## Related Information

For more information on modifying the TDGSS configuration see [Changing the TDGSS Configuration](#). For details about TDNEGO related properties and other properties in the configuration file see [TDGSS Configuration Files, Valid Settings, and Editing Guidelines](#).

## TDNEGO Logging

To view the mechanisms that were considered by TDNEGO turn on TDGSS mechanism negotiation logging at the gateway. TDNEGO logs to the gateway log.

---

### Note:

A large number of log entries may be generated, so be aware of the log file size.

1. To turn logging on or off use the following toggle.

```
gtwcontrol -N
```

---

For more information about the gateway log file and gtwcontrol see Utilities.

## TDNEGO Support Considerations

Teradata recommends making TDNEGO configuration changes on the Teradata Vantage server, whenever possible, to avoid numerous client reconfigurations.

## Using Teradata Wallet to Store and Retrieve Logon Elements

Users can optionally store usernames and passwords on a Teradata Vantage client computer or application server running Teradata Tools and Utilities 14.0 and up, using the included Teradata Wallet software, and then retrieve the needed data when logging on to any compatible Vantage system.

## Benefits

Teradata Wallet storage is especially beneficial for easy retrieval of passwords on application servers or other shared computers that host multiple users and connect to multiple databases. Users wanting to use retrieved data in logon strings must have a personal Teradata Wallet instance on each computer through which they access Teradata Vantage.

Passwords and other data are securely stored in protected form. See [Encryption](#).

Each user can store data only in their own wallet, which is not accessible by other users. The system retrieves data only from the wallet belonging to the logged on user.

Teradata Wallet substitution strings are accepted by both the .logon and .logdata statements in the logon string, and by the corresponding logon functions in ODBC (14.10 and up only) and CLI applications and scripts.

## Use Cases

Users running scripted applications can embed password retrieval syntax into scripts instead of compromising security by including a password.

Users accessing multiple Teradata Vantage systems can automatically retrieve the correct username and password for a system (tdpid) instead of having to remember the information or look it up.

## Restrictions

- On Windows, using the Credential Manager to modify Teradata Wallet string values is not supported because it corrupts the values and they must be deleted and re-added using the tdwallet command-line tool.
- When multiple users log on to the database from a single computer, each user must be uniquely identified on the computer so retrieval of wallet data is user-specific and private.
- Teradata authenticated users (TD2 mechanism) must reset passwords stored in Teradata Wallet to conform to any changes required by database password controls, for example, PasswordExpire. See [Working with Password Controls](#).

## Prerequisites

The topics that follow, showing how to use Teradata Wallet to store and retrieve logon string information, assume that Teradata Wallet is installed and configured on a Teradata Vantage client. For detailed information on Teradata Wallet installation and setup options, see the *Teradata Tools and Utilities Installation Guide* for the client operating system.

## Storing Logon Information in Teradata Wallet

### Storing a Password Using an Alias Name

Users can store a Teradata Vantage password using an alias. For example:

1. Log on to a Vantage client system and obtain a command prompt.
2. Run the tdwallet tool on the client to add a Vantage user password to the user wallet.

For example, on a UNIX client:

```
$ tdwallet add password_alias
```

where *password\_alias* is a character string. For example:

```
PWalias1
```

3. At the system prompt, enter the Vantage password that corresponds to the password alias, for example:

```
MyTDPASSWORD
```

For detailed information on Teradata Wallet installation, see the *Teradata Tools and Utilities Installation Guide* for the client operating system.

## Storing Logon Information by System Tdpid

Users who need to log on to multiple databases, Unity servers, or host groups using a different password for each can store and retrieve the passwords by tdpid. For convenience, you can also store the username for each Tdpid.

In subsequent logons, the user enters the Tdpid and the required substitution syntax to call the username and password for that tdpid. For example:

1. Log on to a Teradata Vantage client system and obtain a command prompt.
2. Run the tdwallet tool on the client to store usernames and passwords for the TDSys1 and TDSys2 systems. The following example is for a Windows client:
  - a. Enter the command to add the username and password for the first system, TDSys1:

```
C:\Documents and Settings\joe>tdwallet add username_for_TDSys1
add password_for_TDSys1
```

### Note:

If a password or passphrase is not already established for accessing the user wallet, the Teradata Wallet software first prompts for the required access password or passphrase. At the prompt, enter the wallet access password or passphrase.

```
NOTICE: No password has been established for this wallet.
Enter desired wallet password: password (not displayed)
Reenter desired wallet password: password (not displayed)
Wallet password established. Remember your password.
```

- b. The system prompts for the values of username\_for\_TDSys1 and password\_for\_TDSys1. At the prompts, enter the username and password for the TDSys1 system, for example:

```
Enter desired value for the item named "username_for_TDSys1":
```

- c. Add the username and password for the second system, TDSys2:

```
C:\Documents and Settings\joe>tdwallet add username_for_TDSys2
add password_for_TDSys2
```

- d. At the prompts, enter the username and password for the TDSys2 system.
- e. Add the string that links stored usernames and passwords for the two systems to the authentication mechanism used to access the systems, for example:

```
C:\Documents and Settings\joe>tdwallet add com.teradata.TD2
```

**Note:**

Each set of usernames and passwords can be associated with only one authentication mechanism.

- f. At the prompt, enter the substitution string syntax to link the tdpid specified in a logon string, for example TDSys1, to the password stored for that tdpid:

```
$tdwallet(password_for_$(tdpid))
```

## Retrieving Logon Information from Teradata Wallet Using BTEQ

Using BTEQ commands, you can retrieve logon information stored in Teradata Wallet.

### Retrieving a Password Using a Password Alias

You can retrieve the password corresponding to the *password\_alias* specified in the logon string.

```
.logon tdpid/username,$tdwallet(password_alias)[,"account"]
```

**Note:**

This logon method requires completion of the setup shown in [Storing a Password Using an Alias Name](#).

### Retrieving a Password for a Specified Tdpid

You can retrieve the password corresponding to the *tdpid* specified in the logon string.

```
.logon tdpid/username,$tdwallet[,"account"]
```

**Note:**

This logon method requires completion of the setup shown in [Storing Logon Information by System Tdpid](#).

## Retrieving a Username and Password for a Specified Tdpid

Teradata Wallet can also provide both the username and password for a *tdpid* specified in the logon string.

```
.logon tdpid/$tdwallet(username_for_$(tdpid)),$tdwallet(password_for_$(tdpid))[,,"account"]
```

**Note:**

This method requires completion of the setup shown in [Storing Logon Information by System Tdpid](#), including optional setup of the username.

## Retrieving a Password from Teradata Wallet Using ODBC Driver for Teradata

You can specify a Teradata Wallet string in an ODBC SQLConnect, SQLDriver or BROWSEConnect function, when using ODBC Driver for Teradata 14.0 and up. For example:

```
SQLConnect(hdbc, "mydsn", SQL_NTS, "myuid", SQL_NTS, "$tdwallet(abcd)", SQL_NTS);
SQLDriverConnect(hdbc, NULL, "DSN=mydsn;UID=myid;PWD=$tdwallet(abcd);", SQL_NTS,
szConnStrout, cbConnStrOutMax, &cbConnectStrOutLen, NULL);
```

On Windows clients, you can also specify a single connect string in the Teradata Wallet String field of the ODBC Driver Setup dialog box, according to the following rules:

- Enter only the wallet string. The \$tdwallet() token is not required.
- Entering a wallet string precludes the specification of a password in the adjacent Password field.

On Linux/UNIX clients you can enter the entire string and token in either the:

- odbc.ini, for example: \$tdwallet(*password\_alias*)
- connection string, for example:

```
DRIVER={Teradata}; DBCNAME=platinum;
AUTHENTICATION=LDAP; AUTHENTICATIONPARAMETER=authcid=$tdwallet(odbc_krb1_ad)
password=$tdwallet(odbc_krb1_ad_pwd); "
```

## Working with Logon Variations by Application

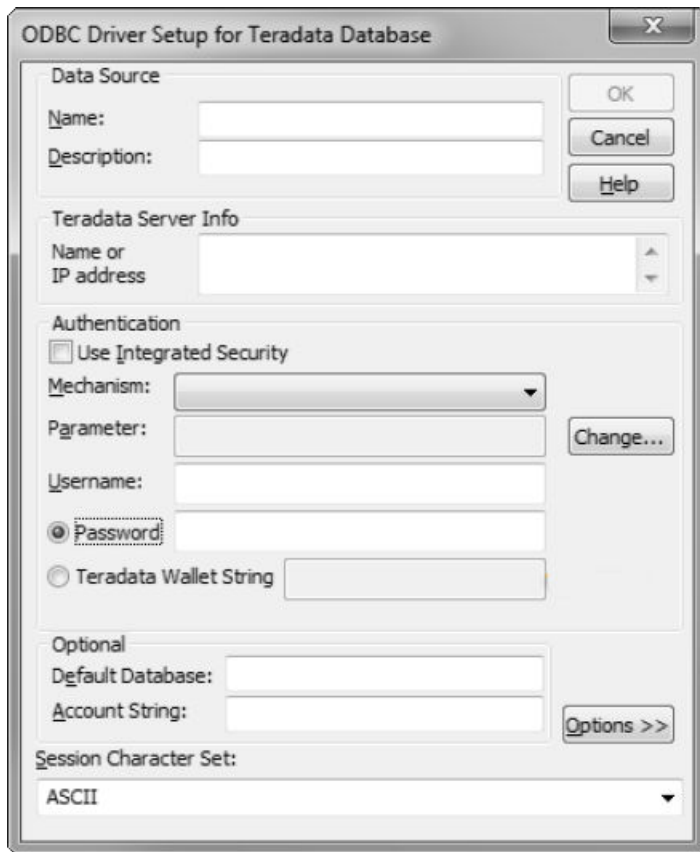
The method for specifying logon parameters varies among applications.

- For Teradata Tools and Utilities applications:
  - Interactive applications such as BTEQ allow specification of logon parameters on the command line. See [About Network Logons](#).
  - Scripted applications such as Teradata Parallel Transporter allow specification of logon parameters as operator attributes or preset variables.
  - ODBC and JDBC based applications allow setting of some or all logon parameters as part of initial configuration of the associated driver.
  - GUI applications allow specification or change of logon parameters as part of connecting to the database.
- For third-party applications:
  - Set logon parameters in the application (varies by application).
  - Set up the application logon user, and optionally set up trusted sessions to identify and authorize application end users as individuals. See [Working with Middle-Tier Application Users](#).

## Specifying Logon Parameters when Setting Up an Application

Applications provide logon dialog boxes to allow users to specify logon data and a security mechanism. A following is a typical logon dialog box.





The parameter values required by the dialog box are the same as those required when logging on from the command line.

After entering the required information, click the **Mechanism** button and the **Mechanism** dialog box displays as shown below. If you do not specify a mechanism, the system uses the default mechanism.

## Logging On from Mainframe Systems

Mainframe clients do not support network security features, such as encryption or directory management of users. Unless you make modifications to the Teradata Director Program (TDP), logons from mainframe clients can use only the .logon command, and are similar to the logon shown in [Example: TD2 Logon](#).

For information on how to use TDP functions to alter logon security requirements in a mainframe environment, see *Teradata® Director Program Reference*, B035-2416.

## Logging On from Teradata Vantage Nodes

You may need to log on to Teradata Vantage from a database node, for example, when you use a utility to perform internal system maintenance. Although communication among Vantage nodes takes place across the network using TCP/IP, you probably do not need network security functions.

You can log on from a Vantage node, using the logon shown in [Example: TD2 Logon](#).

## Logging On through Unity

Users logging on through Unity must specify the tdpid of the Unity server or host group instead of a Teradata Vantage system tdpid. The authentication mechanism active for the logon must be enabled on the client, the Unity server, and all connected Vantage systems. Otherwise, logon requirements are unaffected by the presence of Unity.

You must complete several setup tasks on each Unity server and on each connected database system before users can log on through Unity. For information about Unity, see *Teradata® Unity™ Installation, Configuration, and Upgrade Guide for Customers*, B035-2523 and *Teradata® Unity™ User Guide*, B035-2520.

## Logging On from .NET Clients

Logons to Teradata Vantage from .NET applications require the user to select from among the following mechanisms:

- TD2
- LDAP
- TDNEGO
- SPNEGO - used to access KRB5 for all non-LDAP external authentication. SPNEGO may need to be enabled, depending on the database and client releases. For more information, see [TDNEGO Compatibility with TTU Client Tools](#).

## Using Operating System Logons

System maintenance and the use of some Vantage utilities may require operating system-level (OS-level) access. Teradata Vantage runs as a system-generated, OS-level user.

For information on how to execute an OS-level logon to run a utility, see *Teradata Vantage™ - Database Utilities*, B035-1102.

For information on the default Teradata Vantage OS-level users, see [Controlling Access to the Operating System](#).

## Using Logon Error Handling Options

You can use the Gateway Control utility -n option to determine whether users receive specific information about logon errors.

When the value is set to:

- no (the default), all logon errors return the default message:

```
The User Id, Password, or Account is invalid
```

- yes, logon errors return a message that describes the specific error encountered, for example:

```
Invalid user password
```

Teradata recommends the default setting (no), instead of enabling failure-specific messages (the yes setting), to prevent unauthorized users from getting hints about a failed attempt to break-in to the system.

To aid administrators in debugging logon failures, the Gateway writes failure-specific error messages to the system log `/var/log/messages`, regardless of how the `-n` option is set.

All TDGSS-related errors also appear in the gateway log: `/var/opt/teradata/tdtemp/gtw`

# Directory Management of Database Users

The following topics outline the decisions and processes necessary to implement directory management of Teradata Vantage users, and provides links to required tasks and major options.

## Directory Database User Implementation Process

1. Evaluate the system for directory management of Teradata Vantage users. See [Evaluating the System for Directory Management of Users](#).

- Make sure your directory is compatible with Vantage.
- Run tests to ensure that the directory properly communicates with the database.

2. Determine the directory authentication/authorization strategy and learn the configuration requirements. See [Working with Directory User Management Options](#).

Enable directory authentication/authorization as shown in the topics about setting up the options that you want to implement.

- For auto provisioning perform the steps in [Option 2: Directory Authentication and Authorization](#) or in [Option 3: Non-LDAP External Authentication with Directory Authorization](#) depending on which matches your site configuration.
- For lightweight LDAP authorizations perform the steps in [Option 4: Lightweight LDAP Authorizations](#).

3. Review directory user characteristics, privileges, and required directory setup tasks. See [About Directory User Characteristics](#).

4. In the database, create profiles and external roles for assignment to directory users. See [Creating Users and Granting Privileges](#).

- For information on creating external roles see [Using Roles for Directory Users](#).
- For information about profiles see the topics beginning with [Working with Database Profiles](#).

---

### Note:

Auto provisioned users must be assigned to a role or profile and not mapped to a database user.

---

5. Provision directory users using either of these procedures.
  - [Provisioning Directory Users with Teradata Schema Extensions](#).
  - [Using Native Directory Schema to Provision Directory Users](#).
6. If they do not already exist in the directory, create database objects for roles and profiles. For auto provisioning create directory roles based on the external roles in the database. Assign directory principals to roles or profiles.

- See the examples in [Mapping Directory Users to Vantage External Roles](#).
- For information about profiles see [About Assigning Profiles to Users](#) and see the example in [Mapping Directory Users to Vantage Profiles](#).

**Note:**

Skip this step if you are using lightweight LDAP authorizations. For lightweight LDAP authorizations you do not need to create database objects for users, roles, and profiles in the directory (in the tdatSystem).

7. Test the setup. See [Testing Directory Authentication and Authorization Setup](#).
8. Evaluate, and if necessary configure, LDAP binding and protection options. See:
  - a. [LDAP Binding Options](#).
  - b. [Using TLS with a Directory Server](#).
9. Evaluate, and if necessary, configure directory search options. See [Optimizing Directory Searches](#).
10. If multiple directory services access Teradata Vantage, evaluate the need to complete special setup procedures. See:
  - [Configuring LDAP for Site-Aware Authentication](#).
  - [Configuring LDAP to Use Multiple Directory Services](#).
11. In a multi-system environment, where users log on through Unity, observe the additional directory configuration requirements needed for Unity. For information about Unity, see *Teradata® Unity™ Installation, Configuration, and Upgrade Guide for Customers*, B035-2523 and *Teradata® Unity™ User Guide*, B035-2520.

**Note:**

Teradata recommends that you implement and test LDAP authentication and authorization of users for individual database systems before attempting to configure it for Unity.

12. Evaluate, and if necessary configure, network security policies in the directory. See [Network Security Policy](#).

## Working with Directory User Management Options

Teradata Vantage provides several alternatives for managing database users in a directory.

- [Option 1: Directory Authentication Only](#)
- [Option 2: Directory Authentication and Authorization](#)
- [Option 3: Non-LDAP External Authentication with Directory Authorization](#)
- [Option 4: Lightweight LDAP Authorizations](#)

## Option 1: Directory Authentication Only

The directory authenticates directory users with a user names that match a Vantage username to log on to the database and inherit the privileges of the matching database user.

This option is useful when:

- Site security policy requires directory authentication
- All or most of the directory users who need to access the database already have matching users defined in the database
- The benefits gained from also authorizing user privileges in the directory is not worth the additional setup effort

## Advantages

You do not need to:

- Install Teradata schema extensions in the directory
- Create new directory objects
- Map directory users to new objects

## Limitations

- Directory users have only the privileges already available to the matching database users.
- User management is not centralized in the directory.

## Setting Up Directory Authentication

1. Verify that the database contains a username that matches the username of each directory user that requires access to the database. Create additional database users where required. See [Creating Users and Granting Privileges](#).
2. Enable external authentication in the database. See [About External Authentication Controls](#).
  - For the Vantage nodes with gateway installed, run:

```
gtwcontrol -a ON
```

- And, on all Vantage nodes, run dbscontrol and enter: m g 26 0

```
dbscontrol m g 26 0
```

3. Grant external authentication privileges to the matching database users. See [About External Authentication Requirements](#).
4. Configure the LDAP mechanism in the TdgssUserConfigFile.xml using the following property values. Run dumpcfg to view the configuration.

- MechanismEnabled = “yes” (the default)
  - AuthorizationSupported = “no”
5. If the properties need to be modified, edit the TdgssUserConfigFile.xml and enable the new configuration on all systems.
    - For database nodes perform the steps in [Making Changes to TdgssUserConfigFile.xml on Database Nodes](#).
    - For Unity servers, see *Teradata® Unity™ Installation, Configuration, and Upgrade Guide for Customers*, B035-2523 and *Teradata® Unity™ User Guide*, B035-2520.
  6. Set the LDAP mechanism as the default on all clients that use LDAP authentication, or instruct users to specify the LDAP mechanism in the logon string.
  7. Use the logon format shown for LDAP authentication. See [Logging on Using Sign-on As](#).

## Option 2: Directory Authentication and Authorization

If directory users do not have matching user names in the database, or if the matching user name does not have all the requisite database privileges, you can authorize privileges in the directory by mapping directory users to database users, and to database external roles and profiles.

- You can map directory users with very limited access needs to the system-generated pseudo-user, EXTUSER, which has database privileges similar to PUBLIC, by default. Directory users mapped to one or more Vantage roles or profiles--and not mapped to a database user--automatically gain EXTUSER privileges in addition to those privileges acquired from the roles/profiles.
- If the directory user requires an individual database account, you can enable the DBSControl AutoProvision parameter and map the directory user to a role or profile. In this case, a database account is automatically provisioned for the user and they are logged on to the database as that user instead of EXTUSER.

For information on EXTUSER, see [Characteristics of Directory Users Mapped to Permanent Database Users](#).

For information on auto provisioned users, see [About Auto Provisioned Directory Users](#).

## Advantages

You have more flexibility in assigning user privileges if you map directory users to Vantage objects than if you use the [Option 1: Directory Authentication Only](#) strategy.

- Directory users inherit all the privileges available to the database objects to which they are mapped.
- You can map multiple directory user names to a single database user, external role, or profile.
- You can map a single directory user to multiple database users, external roles, and profiles.

## Limitations

- Directory authorization of privileges requires a complex and time-consuming setup process.
- If you decide to use Teradata directory schema extensions for directory user provisioning, and you install them in Active Directory, ADAM, or AD LDS, you cannot remove the extensions or any related schema objects you create (although you can change object mappings). Other directory types allow you to remove schema objects.

## Setting Up Directory Authentication and Authorization

1. Enable external authentication in the database. See [About External Authentication Controls](#).

- For the Vantage nodes with gateway installed, run:

```
gtwcontrol -a ON
```

- And, on all Vantage nodes, run dbscontrol and enter m g 26 0

```
dbscontrol m g 26 0
```

2. Grant external authentication privileges to the matching database users. See [About External Authentication Requirements](#).
3. Verify that the TdgssUserConfigFile.xml contains the following settings. Run dumpcfg to view the TDGSS configuration.

- MechanismEnabled = "yes" (on both the server and clients)
- AuthorizationSupported = "yes" (on all database nodes)

If AuthorizationSupported is not set to yes, the directory user can only have the database privileges available to the matching database username.

4. (Optional) To use auto provisioning enable the DBSControl AutoProvision parameter.

```
dbscontrol m g 81 T
```

5. Configure the required LDAP mechanism properties in the TdgssUserConfigFile.xml. See [Directory Identification and Search Properties](#):

- LdapServerName
- LdapServerRealm (on some systems)

6. Complete edits for the TdgssUserConfigFile.xml and enable them on the systems. The changes are made in the TDGSS site directory. See [Changing the TDGSS Configuration](#). For database nodes, perform the steps in [Making Changes to TdgssUserConfigFile.xml on Database Nodes](#).

7. To configure Unity servers, see *Teradata® Unity™ Installation, Configuration, and Upgrade Guide for Customers*, B035-2523.



8. Set the LDAP mechanism as the default on all affected clients, or instruct users to specify the LDAP mechanism in the logon string. See the appropriate TTU client guide for how to configure a default mechanism for your client.
9. Use the logon format for LDAP authentication. See [Logging on Using LDAP Authentication and Authorization](#).

## Option 3: Non-LDAP External Authentication with Directory Authorization

This option allows you to use Kerberos (KRB5 or SPNEGO) to authenticate the user, while using the directory to authorize user privileges in the database.

### Advantages

- Some sites require use of Kerberos instead of LDAP for directory user authentication.
- If you enable non-LDAP external authentication, users can use single sign-on, which is not available with LDAP authentication, and still be authorized by the directory instead of by the database. See [About External Authentication](#).

### Limitations

- Directory authorization of privileges requires a complex and time-consuming setup process.
- If you decide to use Teradata directory schema extensions for directory user provisioning, and you install them in Active Directory, ADAM, or AD LDS, you cannot remove the extensions or any related schema objects you create (although you can change object mappings). Other directory types allow you to remove schema objects.

## Setting Up Non-LDAP External Authentication with Directory Authorization

1. Make sure users that will use this method:
  - Are defined to Kerberos.
  - Are defined in the directory in such a way that they can be located by an <Identity Map> or <Identity Search>. See [Optimizing Directory Searches](#).
2. Complete the setup tasks listed for [Option 2: Directory Authentication and Authorization](#), with the following changes:
  - a. Do not configure the LDAP mechanism, because it is not used for authentication.
  - b. Copy the following mechanism properties from the LDAP mechanism in the TDGSS library configuration file, into the TdgssUserConfigFile.xml for the authentication mechanism, KRB5 or SPNEGO:

LdapServerName

Optional LDAP identifications properties, if needed. See [Optimizing Directory Searches](#).

---

**Note:**

Some identification properties do not apply to this option.

---

- c. Because this option requires service binds, Teradata strongly recommends that you implement TLS protection. See [Using TLS with a Directory Server](#).

---

**Note:**

Non-LDAP authentication ignores the LdapClientMechanism property setting.

---

- d. Set the authentication mechanism (KRB5 or SPNEGO) as the default on all affected clients, or instruct users to specify the mechanism in the logon string.

3. You can use either of the these logon forms:

- [Logging on Using Sign-on As](#)
- [Logging on Using Single Sign-on with Kerberos](#)

## Option 4: Lightweight LDAP Authorizations

Lightweight LDAP Authorizations allow you to use your existing directory service to authorize Teradata Vantage users without modifying your directory to include structures or entries from Teradata schema extensions. Lightweight LDAP Authorizations maps Vantage external roles to existing directory groups.

### Advantages

- Works with LDAPv3 compliant directory servers (LDAP and KRB5).
- No Teradata Vantage infrastructure or objects need to be added to the customer directory server.
- Because Teradata-specific entries are not required in the directory, you can use directory management tools, such as Microsoft Management Console snap-ins to manage your directory.
- There is no impact to your current LDAP configuration. If you previously configured Teradata-specific objects in your directory you can continue to use that model and this new capability will not affect you.

You cannot use both lightweight LDAP authorizations and Teradata-specific directory objects. To switch to lightweight authorizations simply modify the TdgsUserConfigFile.xml. You can leave your Teradata-specific objects in the directory. You can optionally remove the Teradata-specific entries from your directory after you are sure lightweight authorizations meets the needs of your site.

### Installs, Upgrades, and Backdown

Lightweight LDAP authorizations must be manually enabled on installs and upgrades. To enable lightweight authorizations, include one or more AuthSearch elements. For more information, see [Setting Up Lightweight LDAP Authorizations](#).

To back down from Release 16.0 or higher to any pre-16.0 release, remove all software and perform a fresh install followed by a system initialization (sysinit).

To backdown:

1. Make a backup of the TdgssUserConfigFile.xml file.
2. Edit TdgssUserConfigFile.xml to remove any edits that are not compatible with the target version.
3. Run tdsstestcfg to verify the new configuration is correct.

For information about the tdsstestcfg command, see [Working with tdsstestcfg](#).

4. Run the run\_tdgssconfig utility: /opt/teradata/tdgss/bin/run\_tdgssconfig
5. If run\_tdgssconfig indicates that a TPA reset is needed, run tpareset to activate the changes to the TDGSS configuration.

```
tpareset -f "use updated TDGSSCONFIG GDO"
```

## Lightweight LDAP Authorization Modes and Compatibility

- If you do not want to use lightweight LDAP authorizations do not add <AuthSearch> to TdgssUserConfigFile.xml.
- If you no longer want to use lightweight LDAP authorizations remove <AuthSearch> from TdgssUserConfigFile.xml.
- Pre-16.0 clients may connect to Teradata Database 16.0 or higher and Teradata Vantage using lightweight LDAP authorizations; any client that is compatible with the currently installed release of Teradata Vantage may use lightweight LDAP authorizations.
- If the user is a member of multiple directory groups, all the groups are included in the search and the names of the groups identify the external roles the user can occupy.
- If a user is not a member of any directory group, then no role is returned. The user is allowed to log on, but the user is not allowed to occupy external roles. This is equivalent to authentication-only logons.
- If the user authentication fails, the logon fails.

## Search Performance

The lightweight LDAP authorizations feature searches your LDAP directory for groups and maps them to Vantage external roles. For the most efficient searches Teradata recommends limiting the scope of the directory search; for example, by adjusting the search base and scope (onelevel vs. subtree). This is similar to how you optimize the scope of searches for users as discussed in [Optimizing Directory Searches](#).

## Setting Up Lightweight LDAP Authorizations

### Prerequisites

- LDAPv3 compliant directory server, either LDAP or KRB5. See [About Certified Directories](#).
- External roles defined in Teradata Vantage. See [Creating and Dropping External Roles](#).

**Note:**

The user does not need to belong to the roles in the database. However, the directory user needs to belong to a group that maps to a role. That gives the user the permission to occupy the role via `SET ROLE <extrolename>`.

- Group entries in the directory which correspond to the external roles.

## Setting Up the Teradata Vantage Server to use Lightweight Authorizations

1. On the Vantage node with the lowest ID number, navigate to the directory where `TdgssUserConfigFile.xml` is located:

```
cd /opt/teradata/tdat/tdgss/site
```

2. Make a backup copy of `TdgssUserConfigFile.xml`.
3. Edit `TdgssUserConfigFile.xml` to allow TDGSS to search the directory for group-like entries:
  - a. Set `AuthorizationSupported` to yes.
  - b. Set `AuthenticationSupported` to yes.
  - c. Add an `<AuthSearch>` section.

The `<AuthSearch>` section goes in the `<Canonicalizations>` area of the `<LdapConfig>` section or as a child of the `<MechanismProperties>` element. In both cases, `<AuthSearch>` is a sibling of `<IdentitySearch>` and `<IdentityMap>` elements.

For example:

```
Mechanism Name="ldap">
  <MechanismProperties
    AuthenticationSupported="yes"
    AuthorizationSupported="yes"
    LdapBaseFQDN="dc=example,dc=com"
    ... />
  <AuthSearch
    Ref="service-id"
    Base="search-base"
    Scope="{onelevel|subtree}"
    MemberAttribute="member-attribute-name"
    ObjectClass="object-class-name"
    NamingAttribute="naming-attribute-name"
    <AuthSearchMap Match="regex" Pattern="pattern"/>
  />
</Mechanism>
```

See [<AuthSearch>](#) for details about each element.

## 4. Verify the configuration is correct:

- a. Run `tdgssstestcfg` to test the configuration. It launches a test environment in a new shell that contains the updates to the configuration file.

```
/opt/teradata/tdgss/bin/tdgssstestcfg
```

- b. Run the `tdgssauth` utility to test the new configuration before you commit the changes to the TDGSSCONFIG GDO.

See [Working with tdgssauth](#).

- c. Exit the test shell:

```
exit
```

- d. Continue editing and testing until the configuration is correct.

## 5. Update TDGSSCONFIG.GDO. Run:

```
/opt/teradata/tdgss/bin/run_tdgssconfig
```

**<AuthSearch>**

This feature maps existing or new directory entries that behave like groups to Vantage external roles. The directory is searched to establish a principal's authorizations.

The `<AuthSearch>` element enables lightweight authorizations.

The configuration is added to `TdgssUserConfigFile.xml` and defines the search parameters needed to locate group-like entries in a directory server.

```
<AuthSearch
  Ref="service-id"
  Base="search-base"
  Scope="{onelevel|subtree}"
  MemberAttribute="member-attribute-name"
  ObjectClass="object-class-name"
  NamingAttribute="naming-attribute-name"
  <AuthSearchMap Match="regex" Pattern="pattern"/>
/>...

  <AuthSearch
  <AuthSearchMap Match="manager" Pattern="admin"/>
  <AuthSearchMap Match="social" Pattern="tduser">
  />

</Mechanism>
```

## Attributes

Attribute	Description
Ref	<p>The Ref attribute is used only when the &lt;AuthSearch&gt; is placed inside the &lt;Canonicalizations&gt; area of the &lt;LdapConfig&gt; section. It is not used when &lt;AuthSearch&gt; is placed inside the &lt;MechanismProperties&gt; element.</p> <p>This attribute names the &lt;Service&gt; that performs the search. The value in the Id attribute of &lt;Service&gt; must be provided. The same rules apply to the Ref attribute that apply to the Ref attribute in the &lt;IdentitySearch&gt; and &lt;IdentityMap&gt; elements.</p> <p>The search is performed in the service that identified the user. The same service associated with the canonicalization that generates the principal's distinguished name (DN) will be searched for mappings.</p>
Base	<p>The Base attribute is optional. If not provided, the value is taken from the mechanism's or service's LdapGroupBaseFQDN property. If an LdapGroupBaseFQDN property is not provided, the value comes from the LdapBaseFQDN. The value defines the location where the search is to start to locate the group-like entries that represent Vantage external roles.</p>
Scope	<p>The Scope attribute is optional. This attribute specifies the search scope. It may take one of two values, "onelevel" or "subtree" with "subtree" being the default.</p>
MemberAttribute	<p>The values of the MemberAttribute and ObjectClass attribute are used to construct a search filter.</p> <p>The MemberAttribute attribute is optional. MemberAttribute names the attribute in the directory which contains the DN of the identified user. If the attribute is omitted, the value defaults to "member". This attribute MUST use the distinguishedNameMatch equality matching rule and the 1.3.6.1.4.1.1466.115.121.1.12 (distinguished name) syntax rule.</p>
ObjectClass	<p>The ObjectClass attribute is optional. If provided, it names the object class of the authorization entry and causes an ObjectClass term to be included in the search. If the attribute is omitted, no ObjectClass term is included in the search filter.</p> <p>The ObjectClass "group" has a member attribute (MemberAttribute) that is optional. The ObjectClass "groupOfNames" has a member attribute that must have at least one user named in it. Another ObjectClass is "groupOfUniqueNames" with a member attribute called "uniqueMember" that only allows a name to appear once.</p> <p>If you don't set the ObjectClass the default sets memberAttribute to "member" and ObjectClass to "groupOfNames". For Active Directory, ADAM, and AD LDS set ObjectClass to "group".</p> <p>If an object class name is present, the generated search filter will be of the following form: (&amp;(member-attribute-name=dn-of-principal)(objectClass=object-class-name))</p> <p>If an object class name is not present, the generated search filter will be of the form: (member-attribute-name=dn-of-principal)</p>
NamingAttribute	<p>The NamingAttribute attribute is optional. The value of this attribute names the attribute in the directory entry which contains the external role name. It is used to limit the returned attributes to this one attribute. If omitted, it defaults to "cn". If the attribute contains multiple values, then all values are candidates for permitted roles. If the attribute is not present (has no values), the containing entry is ignored.</p>
AuthSearchMap	<p>The AuthSearchMap element is optional. Configure as many AuthSearchMap elements as necessary. The Match and Pattern attributes of &lt;AuthSearchMap&gt; are used to filter out unwanted group names. If the group name returned from the directory</p>

Attribute	Description
	is not completely consumed by the regex pattern, it is discarded. If it matches, then the external role is derived according to the substitution pattern in the Pattern attribute. If the <AuthSearchMap> element is absent in the corresponding <AuthSearch> element then the group names are used as external roles as is.

## Editing Guidelines

- To set a value, you must manually add this property to the TdgssUserConfigFile.xml on the mechanisms that require it. See [About Editing Configuration Files](#).
- Zero or more <AuthSearch> elements are placed in the <Canonicalizations> area of the <LdapConfig> section or as a child of the <MechanismProperties> element. In both cases, the <AuthSearch> is a sibling of <IdentitySearch> and <IdentityMap> elements.
- Like <IdentitySearch> and <IdentityMap>, as many <AuthSearch> elements as necessary may be placed in the configuration. These elements may appear anywhere in the appropriate sections. They can be placed before or after the <IdentitySearch> and/or <IdentityMap> elements or they can be interleaved with the <IdentitySearch> and <IdentityMap> elements.
- The LdapSystemFQDN attribute does not need to be populated in the configuration. If populated, it will be ignored when <AuthSearch> is present.
- If UseLdapConfig="yes" for the mechanism, then the <LdapConfig> section of the configuration file should contain the <AuthSearch> entries. To use this feature a value for the LdapServiceFQDN and LdapServicePassword properties must be provided and user identities must be discovered using <IdentitySearch> and/or <IdentityMap> elements. The LdapServiceFQDN and password are used to do the canonicalization of the user. After canonicalization, but before Authentication, the authorization is retrieved using the service bind. All the authorization entries (roles) will be added to a roles list. The roles list will be returned after successful authentication. These roles will be used for database logon.

## Examples Using Lightweight LDAP Authorizations

The following examples show how to use <AuthSearch> to enable lightweight LDAP authorizations.

For an explanation of <AuthSearch> elements, such as Ref, Base, Scope, and so on, see [<AuthSearch>](#).

### Example: <AuthSearch> in the <Mechanism> Section of TdgssUserConfigFile.xml

The example uses a default scope="subtree", default MemberAttribute="member", and default NamingAttribute="cn".

If the base is not provided, the value is taken from the mechanism's or service's LdapGroupBaseFQDN property. If an LdapGroupBaseFQDN property is not provided, the value comes from the LdapBaseFQDN. So the default value of base comes from LdapBaseFQDN which is "dc=example,dc=com".

In this case, a search of `dc=example,dc=com` is done. The search matches the contents of the attribute named `member` with the DN of the principal that represents the user logging on. The contents of the Common Name (CN) attribute are fetched and merged into a list that becomes the list of groups. The generated search filter used for directory search is `(member=dn-of-principal)`.

```
<Mechanism Name="ldap">
  <MechanismProperties
    AuthenticationSupported="yes"
    AuthorizationSupported="yes"
    LdapBaseFQDN="dc=example,dc=com"
    ... />
  <AuthSearch/>
</Mechanism>
```

For information about `LdapBaseFQDN` and `LdapGroupBaseFQDN`, see [Configuring LDAP Properties to Narrow the Search Base](#).

#### **Example: <AuthSearch> in the <LdapConfig> Section of TdgssUserConfigFile.xml**

In the example, if the user is authenticated in the service “my-svc”, then the `<AuthSearch>` elements whose `Ref` attributes contain “my-svc” are used to locate the lightweight authorizations for the user. And the search filter used to search the directory is `(member=dn-of-principal)`.

```
<Mechanism Name="ldap">

  <MechanismProperties
    AuthenticationSupported="yes"
    AuthorizationSupported="yes"
    UseLdapConfig="yes"
    ... />

</Mechanism>
...

<LdapConfig>
  ...

  <Services>
    <Service
      Id="my-svc"
      LdapBaseFQDN="dc=example,dc=com"
      ... />
    ...
  </Services>
```



```

    <Canonicalizations>
      <AuthSearch Ref="my-svc"/>
      ...
    </Canonicalizations>
    ...
  </LdapConfig>

```

### Example: Using Nested Groups in Active Directory

In this example, the extensible match operator LDAP\_MATCHING\_RULE\_IN\_CHAIN is added to the search filter by including the OID in the MemberAttribute. The OID 1.2.840.113556.1.4.1491 asks Active Directory to find all groups that claim the user as a member. For example, if group A were a member of group B and the user is a member of group A, then this match causes both groups A and B to be returned. The user has a membership in group B because the user is a member of group A and group A is a member of group B. If the OID were dropped from the MemberAttribute attribute's value, then the search would yield only group A. The search filter used to do the directory search is (member:1.2.840.113556.1.4.1491:=dn-of-principal).

This kind of search performs poorly in Active Directory because it requires multiple passes over the directory information tree. The more deeply nested a candidate group is, the worse the search performs. Teradata does not recommend this kind of search in high performance environments, but it is presented here to illustrate the flexibility of the <AuthSearch> element.

```

<Mechanism Name="ldap">

  <MechanismProperties
    AuthenticationSupported="yes"
    AuthorizationSupported="yes"
    LdapBaseFQDN="dc=example,dc=com"
    ... />

  <AuthSearch
    MemberAttribute="member:1.2.840.113556.1.4.1491:"
    Base="dc=example,dc=com"
    Scope="subtree"
    NamingAttribute="cn"
    <AuthSearchMap Match="." Pattern="{0}"/>
  />

</Mechanism>

```

For information about supported match operators, see the documentation for your particular directory server.

**Example: Using groupOfUniqueNames in <AuthSearch>**

In the example, ObjectClass is used to construct a search filter. ObjectClass names the object class of the authorization entry and causes an objectClass term to be included in the search. In the example, the search filter used in the directory search is (&(ObjectClass=groupOfUniqueNames)(uniqueMember=dn-of-principal)).

```
<Mechanism Name="ldap">

  <MechanismProperties
    AuthenticationSupported="yes"
    AuthorizationSupported="yes"
    LdapBaseFQDN="dc=example,dc=com"
    ... />
  <AuthSearch
    ObjectClass="groupOfUniqueNames"
    MemberAttribute="uniqueMember"/>

</Mechanism>
```

For more information on ObjectClass, see [<AuthSearch>](#).

**Example: Using Multiple <AuthSearch> Elements**

The example performs three different searches each with subtree scope. Each search gets its own search base. The generated search filter is "(member=dn-of-principal)" and the role names are gathered from the values in the returned object's CN attribute.

```
<Mechanism Name="ldap">

  <MechanismProperties
    AuthenticationSupported="yes"
    AuthorizationSupported="yes"
    LdapBaseFQDN="dc=example,dc=com"
    ... />

  <AuthSearch
    Base="ou=groups,ou=americas,dc=example,dc=com"/>
  <AuthSearch
    Base="ou=groups,ou=emea,dc=example,dc=com"/>
  <AuthSearch
    Base="ou=groups,ou=apj,dc=example,dc=com"/>

</Mechanism>
```

**Example: Using Multiple AuthSearchMap Elements in <AuthSearch>**

In the example, the generated search filter used for directory search is (member=dn-of-principal) and group names are returned from the directory search. If the directory group name is *manager* then the external role in Vantage is *admin*. If the directory group name is *socal* then the external role in Vantage is *tduser*.

```
<Mechanism Name="ldap">

  <MechanismProperties
    AuthenticationSupported="yes"
    AuthorizationSupported="yes"
    LdapBaseFQDN="dc=example,dc=com"
    ... />

  <AuthSearch>
    <AuthSearchMap Match="manager" Pattern="admin"/>
    <AuthSearchMap Match="socal" Pattern="tduser"/>
  </AuthSearch>

</Mechanism>
```

## About Directory User Characteristics

Directory users have different characteristics and database privileges depending on whether the users are mapped to Vantage directory objects.

## Characteristics of Unmapped Directory Users

### If All Directory Users Are Unmapped

If no directory users are mapped to Teradata Vantage users, you can set the LDAP mechanism AuthorizationSupported property no, to allow directory users with a username that matches a Vantage username to:

- Log on to the database and be authenticated by the directory
- Inherit all the database privileges of the matching database user

Unmapped directory users whose user names do not match a database username cannot access the database. The exception is EXTUSER. If a user is assigned to EXTUSER, the user is provided limited database access in the same way that PUBLIC provides limited access to permanent database users. Additionally, if a user is assigned to EXTUSER or assigned to a role or profile, and if auto provisioning is configured on the system, an individual database account will automatically be created for the user.

## If Some Directory Users Are Unmapped

If some directory users are mapped to Vantage users and other users are not, the setting of the LDAP AuthorizationSupported property determines which users can log on:

- If AuthorizationSupported="no", only the unmapped users with names that match database user names can log on to the database, with the privileges of the matching database users.
- If AuthorizationSupported="yes", only directory users that are mapped to Vantage objects can log on to the database, with privileges based on the mapped objects.

## Characteristics of Directory Users Mapped to Permanent Database Users

Mapped directory users function in much the same way as the permanent Teradata Vantage users to which they are mapped.

Directory users mapped to a permanent Vantage users:

- Inherit access privileges, roles, and profile settings from the permanent user. If the directory user is mapped to more than one database user, logons must specify user= username in the .logdata statement to identify the operant user, profile, and default role.
- Inherit the spool and temp space available the permanent user, unless also mapped to a profile object, in which case the mapped profile takes precedence.
- Can be mapped to role objects in addition to the roles inherited from mappings to database users.
- Must use the SET ROLE statement (within a session) to enable any roles other than the default role inherited from the operant permanent user.

## Characteristics of Directory Users Mapped to EXTUSER

Teradata Vantage automatically generates the default user EXTUSER. You can map directory users to EXTUSER to provide limited database access in the same way that PUBLIC provides limited access to permanent database users.

- EXTUSER has privileges similar to PUBLIC by default.  
For information on PUBLIC privileges, see [Default PUBLIC Privileges](#).
- You can map a directory user to EXTUSER.
- Mapping a directory user to any Vantage object also gives the user EXTUSER privileges.
- EXTUSER has no PERM space, default database, default role, or profile.
- EXTUSER has the same default spool and temp space allocations as user DBC. The allocations apply collectively to all users mapped to EXTUSER. You can create profiles that set lower spool and temp limits and map them to directory users mapped to EXTUSER.
- Although the system creates an entry for EXTUSER in DBC.Dbase, it is not a valid user or database, and you cannot reference it in a logon string or SQL statement.

- A user mapped only to EXTUSER can set the default database using a SET SESSION DATABASE statement.
- You cannot configure workload management rules for users mapped only to EXTUSER.
- If a directory user is mapped to EXTUSER or there is a role or profile mapping for a directory user, the directory user is logged on to the database as EXTUSER; however, if the DBSControl AutoProvision parameter is enabled, a database account is automatically provisioned for the user and they are logged on to the database as that user.

## Characteristics of Directory Users Mapped to Database Roles and Profiles

You can map directory users to roles and profiles other than those they inherit from the database users to which they are mapped.

If auto provisioning is configured on the system, users assigned to a role or profile are automatically provisioned with an individual database account.

Consider the following conditions and limitations when mapping directory users to roles and profiles:

- You cannot map directory users to standard database roles. Instead, you must create external roles, using the CREATE EXTERNAL ROLE statement, and then map the directory users to directory role objects named for the external roles.
- Mappings to directory profile and role objects take precedence over those inherited from a mapped database user.
- Directory users must use the SET ROLE statement (within a session) to enable the roles inherited from the permanent users to which they are mapped if they are also mapped to other roles.
- Although there is no limit to the number of external roles you can map to a directory group object, the database recognizes a maximum of 50 roles. If the number of external roles mapped to a group exceeds 50, database logons by members of the group fail.

For information on creating external roles, see [Using Roles for Directory Users](#).

For information profiles, see the topics beginning with [Working with Database Profiles](#).

## Row Level Access Privileges for Directory Users

Mapped directory users can access objects protected by row level security based on the security constraint assignments for any mapped profiles and users.

Unmapped directory users who sign on as a permanent database user inherit the constraint assignments for the user.

For directory user row level security session processing and options, see [Session Constraint Values for Directory Users](#).

## Ownership of Database Objects Created by Directory Users

Directory users can create users and databases if permitted by the privileges granted to the users or roles to which they are mapped, however, ownership of these objects follows different rules than those that apply to permanent users.

- Directory users do not have database identifiers and therefore cannot be owners or creators of database objects.
- The system registers the users and databases created by directory users in DBC.Dbase, with the mapped permanent database user listed as the object owner and creator.
- Objects created by directory users who are not mapped to permanent database users are owned by the database that contains the created object.

# Evaluating the System for Directory Management of Users

Before you implement directory management of database users, you should check the system to determine whether it is ready for you to execute the directory setup tasks.

## Evaluation Process

- Determine if your directory is certified. See [About Certified Directories](#).
- Check network connections to ensure they follow the required naming conventions. See [Checking the Network Setup](#).
- Test communication with the directory server to ensure that it is adequate for authenticating users. See [Querying the Directory Server](#).
- Review common connection error messages and hints on possible solutions. See [Common Errors with Active Directory, ADAM, or AD LDS](#) and [Common Errors with Sun Java Directory Server](#).

## Prerequisites

The evaluator must have access to the ldapsearch and tdgssauth tool, and administrator privileges in the directory and the database.

## About Certified Directories

Teradata Vantage interfaces with all LDAP-compliant directories, but it is only certified with the directories in the following table.

With DIGEST-MD5 Binding	With Simple Binding
<ul style="list-style-type: none"> <li>• Active Directory on Windows 2003 or later.</li> <li>• Sun Java System Directory Server 5.2 or later.</li> <li>• Novell eDirectory 8.2 or later.</li> <li>• OpenLDAP 2.3 or later</li> </ul> <p><b>Note:</b> The DIGEST-MD5 authentication protocol used by LDAP is deprecated. Teradata strongly recommends you use simple binding with TLS protection, and stop using DIGEST-MD5.</p>	<ul style="list-style-type: none"> <li>• Active Directory (All versions)</li> <li>• Active Directory Application Mode (ADAM), all versions; on Windows Server Release 2 or later</li> <li>• Active Directory Lightweight Directory Services (AD LDS), all versions</li> <li>• Novell eDirectory SP2</li> <li>• Sun Java System Directory Server version 5.2 or later</li> <li>• OpenLDAP 2.3 or later</li> </ul>

## Working with LDAPv3-Compliant Directories

- RootDSE reads can be done using an anonymous bind or a service bind. For directories that do not allow anonymous binds to the RootDSE, a service bind is done if a service DN is provided. If the Service DN is not specified or its password is not good, the directory must be set up to allow anonymous reads of the RootDSE object.
- Implementation requirements for uncertified LDAPv3- compliant directories are similar to the requirements for certified directories, with the following exceptions:
  - Teradata directory schema extensions do not support uncertified LDAPv3-compliant directories. You must use native directory schema to implement management of Teradata Vantage users on these uncertified directories. See [Using Native Directory Schema to Provision Directory Users](#).
  - Authentication of Vantage users by an uncertified LDAPv3-compliant directory must use simple binding. See [LDAP Binding Options](#).

Contact Teradata Professional Services if you need to use an uncertified LDAP-compliant directory.

## Checking the Network Setup

Do the following to check network connections to the directory from both the client and server (database).

## Connecting Teradata Vantage Clients to the Database

To connect a client to the four node Vantage system named TDSYS, you must configure four network interfaces. The DNS configuration (or client hosts file) binds each interface IP address to a named node, based on a numbered cop designation.

## Typical Server Configuration in Client Machine Host File or DNS

```
192.168.1.50 tdsyscop1
192.168.1.51 tdsyscop2
192.168.1.52 tdsyscop3
192.168.1.53 tdsyscop4
```

### Note:

Kerberos has special DNS requirements that differ from those required for LDAP. If you plan to use both Kerberos and LDAP authentication, you must set up the DNS to include the Kerberos requirements. See Kerberos documentation for details.



## About Teradata Vantage Node Connections to the Directory

The DNS host and domain names for the directory server, which appear on each Vantage node, must match the actual DNS host and domain name for the directory. If you do not consistently define the directory server throughout the DNS, LDAP user authentication fails.

## Checking the Directory DNS Name from the Database

1. On every node of the Vantage system, use dig or nslookup to see the fully qualified DNS name of the directory server. If either utility returns multiple name registrations for a node, Digest-MD5 binding is probably not usable for LDAP authentication of database users, and you should use simple binding. See [LDAP Binding Options](#).
2. On the directory server, use a directory tool to find the DNS name in the directory.
3. The names you obtain from the directory server and the Vantage nodes must match. Fix any mismatches on the database nodes.

## Querying the Directory Server

After you verify basic network setup, you can use the ldapsearch tool to explore the communication link between the directory server and Vantage nodes to ensure that the level of communication is sufficient to authenticate directory users. Use ldapsearch to:

- Find the RootDSE object for your directory type, and compare critical attribute values to database settings and requirements. See [Finding the RootDSE Object](#).
- Authenticate a Directory User. See [Authenticating a Directory User](#).

## Prerequisites

The examples that follow use ldapsearch to query the directory. On the Teradata Vantage system, the tdgss.bin directory contains the ldapsearch tool.

For detailed information on how to use the ldapsearch tool, see [Working with Ldapsearch](#).

---

### Note:

Make sure that the tdgss.bin directory appears somewhere in your PATH environment variable or the scripted examples that follow cannot function.

---

## About Ldapsearch

The following table contains descriptions of the ldapsearch command arguments. For additional information, see [Working with Ldapsearch](#).

Command Arguments	Description
-b " "	Specifies the search base. Because the RootDSE object has no name, the argument passes a zero-length string.
-D	Directory user ID in the form of a dn. Use only when ldapsearch specifies simple binding (-x).
-H ldap://esroot	Identifies the binding scheme and the directory server name.
namingContexts	Limits the output of the ldapsearch and does not return other information.
-s base	Specifies the scope of the search, from among three choices: <ul style="list-style-type: none"> <li>• base</li> <li>• one</li> <li>• sub</li> </ul> The examples use base to indicate that the scope is the base.
-U	The SASL user ID. Use this argument to specify the user ID only when also specifying -Y DIGEST-MD5 binding.
-W	Requests a prompt for entry of the password for the user.
-x	Specifies simple binding for the ldapsearch. The binding method specified for ldapsearch is independent of the binding method you use generally for LDAP authentication.
-Y DIGEST-MD5 [Deprecated]	Specifies DIGEST-MD5 binding for the ldapsearch. The binding method specified for ldapsearch is independent of the binding method you use generally for LDAP authentication.  <b>Note:</b> The DIGEST-MD5 authentication protocol used by LDAP is deprecated. Teradata strongly recommends you use simple binding with TLS protection, and stop using DIGEST-MD5.
-Z	Requests TLS protection. The directory server may or may not grant the request, and if not, the connection runs in plain text. To demand TLS protection, use -ZZ.

## Finding the RootDSE Object

All directories contain the RootDSE object. This object contains information about the directory server, which you must compare to certain Teradata Vantage conditions and requirements.

## Example: Using Ldapsearch to Find the RootDSE in Active Directory, ADAM, or AD LDS

You can use the ldapsearch tool to find and display the contents of the RootDSE object from an Active Directory, ADAM, or AD LDS directory server.

For descriptions of the options used in this search, see [About Ldapsearch](#).

### Note:

The phrase ...snipped... indicates output sections that the example does not show, because they do not apply to the directory interface with Teradata Vantage.

```
$ ldapsearch -x -H ldap://esroot -b "" -s base
dn:
currentTime: 20040820001616.0Z
subschemaSubentry: CN=Aggregate,CN=Schema,CN=Configuration,
DC=esrootdom,DC=esdev,DC=tdat
dsServiceName: CN=NTDS Settings,CN=ESROOT,CN=Servers,
CN=Default-First-Site-Name,CN=Sites,CN=Configuration,DC=esrootdom,
DC=esdev, DC=tdat
namingContexts: DC=esrootdom,DC=esdev,DC=tdat
namingContexts: CN=Configuration,DC=esrootdom,DC=esdev,DC=tdat
namingContexts: CN=Schema,CN=Configuration,DC=esrootdom,DC=esdev,
DC=tdat
namingContexts: DC=DomainDnsZones,DC=esrootdom,DC=esdev,DC=tdat
namingContexts: DC=ForestDnsZones,DC=esrootdom,DC=esdev,DC=tdat
defaultNamingContext: DC=esrootdom,DC=esdev,DC=tdat
schemaNamingContext: CN=Schema,CN=Configuration,DC=esrootdom,DC=esdev,
DC=tdat
configurationNamingContext: CN=Configuration,DC=esrootdom,DC=esdev,
DC=tdat
rootDomainNamingContext: DC=esrootdom,DC=esdev,DC=tdat
supportedControl: 1.2.840.113556.1.4.319
...snipped...
supportedLDAPVersion: 3
...snipped...
supportedSASLMechanisms: DIGEST-MD5
dnsHostName: esroot.esrootdom.esdev.tdat
ldapServiceName: esrootdom.esdev.tdat:esroot$@ESROOTDOM.ESDEV.TDAT
serverName: CN=ESROOT,CN=Servers,CN=Default-First-Site-Name,CN=Sites,
CN=Configuration,DC=esrootdom,DC=esdev,DC=tdat
...snipped...
```

```
domainControllerFunctionality: 2
$
```

**Note:**

If the directory does not allow an anonymous read, a valid user identity and password must be presented and the database will require a service ID and password in order to use this directory service.

The output of the example `ldapsearch` command shows the contents of the RootDSE object, including the following critical attributes:

- The `supportedLDAPVersion` attribute is set to 3. This value indicates that the directory is compliant with LDAPv3, the only LDAP version that Teradata Vantage supports.
- The `supportedSASLMechanisms` attribute shows DIGEST-MD5, indicating that the RootDSE object supports DIGEST-MD5. This is not related to the binding method specified in the `ldapsearch`, in this case `-x` simple binding.

**Note:**

The DIGEST-MD5 authentication protocol used by LDAP is deprecated. Teradata strongly recommends you use simple binding with TLS protection, and stop using DIGEST-MD5.

- The `dnsHostName` attribute contains the fully qualified DNS name for the directory server. All nodes of the Teradata Vantage server must resolve the host name of the directory through the system name resolution/lookup service in a way that exactly matches the data in this attribute.

## Finding the RootDSE on Sun Java System Directory Server

You can use the `ldapsearch` tool to find and display the contents of the RootDSE object from an a Sun Java System Directory Server. The command for Sun Java Directory Server is the same as for Active Directory. See [Example: Using ldapsearch to Find the RootDSE in Active Directory, ADAM, or AD LDS](#).

The output of the command is similar for all directories, with the exception of Sun Java Directory Server, which does not support the `dnsHostName` attribute. To find the `dnsHostName` and ensure that all Vantage nodes resolve the host name in a way that matches how the name appears in the directory, you must examine the directory with a standard directory tool.

**Note:**

Sun includes a `vendorName` and `vendorVersion` attribute (among others). Microsoft includes a larger set of attributes, and as result, returns a larger output.

## Example: Using Ldapsearch to Find the RootDSE in Sun Java Directory Server

```
$ ldapsearch -H ldap://elsegundoca -b "" -s base
dn:
objectClass: top
namingContexts: dc=elsegundoca,dc=teradata,dc=com
namingContexts: o=NetscapeRoot
supportedExtension: 2.16.840.1.113730.3.5.7
...snipped...
supportedControl: 2.16.840.1.113730.3.4.2
...snipped...
supportedSASLMechanisms: DIGEST-MD5
...snipped...
supportedLDAPVersion: 3
vendorName: Sun Microsystems, Inc.
vendorVersion: Sun-ONE-Directory/5.2
dataversion: 020040814221707020040814221707
netscapemdsuffix: cn=ldap://dc=djltnt,dc=elsegundoca,dc=teradata,dc=com:389
$
```

## Ldapsearch Troubleshooting

If Ldapsearch returns an LDAP error rather than the RootDSE object, you may need to authenticate before reading the RootDSE. A possible correction is to create a service user for the database and repeat the test using -D and -w options to pass the user's identity and password respectively to the directory.

## Authenticating a Directory User

After you successfully obtain a RootDSE object and resolve the DNS name with all nodes of the Vantage system, you can use Ldapsearch to authenticate a directory user.

## Example: Using Ldapsearch to Authenticate an Active Directory, ADAM, or AD LDS User

This example demonstrates the command to cause the Administrator user on a Windows network to be authenticated with Active Directory, ADAM, or AD LDS. For demonstration purposes, the example uses a -u option to name the user, and shows an attempt to obtain the RootDSE object as an authenticated user. However, the example input only asks for the namingContext attribute rather than all attributes as shown in other examples, such as [Example: Using Ldapsearch to Find the RootDSE in Active Directory, ADAM, or AD LDS](#).

If this command is run from a Windows machine, a -d option, which names the Windows domain where the authentication is to occur, can be included in the options.

If the command is successful it indicates that you can setup the directory and the TDGSS configuration files for directory authentication and authorization of database users.

If the command fails, refer to [Common Errors with Active Directory, ADAM, or AD LDS](#), to debug the problem.

```
$ ldapsearch -H ldap://esroot -b "" -s base -x -D user_dn -W namingContexts
Enter LDAP password:
dn:
namingContexts: DC=esrootdom,DC=esdev,DC=tdat
namingContexts: CN=Configuration,DC=esrootdom,DC=esdev,DC=tdat
namingContexts: CN=Schema,CN=Configuration,DC=esrootdom,DC=esdev,
DC=tdat
namingContexts: DC=DomainDnsZones,DC=esrootdom,DC=esdev,DC=tdat
namingContexts: DC=ForestDnsZones,DC=esrootdom,DC=esdev,DC=tdat
$
```

## Example: Using ldapsearch to Authenticate a Sun Java Directory Server User

This example authenticates the user against the Sun Java System Directory Server. You may find it beneficial to consult your directory administrator for the proper syntax for user names accepted by the directory.

In this test example, the user ID diperm01@testing is a valid user dn.

```
$ ldapsearch -H ldap://elsegundoca -b "" -s base -x -D diperm01@testing -W
-Z namingContexts
Enter LDAP password:
dn:
namingContexts: dc=elsegundoca,dc=teradata,dc=com
namingContexts: o=NetscapeRoot
$
```

## Common Errors with Active Directory, ADAM, or AD LDS

This topic describes common errors that you may encounter when using ldapsearch to verify information in Active Directory, ADAM, or AD LDS.

## DNS Naming Issue

An Active Directory, ADAM, or AD LDS server may see itself differently in DNS than the Teradata Vantage server sees it. This error can occur on any node in the Vantage system.

### Example: Error -- Mismatched DIGEST-uri and LDAP SPN

```
$ ldapsearch -H ldap://esroot -b "" -s base -Y DIGEST-MD5 -U diperm01@testing -W
Enter LDAP password:
Invalid credentials
additional info: 80090303: LdapErr: DSID-0C0903FB, comment:
The digest-uri does not match any LDAP SPN's registered for this
server., data 0, vece
$
```

#### Note:

The DIGEST-MD5 authentication protocol used by LDAP is deprecated. Teradata strongly recommends you use simple binding with TLS protection, and stop using DIGEST-MD5.

## Correcting the Mismatched DIGEST-uri and LDAP SPN Error

To correct this problem, add a line in the node hosts file that provides the correct fully qualified DNS name for the server or modify the actual DNS server and suffix search order on the node. Correcting the problem in the DNS configuration is the preferred solution.

To restart the DNS client, do the following:

1. Click **Start > Settings > Control Panel > Administrative Tools > Services**.
2. Right click **DNS Client**.
3. Click **Restart**.
4. Once the erroneous name resolution is corrected, use ping to verify that the Vantage node sees the directory server correctly. If the problem cannot be corrected, it may be necessary to ask your network, DNS or domain administrator for help

## Invalid User, Password, or Realm

These examples demonstrate the effect of a bad directory user name, password or realm name. The remedy is to correct the user name, password or realm and try the request again. Note that you can specify the realm name in the -d option on a Windows machine that uses Active Directory, ADAM, or AD LDS. All other uses of -d are ignored.

**Example: Error - Invalid User, Password, or Realm (Simple Binding)**

```
$ ldapsearch -H ldap://esroot -b "" -s base -D user_dn -W -x -Z
Enter LDAP password:
Invalid credentials
additional info: 8009030C: LdapErr: DSID-0C090419, comment:
AcceptSecurityContext error, data 0, vece
$
```

**Example: Error - Invalid User, Password, or Realm (DIGEST-MD5 Binding)**

```
$ ldapsearch -H ldap://esroot -b "" -s base -U user_dn -W -Y DIGEST-MD5 -Z
Enter LDAP password:
Invalid credentials
additional info: 8009030C: LdapErr: DSID-0C090419, comment:
AcceptSecurityContext error, data 0, vece
$
```

**Note:**

The DIGEST-MD5 authentication protocol used by LDAP is deprecated. Teradata strongly recommends you use simple binding with TLS protection, and stop using DIGEST-MD5.

**Bad Password****Note:**

The DIGEST-MD5 authentication protocol used by LDAP is deprecated. Teradata strongly recommends you use simple binding with TLS protection, and stop using DIGEST-MD5.

The SASL (Simple Authentication and Security Layer) DIGEST-MD5 mechanism, used in LDAP authentication, works using a shared secret. The directory server stores this secret in a database. The LDAP client obtains this shared secret from the user. Under the boards, DIGEST-MD5 on the directory server issues a challenge to the LDAP client. The client builds a response based on data in the server challenge and the secret obtained from the user and sends that response back to the server. At no time does the password appear in the communications between the client and server. The server also generates a response based on its stored secret. If the client's response to the server's challenge does not match what the server generated as a proper response, this error occurs.

These examples demonstrate the error that occurs when the user is successfully mapped to an FQDN that references an existing object in the directory and the password stored in the directory does not match what the user specified.



To remedy, correct the password on the client or have the directory administrator change the password on the directory server and retry the failed command.

### Example: Bad Password Error (DIGEST-MD5 Binding)

```
$ ldapsearch -U diperm01@testing -H ldap://esroot -b "" -s base -Z
Enter LDAP password:
Invalid credentials
additional info: SASL(-13): authentication failure: client
response doesn't match what we generated
$
```

### Example: Bad Password Error (Simple Binding)

```
$ ldapsearch -x -b "" -s base -H ldap://sussan140 -D user_dn -W
Enter LDAP Password:
ldap_bind: Invalid credentials (49)
additional info: 80090308: LdapErr: DSID-0C090334, comment:
AcceptSecurityContext error, data 52e, vece
```

The “52e” error code, shown in the last line of output (“data 52e”), indicates that a bad password was used.

## Ldapsearch Error Codes

Other common user errors are:

Error Code	Description
525	User not found.
52e	Invalid credentials.
530	Not permitted to log on at this time.
531	Not permitted to log on at this workstation.
532	Password expired.
533	Account disabled.
701	Account expired.
773	User must reset password.
775	User account locked.

## Common Errors with Sun Java Directory Server

This topic describes common errors you may encounter when using `ldapsearch` to verify information in Sun Java Directory Server.

### Server Down

Server down errors in Sun Java Directory Server are similar to those errors in Active Directory.

### Bad Canonicalization

#### Example: Detecting Bad Canonicalization

This example demonstrates an error that occurs when the directory server fails to translate the user name specified in the `-u` option to a fully qualified distinguished name (FQDN). In the directory is an object located by the FQDN, `cn=DIGEST-MD5, cn=identity mapping, cn=config`. The children of this object contain configuration information that assists the directory server in transforming the user name into an FQDN. In order to view the identity mappings, you must search the directory as the directory administrator.

The identity mappings found in the directory take one of two forms. The most efficient form is the one that uses pattern matching and substitutions. The other form executes a directory search based on the form of the user name.

```
$ ldapsearch -U diperm01@testing -H ldap://esroot -b "" -s base -W -Y DIGEST-
MD5 -Z
Enter LDAP password:
Invalid credentials
additional info: SASL(-1): generic failure: unable canonify user
and get auxprops
$
```

---

#### Note:

The DIGEST-MD5 authentication protocol used by LDAP is deprecated. Teradata strongly recommends you use simple binding with TLS protection, and stop using DIGEST-MD5.

---

#### Example: Bad Canonicalization with Identity Mapping

This example illustrates an identity mapping object that transforms a user name of the form `user@realm` to an appropriate FQDN. The content of the `dsMatching-pattern` specifies that the user name obtained from the `-u` option be transformed to an FQDN. The user name is then matched against the expression contained in the `dsMatching-regexp` attribute. Substitutions are made in the substitution pattern contained

in the `dsMapped` attribute. Then if you run the user name `diperm01@testing` through this identity mapping rule, the FQDN is `uid=diperm01, ou=people, ou=testing, dc=elsegundo, dc=teradata, dc=com`.

Before you design or change identity mappings, you should consult the directory and security administrators, since these objects represent closely guarded configuration information that could adversely affect other directory users and potentially compromise directory security.

For further information on identity mappings, please consult the *Directory Server Administration Guide* for the Sun Java System Directory Server. This guide can be found on the following website: <http://download.oracle.com>.

```
dn: cn=test mapping,cn=DIGEST-MD5,cn=identity mapping,cn=config
objectClass: top
objectClass: nsContainer
objectClass: dsIdentityMapping
objectClass: dsPatternMatching
cn: test mapping
dsMatching-pattern: ${Principal}
dsMappedDN: uid=$1,ou=people,ou=$2,dc=elsegundoca,dc=teradata,dc=com
dsMatching-regexp: ([^:]*)(.*)
```

---

**Note:**

The DIGEST-MD5 authentication protocol used by LDAP is deprecated. Teradata strongly recommends you use simple binding with TLS protection, and stop using DIGEST-MD5.

---

## Bad User

### Example: Bad User Error

This example demonstrates the error that occurs when either the identity mapping yields a FQDN that identifies an object that does not exist, or the `userPassword` attribute in the mapped directory object is missing or contains something other than cleartext data.

If the mapped FQDN does not identify an existing object, either the user name was entered incorrectly, or the object does not exist in the directory. If the user name was entered incorrectly, try the command again with a correct user name. If the problem persists, have the directory administrator either create or replace the contents of the `userPassword` attribute in the mapped object with a cleartext value and retry your command using the new password.

```
$ ldapsearch -D diperm01@testing -H ldap://esroot -b "" -s base -W -Y DIGEST-
MD5 -Z
Enter LDAP password:
Invalid credentials
```

```
additional info: SASL(-13): user not found: no secret in database
$
```

---

**Note:**

The DIGEST-MD5 authentication protocol used by LDAP is deprecated. Teradata strongly recommends you use simple binding with TLS protection, and stop using DIGEST-MD5.

---

# Provisioning Directory Users with Teradata Schema Extensions

If you manage Teradata Vantage users in a directory, you must provision the users, that is, create and configure the directory objects that define user privileges in the database.

You can provision users using only the native directory schema, or you can add Teradata schema extensions, which provide additional capabilities.

---

**Note:**

You cannot restrict user access by IP address unless you use Teradata schema extensions. See [Restricting Logons by IP Address](#).

---

For information on directory provisioning using the native directory schema, see [Using Native Directory Schema to Provision Directory Users](#).

## Provisioning Process

1. Review the descriptions of the Teradata schema extensions and related Teradata Vantage objects that you must use to provision directory users. See [About Teradata Schema Extensions](#).
2. Install Teradata schema extensions in the directory. See [Installing Teradata Schema Extensions in a Certified Directory](#).
3. Review the required hierarchical structure for Teradata schema objects in the directory information tree (DIT). See [About Teradata Schema Objects in the DIT](#).
4. Create the top level Teradata schema objects in the DIT structure. See [Creating the Top-Level Objects in the DIT](#).
5. Create the directory objects that represent Vantage users, external roles, profiles and IP restrictions and the container objects that contain them. See [Creating Containers and Inserting Objects](#).
6. Map directory users to Vantage users, external roles, profiles, and IP restrictions. See [Mapping Directory Users to Teradata Vantage Objects](#).

## Prerequisites

- Evaluate the readiness of the directory to manage Teradata Vantage users. See [Evaluating the System for Directory Management of Users](#).
- Develop a directory user management strategy. See [Working with Directory User Management Options](#).
- If directory users log on through Teradata Unity, review the additional configuration requirements in *Teradata® Unity™ Installation, Configuration, and Upgrade Guide for Customers*, B035-2523 and *Teradata® Unity™ User Guide*, B035-2520.

- Create Teradata user, external role, and profile objects in the database before creating their counterparts in the directory. See [Creating Users and Granting Privileges](#).
- Determine the binding and protection strategies for your site, and implement these strategies before proceeding. See:
  - [LDAP Binding Options](#).
  - [Using TLS with a Directory Server](#).

## About Teradata Schema Extensions

All directories contain a schema that defines:

- Objects that can inhabit the directory
- Attributes that each object must or can contain

The directory checks any new objects you add against the schema to ensure that the objects conform to schema rules.

In addition to the predefined schema that is native to each supported directory, Teradata provides schema extensions that you can load into the directory to link directory users to Teradata Vantage users, external roles, and profiles.

---

### Note:

For information on Teradata schema extensions used to define security policy, see [Network Security Policy](#).

---

## Teradata Directory Object Classes

Teradata schema extensions add two object classes to the standard directory schema:

- Required: Must appear in the Directory Information Tree hierarchy.
- Optional: Need not be present if the function of the object is not used.

For further information on where objects appear in the hierarchy, see [Teradata Schema Objects in the DIT Hierarchy](#).

## Teradata Schema Extensions

### Teradata Schema Objects

After you install Teradata schema extensions in a supported directory, you can create several objects that can link directory users to Teradata Vantage users, roles, and profiles.

Object Type	Description	Class
tdatRootNode	Parent object for all tdatSystem objects. The directory administrator determines the value for cn attribute.	Required
tdatSystem	Identifies a Teradata Vantage system or Unity server, where <i>cn=system_name</i> . The tdatSystem object is the parent of the Teradata schema objects that define the LDAP authorization structure and mappings to directory users.  <b>Note:</b> The tdatSystem object maybe named for a single system, but multiple systems that use the same authorization structure can specify the same tdatSystem object.	Required
tdatContainer	tdatContainer objects contain individual Teradata user, role, profile, and IP filter objects. Valid containers are: <ul style="list-style-type: none"> <li>• cn=users</li> <li>• cn=profiles</li> <li>• cn=roles</li> <li>• cn=ipfilters (optional)</li> </ul>	Required (except as noted)
tdatUser	Describes a Vantage user, where <i>cn=user_name</i> .	
tdatRole	Describes a Vantage role, where <i>cn=role_name</i> .	
tdatProfile	Describes a Vantage profile, where <i>cn=profile_name</i> .	
tdatIPFilter	Describes a Vantage IP filter, where <i>cn=filter_name</i> .	Optional

## Teradata Schema Object Attributes in the Directory Information Tree

The Teradata extensions to the directory schema include attributes that Teradata schema objects can or must contain:

- Required: Attributes that must appear in the objects that can contain them.
- Optional: Attributes that may be required if certain conditions are present.
- Generated: Attributes automatically generated by Active Directory, ADAM, and AD LDS.

Attribute Name	Description	Occurrence	Directory
cn	The common name of the object.	Required. One occurrence per tdat object.	All directories
description	A description of the object, how it is used, or other wording to help place the object within its overall context.	Optional. No limit on occurrences.	

Attribute Name	Description	Occurrence	Directory
tdatUserMember	FQDN of a directory user that maps to the Vantage User named in the cn attribute of the tdatUser object.	Required to map directory users to tdatUser objects. One or more occurrences per mapped object.	All Directories
tdatRoleMember	FQDN of a directory group that maps to the Vantage role named in the cn attribute of the tdatRole object.	Required to map directory groups to tdatRole objects. One or more occurrences per mapped object.	
tdatProfileMember	FQDN of a directory profile that maps to the Vantage profile named in the cn attribute of the tdatProfile object.	Required to map directory users to tdatProfile objects. One or more occurrences per mapped object.	
tdatAllowDeny	This attribute is a boolean switch in an tdatIPFilter object. When set to TRUE, the IP filter is a restrictive filter. When set to FALSE, the filter is a permissive filter.	Required to define the type of IP filter. One occurrence per object.	
tdatAllowedIP	Each attributes contains an IP address and a mask, which define filter criteria. In a restrictive filter: <ul style="list-style-type: none"><li>• Use the tdatAllowIP attribute to specify the range of IP addresses allowed to log on to the database.</li><li>• Use the tdatDenyIP to define exceptions to the IP range allowed by the tdatAllowIP.</li></ul> In a permissive filter: <ul style="list-style-type: none"><li>• Use the tdatDenyIP attribute to specify the range of IP addresses denied permission to log on to the database.</li><li>• Use the tdatAllowIP to define exceptions to the IP range denied by the tdatDenyIP.</li></ul>	Required. A tdatIPFilter must contain at least the primary attribute for the filter type. For information creating IP filters, see <a href="#">About IP Filters</a> .	
tdatDeniedIP			
tdatIPFilterMember	FQDN of a directory profile that maps to the Vantage profile named in the cn attribute of the tdatProfile object.	Required to map directory users to tdatIPFilter objects. One or more occurrence per mapped object.	



Attribute Name	Description	Occurrence	Directory
tdatIPFilterMemberOf	The FQDN of an IP filter named in an <i>ipFilters</i> object.	Generated. For further information on generated objects and attributes, see <a href="#">Special Objects and Attributes Required for Active Directory, ADAM, and AD LDS</a> .	Active Directory, ADAM, or AD LDS only
tdatUserMemberOf	FQDN of a Vantage user in an Active Directory, ADAM, or AD LDS <i>user</i> object.		
tdatRoleMemberOf	FQDN of a Vantage role in an Active Directory, ADAM, or AD LDS <i>group</i> object.		
tdatProfileMemberOf	FQDN of a Vantage profile in an Active Directory, ADAM, or AD LDS <i>user</i> object.		

## Special Objects and Attributes Required for Active Directory, ADAM, and AD LDS

To fully utilize the objects in the Teradata schema extensions, Active Directory, ADAM, and AD LDS automatically generate three additional objects, along with associated attributes and values, when you install Teradata schema extensions in the directory.

Object	Related Attribute
tdatUserExt	Optional for: <ul style="list-style-type: none"> <li>• tdatUserMemberOf</li> <li>• tdatProfileMemberOf</li> </ul>
tdatGroupExt	Optional for tdatRoleMemberOf
tdatIPFilterExt	Optional for tdatIPFilterMemberOf

The attributes of these special Active Directory/ADAM/AD LDS objects are linked to other attributes common to all directories.

This common attribute...	Links to this special Active Directory, ADAM, or AD LDS attribute...
tdatUserMember	tdatUserMemberOf
tdatRoleMember	tdatRoleMemberOf
tdatProfileMember	tdatProfileMemberOf
tdatIPFilterMember	tdatIPFilterMemberOf

When you map a Teradata Vantage user to a directory user by adding a tdatUserMember attribute to the tdatUser object, you must set the value of the tdatUserMember attribute to the FQDN of the directory user.

Because the two attributes are linked, the directory automatically creates a `tdatUserMemberOf` attribute in the directory user object, which points back to the `tdatUser` object.

Mapping of `tdatProfile` objects to users and `tdatRole` objects to groups is similar, in that it requires setting a value for the `tdatProfileMember` and `tdatRolemember` attributes.

Removing values from the member attributes also has some automatic consequences in Active Directory, ADAM, and AD LDS, for example:

- When you remove a `tdatUserMember` attribute from a `tdatUser` object, the directory automatically removes the corresponding `tdatUserMemberOf` attribute.
- If you remove a user from the directory, the directory automatically removes the corresponding `tdatMember` attributes from any objects mapped to the user.

## Installing Teradata Schema Extensions in a Certified Directory

You must install Teradata schema extensions in a directory before you can configure the directory to manage Teradata Vantage users. The schema extensions allow the directory to recognize the Teradata directory objects that must be added to the directory information tree as part of directory user provisioning.

### Note:

When you install Teradata schema extensions on Active Directory, ADAM, or AD LDS, and then make associated directory configuration changes, you cannot undo the changes without re-initializing the directory server. Be sure that you fully understand the content of the schema extensions and the consequences of using them before proceeding.

Note also that Active Directory, ADAM, and AD LDS use a built-in replication feature. If your Vantage system is configured with multiple Active Directory, ADAM, or AD LDS directory servers, Teradata schema extensions and configurations on one directory server automatically replicate to all other directory servers in the domain or forest of domains. You cannot undo replicated changes without re-initializing all directory servers and reloading the entire forest of domains.

## About Schema Installation Options

The Teradata GSS files contain two types of Teradata directory schema extensions for each supported directory type.

Schema Type	Description
<code>tdat.&lt; dir type &gt;.schema</code>	The base schema. Allows you to create Teradata directory objects and map them to directory users.
<code>ipfilter.&lt; dir type &gt;.schema</code>	Optional IP restriction schema. Allows you to create IP access restrictions and map them to directory users.

Schema Type	Description
	<b>Note:</b> You cannot use the IP restriction schema unless you install the base schema.
policy.<dir type>.schema	Optional security policy schema. Allows you to create security policies and policy elements, and to assign the policies to users.  <b>Note:</b> You cannot use the Teradata security policy schema unless you install the Teradata base schema.
ipnetwork.adam.schema	Optional ipNetwork schema. Used only to assign security policy by IP address range, and only on ADAM or AD LDS directories. See <a href="#">Configuring ipNetworks and Network Groups</a> .  <b>Note:</b> This schema does not depend on Teradata base schema and can be used with native directory schema implementations.

## Installing Schema Extensions on Active Directory, ADAM, or AD LDS

You can use a script to install Teradata schema extensions from a Teradata Vantage node into Active Directory, ADAM, or AD LDS. The TDGSS/bin directory contains the ldapmodify tool. Use the schema load script (designated by the - lines in the following example) to install the schema from Vantage.

### Procedure

1. From the TDGSS/bin directory, use ldapmodify to run the schema installation script:

```
#!/bin/sh
#
# usage: loadschema server
#
#
- if [ $# != 1 ]; then
-   echo "Wrong # args"
-   echo "usage: $0 server"
-   exit 1
- fi
- cd /opt/teradata/tdgss/etc
- SNC='ldapsearch -H ldap://$1 -b "" -s base schemanamingcontext | \
-   grep -i schemanamingcontext | \
-   cut -d' ' -f2'
- if [ "$SNC" = "" ]; then
-   echo "Schema naming context not found on server $1"
-   exit 1
- fi
- cat tdat.actdir.schema ipfilter.actdir.schema policy.actdir.schema | \
```

```
- sed -e "s/CN=Schema/$SNC/" | \
- ldapmodify -c -H ldap://$1 -x -D admin_DN -W -Z
```

where the ldapmodify syntax operates as follows:

Syntax Element	Explanation
server	Names an Active Directory, ADAM, or AD LDS directory server where the schema extensions are loaded.
-c	Causes ldapmodify to ignore errors and keep running.
-H ldap://\$1	Specifies the ldap server naming convention according to binding type: <ul style="list-style-type: none"> <li>For TLS protection (requires concurrent use of the -Z option): ldap://server/</li> </ul>
-x	Requests simple binding.
-D admin_DN	Specifies the DN of a user with administrative privileges in the directory.
-W	Causes ldapmodify to prompt for the password of the user identified in -D
-Z	Requests TLS protection and requires a successful response before continuing.

- The system prompts for the directory password of the user running the ldapmodify command. Enter the password.

You can use the script as shown above based on the following assumptions:

- The Teradata GSS server package, which includes the tdat.actdir.schema file, has already been installed on the Teradata Vantage nodes.
- The administrator specified in the script must have the required access privileges.
- Active Directory/ADAM is running on Windows 2003 or later.
- If you have already installed the base schema and only want to add the IP restriction schema or the security policy schema, omit the tdat.actdir.schema.

Do the following to use the script to install schema from Teradata Vantage to Active Directory, ADAM, or AD LDS running on the system, for example, system esroot:

- From the Vantage command prompt, after pasting in the install script, run the script by entering:

```
./loadschema esroot
```

- The administrator is prompted for a password with the following:

```
Please enter your password:
```

#### Note:

With simple binding you must include a -W specification to initiate a password prompt. The password you submit must be the correct password for the username shown in the script.

- The system then returns the following output, showing that the Teradata schema extensions are installed in the directory:

**Note:**

The output shown below is not complete. It was edited to provide a brief example of what you would see at the completion of schema installation.

```
adding entry "cn=tdatProfileMember,CN=Schema,CN=Configuration,
DC=esrootdom,DC=esdev,DC=tdat"
adding entry "cn=tdatProfileMemberOf,CN=Schema,CN=Configuration,
DC=esrootdom,DC=esdev,DC=tdat"
...snipped...
adding entry "cn=tdatUser,CN=Schema,CN=Configuration,
DC=esrootdom,DC=esdev,DC=tdat"
adding entry "cn=tdatRole,CN=Schema,CN=Configuration,
DC=esrootdom,DC=esdev,DC=tdat"
modifying entry ""
$
```

## Installing Schema Extensions on Sun Java System Directory Server

- On the Teradata Vantage server, navigate to TDGSS/etc.
- Retrieve the needed schema files.
- Copy the schema extension files you want to install into the directory server schema directory:
  - Copy the tdat.sunone.schema file as 75tdat.ldif
  - Copy the ipfilter.sunone.schema file as 75tdat01.ldif (optional)

**Note:**

To locate the schema directory, see the *Sun Java System Directory Server Administration Guide*.

- Restart the directory server.

## Installing Schema Extensions on Novell eDirectory

- On the Teradata Vantage server, navigate to TDGSS/etc.
- Run the ldapmodify utility, bundled with TDGSS, to install the Teradata schema extensions on a directory server running Novell eDirectory.

**Note:**

Specify simple binding for executing the `ldapmodify` command, and also the recommended TLS protection, to ensure a secure and successful schema installation. You may need to do additional configuration of TLS on the computer containing the schema extension files to ensure presence of the certificate chain. For information, see [Using TLS with a Directory Server](#).

Customize the `ldapmodify` command shown below to install the Teradata schema extension files you need, based on the protection scheme and schema file name. Install one schema file for each command.

For example, to install the main Teradata schema extension file:

- With a connection to the directory server that uses simple binding and TLS protection:

```
../bin/ldapmodify -x -D admin_DN -W -H ldap://dir_server -Z
-f tdat.edir.schema
```

- With a connection to the directory server without protection, that is, in plain text (not recommended):

```
../bin/ldapmodify -x -D admin_DN -W -H ldap://dir_server
-f tdat.edir.schema
```

**Note:**

Installation of other schema extensions is similar.

**-x**

Specifies simple binding.

**-D admin\_DN**

Specifies the DN of a user with administrative privileges in the directory.

**-W**

Causes `ldapmodify` to prompt for the password of the user identified in `-D`.

**-H**

Specifies the ldap server naming convention according to binding type:

- For TLS protection (requires concurrent use of the `-Z` option):

`ldap://server/`

**-Z**

Requests TLS protection and requires a successful response before continuing.

**-f**Specifies the name of the schema extension file, for example, `tdat.edir.schema` (base schema).**Note:**

Like Active Directory, eDirectory uses dynamic schema updates, so you do not have to restart the system after installation of the schema extensions. eDirectory also automatically updates all directories in a replicated environment.

## Installing Schema Extensions on OpenLDAP

1. On the Teradata Vantage server, navigate to `TDGSS/etc` and copy the `tdat.openldap.schema`, and if required, the `ipfilter.openldap.schema` files.
2. Using `vi` or other text editor, copy the file(s) into the OpenLDAP schema directory, usually located at `/etc/openldap/schema`, as follows:

**Note:**

Details of the include command shown below are based on OpenLdap running on SUSE Linux Enterprise Server 10. Refer to OpenLdap documentation for use of this procedure when OpenLdap runs on other operating systems.

```
vi /etc/openldap/slapd.conf
include /etc/openldap/schema/tdat.openldap.schema
include /etc/openldap/schema/ipfilter.openldap.schema
include /etc/openldap/schema/policy.openldap.schema
```

**Note:**

You only need to include the schema extensions you plan to use.

3. Restart the OpenLDAP server `slapd` daemon:

```
/etc/init.d/ldap restart
```

or

```
/etc/init.d/ldap stop
/etc/init.d/ldap start
```

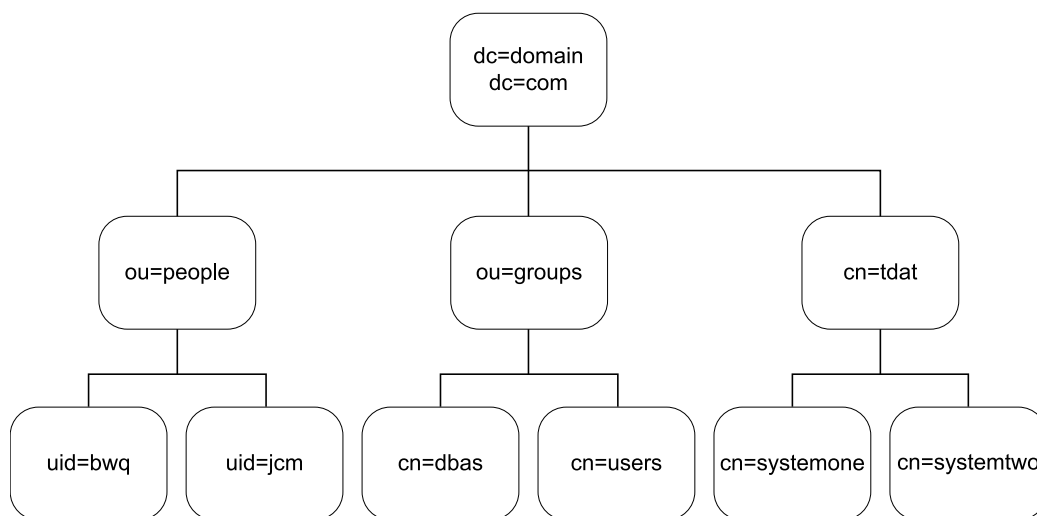
4. Repeat steps 2 and 3 for each server replica that exists.

**Note:**

Do not add any Teradata objects to the directory until you install copies of the Teradata schema extensions on all servers, and perform the required restart, or the replication of Teradata objects may fail.

## About Teradata Schema Objects in the DIT

The directory information tree (DIT) hierarchical representation of directory objects.



## Teradata Schema Objects in the DIT Hierarchy

Before a supported directory can authenticate and authorize user access to Teradata Vantage, you must create Teradata schema objects in the DIT, and arrange them in the required hierarchical relationship to other directory objects.

**Note:**

You can place the tdatRootNode object, labeled cn=tdat, anywhere in the hierarchy, but you must locate all child objects exactly as shown in diagram.

where:



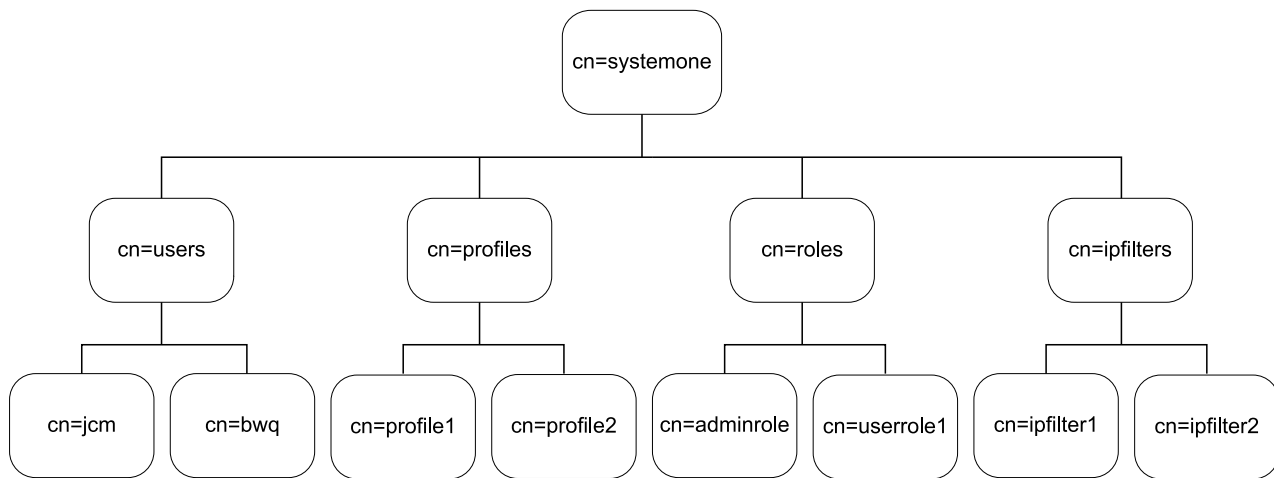
Row	DN	Object Class	Description	Defined by Object Class
Top	dc=domain, dc=com	dcObject	Top-level directory object	Standard directory schema
Middle	ou=people	organizationalUnit	Directory object that contains individual person objects	
	ou=groups	organizationalUnit	Directory object that contains user and database objects	
	cn=tdat	tdatRootNode	Teradata Root Node object. <b>Note:</b> This is the suggested location for the Root Node object. However, you can locate it elsewhere in the directory.	Teradata directory schema extensions
Bottom	cn=systemone	tdatSystem	Directory object that defines Teradata Vantage system 1.	Teradata directory schema extension
	cn=systemtwo	tdatSystem	Teradata object that defines Teradata Vantage system 2.	
	cn=administrators	groupOfNames	Directory object that defines the administrator group.	Standard directory schema
	cn=dbas	groupOfNames	Directory object that defines the group dbas. Enter the DN of each directory user in the groupOfNames member attribute, to give them group membership.	
	uid=bwq	person	Directory objects that define individual directory users.	
	uid=jcm	person		

## About Teradata Security Policy Objects

When configuring security policy, you should use a separate tdatRootNode object (cn=tdat) to be the parent object for Teradata security policy objects. For more information see [Configuring Top-Level Security Policy Objects](#).

## Lower-level Teradata Schema Objects in the DIT Hierarchy

The following diagram shows the tree of lower-level Teradata directory objects that are children of the cn=systemone object shown in [Teradata Schema Objects in the DIT Hierarchy](#). The overall structure of the DIT implies that a similar tree also exists under the systemtwo object.

**Note:**

The structure of the sub-trees contained by the tdatRootNode (cn=tdat) is rigid and must appear in the directory exactly as shown.

where:

Row	DN	Object Class	Description
Top	cn=systemone	tdatSystem	The name of a Teradata Vantage system to which directory users can connect.
Middle	cn=users	tdatContainer	Teradata directory object that contains user objects
	cn=roles		Teradata directory object that contains role objects
	cn=profiles		Teradata directory object that contains profile objects
	cn=ipfilters		Teradata directory object that contains IP filter objects
Bottom	cn=bwq	tdatUser	Vantage username
	cn=jcm		
	cn=profile1	tdatProfile	Vantage profile name
	cn=profile2	tdatProfile	Vantage profile name
	cn=adminrole	tdatRole	Vantage external role name
	cn=userrole1		
Bottom (continued)	cn=ipfilter1	tdatIPFilter	IP restriction filter name

Row	DN	Object Class	Description
	cn=ipfilter2		

**Note:**

The Teradata directory schema extensions define the object class.

## Creating the Top-Level Objects in the DIT

Before creating Vantage user, role, profile, and IP filter objects in the directory, you must create the top-level Teradata directory objects, tdatRootNode and tdatSystem.

### Object Creation Process

1. Create directory objects in a text editor, such as vi or Notepad, using the example syntax.
2. Use an LDIF-compatible directory tool to add the objects to the directory.

For Directories that Run on...	Use...
Windows	LDIFDE
systems based on the UNIX operating system	<ul style="list-style-type: none"> <li>• ldapadd</li> <li>• ldapmodify</li> </ul>

See [About Other LDAP Tools](#).

3. Restart the directory to enable the changes and replicate them to any other directory instances in the domain.

### Creating the tdatRootNode Object

The tdatRootNode is the primary Teradata object to be defined in the DIT. It contains all other Teradata directory objects.

#### Example: tdatRootNode

Create the tdatRootNode object using the following syntax:

```
dn: cn=tdat,dc=domain,dc=com
objectClass: top
objectClass: tdatRootNode
cn: tdat
```

## Creating the tdatSystem Object

Each tdatSystem object is the parent of a structure of lower level containers and database objects. Mapping a directory user to one of these objects defines a directory user privilege in the database. The LdapSystemFQDN property on a database system or Unity server points to the tdatSystem object that defines LDAP authorizations for directory users who use that database access point.

### Example: tdatSystem

Create tdatSystem objects using the following information in the DIT:

```
dn: cn=systemone,cn=tdat,dc=domain,dc=com
objectClass: top
objectClass: tdatSystem
cn: systemone
```

## Configuring tdatSystem Objects

The requirements for tdatSystem objects in the directory varies, depending on the number and configuration of Teradata Vantage systems:

- If the directory serves a single Vantage system, create a single tdatSystem object, as shown in [Example: tdatSystem](#).
- If the directory serves multiple Vantage systems:
  - If the systems maintain the same set of users, profiles, roles and IP filters, and maps them to directory users in the same manner, you only need a single tdatSystem object for all systems.
  - If the users, profiles, roles, and IP filters diverge among multiple database systems, then you should create a separate tdatSystem object for each Vantage system.
  - If directory users for multiple Vantage systems log on through Unity, Teradata recommends that the LdapSystemFQDN property on all Vantage systems and connected Unity servers use the same authorization structure. If all Unity managed systems authenticate to the same directory, they should point to the same tdatSystem object. If they authenticate to different directories, the mappings to child objects for the tdatSystem object in each directory should be the same in all directories. See *Teradata® Unity™ Installation, Configuration, and Upgrade Guide for Customers*, B035-2523 and *Teradata® Unity™ User Guide*, B035-2520.

## Creating Containers and Inserting Objects

You must create tdatContainer objects to hold Teradata schema extension user, role, profile, and ipfilter objects, and then create the individual objects and insert them into the matching containers. Be sure to save the containers and objects you create.

**Note:**

For information on creating security policy containers and objects, see [Network Security Policy](#).

## Container and Object Creation Process

1. Create container objects and their child objects in a text editor such as vi or Notepad.
2. Use a directory tool, for example, LDIFDE, ldapadd, or ldapmodify to add the objects to the directory.
3. Restart the directory to enable the changes and replicate them to any other directory instances in the domain.

## About Naming Conventions

Object Class	Naming Convention
tdatContainer	<p>The distinguished name (dn) of a tdatContainer includes the:</p> <ul style="list-style-type: none"> <li>• Common name (cn) of the container based on the type of objects it contains, that is, users, roles, profiles, or ipfilters.</li> <li>• DN of the tdatSystem object to which it applies. See <a href="#">Configuring tdatSystem Objects</a>.</li> </ul>
<ul style="list-style-type: none"> <li>• tdatUser</li> <li>• tdatRole</li> <li>• tdatProfile</li> <li>• tdatIPFilter</li> </ul>	<p>The DN for child objects of a tdatContainer include the:</p> <ul style="list-style-type: none"> <li>• CN of a Vantage user, external role, profile, or IP filter</li> <li>• CN of the object type, that is, users, roles, profiles, or IP filters</li> <li>• DN of the tdatSystem object to which it applies</li> </ul>

## Creating Users Containers and Inserting User Objects

### Example: tdatContainer (Users)

Use the following syntax to construct a user's container to hold individual user objects:

```
dn: cn=users,cn=system1,cn=tdat,dc=domain,dc=com
objectClass: top
objectClass: tdatContainer
cn: users
```

### Example: tdatUser

Use the following syntax to create a user object that matches a Vantage username:

```
dn: cn=jcm,cn=users,cn=system1,cn=tdat,dc=domain,dc=com
objectClass: top
objectClass: tdatUser
cn: jcm
```

## Creating Roles Containers and Inserting Role Objects

### Example: tdatContainer (Roles)

Use the following syntax to construct a roles container to contain tdatRole objects:

```
dn: cn=roles,cn=system1,cn=tdat,dc=domain,dc=com
objectClass: top
objectClass: tdatContainer
cn: roles
```

### Examples: tdatRole

Use the following syntax to create the tdatRole objects, administrator and user, which correspond to Vantage external role names.

```
dn: cn=adminrole,cn=roles,cn=system1,cn=tdat,dc=domain,dc=com
objectClass: top
objectClass: tdatRole
cn: adminrole

dn: cn=userrole,cn=roles,cn=system1,cn=tdat,dc=domain,dc=com
objectClass: top
objectClass: tdatRole
cn: userrole
```

## Creating Profiles Containers and Inserting Profile Objects

### Example: tdatContainer (Profiles)

Use the following syntax to construct a profiles container in the directory to contain tdatProfile objects.

```
dn: cn=profiles,cn=system1,cn=tdat,dc=teradata,dc=com
objectClass: top
```

```
objectClass: tdatContainer
cn: profiles
```

## Examples: tdatProfile

Use the following syntax to create tdatProfile objects that correspond to Vantage profile names.

```
dn: cn=profile1,cn=profiles,cn=system1,cn=tdat,dc=domain,dc=com
objectClass: top
objectClass: tdatProfile
cn: profile1

dn: cn=profile2,cn=profiles,cn=system1,cn=tdat,dc=domain,dc=com
objectClass: top
objectClass: tdatProfile
cn: profile2
```

## Creating IP Filters Containers and Inserting IP Filters

You can use IP filters to allow or deny user access to Teradata Vantage from particular IP addresses. Make sure you are familiar with the function of allow and deny type IP filters before configuring tdatIPFilter objects in the directory. For information, see [Restricting Logons by IP Address](#).

### Note:

If directory users log on through Unity, you must configure the same IP restrictions for each database system managed by Unity. IP restrictions are not configured on Unity, which only passes the user IP address to Teradata Vantage.

IP Filter Type and Setting	Function
Restrictive Set tdatAllowDeny to TRUE	Denies all IP addresses except those specifically allowed.
Permissive Set tdatAllowDeny to FALSE	Allows all IP addresses except those specifically denied.

## Example: tdatContainer (IPFilters)

The following syntax creates an ipfilters container in the directory to hold tdatIPFilter objects.

```
dn: cn=ipfilters,cn=systemone,cn=tdat,dc=teradata,dc=com
objectClass: top
objectClass: tdatContainer
cn: ipfilters
```

## Examples: tdatIPFilter

Use the following syntax to create a tdatIPFilter object.

### Note:

Also see [Applying IPFilters to Directory Users](#).

```
dn: cn=filter1,cn=ipfilters,cn=system1,cn=tdat,dc=domain,dc=com
objectClass: top
objectClass: tdatIPFilter
tdatAllowDeny: TRUE
tdatAllowedIP: 141.206.0.0/16
tdatDeniedIP: 141.206.35.0/24
cn: filter1
```

```
dn: cn=filter2,cn=ipfilters,cn=system1,cn=tdat,dc=domain,dc=com
objectClass: top
objectClass: tdatIPFilter
tdatAllowDeny: FALSE
tdatAllowedIP: 141.206.35.175/32
tdatDeniedIP: 141.206.35.0/24
cn: filter2
```

## Applying IPFilters to Directory Users

You cannot map an IP filter to a directory user. To apply IP filter restrictions to a directory user, you must:

1. In a tdatIPFilterMember attribute for the filter, enter the DN of one of the following:
  - tdatUser object to which the directory user is mapped
  - the tdatUsers container object (applies the filter to all tdatUser objects)
2. Directory users inherit the IP restrictions that apply to the tdatUser object to which they are mapped. If a directory user is not mapped to a tdatUser object, no IP restrictions apply. See [Mapping Directory Users to Vantage Users](#).



**Note:**

Directory-based IP filters have no effect until you enable them in the system. See [Enabling Directory-Based IP Restrictions with the ipdir2bin Utility](#).

**Example: Applying an IP Filter to tdatUser Objects**

```
dn: cn=filter1,cn=ipfilters,cn=system1,cn=tdat,dc=domain,dc=com
changeType: modify
add: tdatIPFilterMember
tdatIPFilterMember:
  cn=jcm,cn=users,cn=system1,cn=tdat,cn=domain,cn=com

dn: cn=filter2,cn=ipfilters,cn=system1,cn=tdat,dc=domain,dc=com
changeType: modify
add: tdatIPFilterMember
tdatIPFilterMember:
  cn=bwq,cn=users,cn=system1,cn=tdat,cn=domain,cn=com
```

**Example: Applying an IP Filter to All Vantage Users**

```
dn: cn=filter2,cn=ipfilters,cn=system1,cn=tdat,dc=domain,dc=com
changeType: modify
add: tdatIPFilterMember
tdatIPFilterMember:
  cn=users,cn=system1,cn=tdat,dc=domain,dc=com
```

**Mapping Directory Users to Teradata Vantage Objects****Mapping Directory Users to Vantage Users**

To map a directory user to a Vantage user, modify the tdatUser object for the Vantage user, shown as dn: cn=jcm..., to add a tdatUserMember attribute that represents the directory user.

**Note:**

The directory user inherits all privileges and IP restrictions that apply to the mapped Vantage user.

**Example: Mapping a Directory User to a tdatUser**

```
dn: cn=jcm,cn=users,cn=system1,cn=tdat,dc=domain,dc=com
changeType: modify
add: tdatUserMember
tdatUserMember: uid=dirUser1,ou=people,dc=domain,dc=com
```

**Mapping Directory Users to Vantage External Roles**

To map a directory group object (a group of users) to a Vantage external role, place the FQDN of the group object in a tdatRoleMember attribute for a tdatRole object.

**Examples: Mapping a Directory Group to a tdatRole**

```
dn: cn=adminrole,cn=roles,cn=system1,cn=tdat,dc=domain,dc=com
changeType: modify
add: tdatRoleMember
tdatRoleMember: cn=dbas,ou=groups,dc=domain,dc=com
```

```
dn: cn=userrole1,cn=roles,cn=system1,cn=tdat,dc=domain,dc=com
changeType: modify
add: tdatRoleMember
tdatRoleMember: cn=endusers,ou=groups,dc=domain,dc=com
```

**Mapping Directory Users to Vantage Profiles**

To map a directory user to a Vantage profile, place the FQDN of the directory user in a tdatProfileMember attribute of a tdatProfile object.

**Examples: Mapping a Directory User to a tdatProfile**

```
dn: cn=profile1,cn=profiles,cn=system1,cn=tdat,dc=domain,dc=com
changeType: modify
add: tdatProfileMember
tdatProfileMember: uid=jcm,ou=people,dc=domain,dc=com

dn: cn=profile2,cn=profiles,cn=system1,cn=tdat,dc=domain,dc=com
changeType: modify
```

```
add: tdatProfileMember  
tdatProfileMember: uid=bwq,ou=people,cn=domain,cn=com
```

# Using Native Directory Schema to Provision Directory Users

You can manage Teradata Vantage users in a supported directory, without installing Teradata schema extensions, using only native directory schema and tools. This method is required for uncertified LDAP-compliant directories. See [About Certified Directories](#).

---

**Note:**

Restricting database access by user IP address (see [Restricting Logons by IP Address](#)) is not available when using only native schema to provision directory users. If you want to use IP address restriction, you must install and use Teradata schema extensions in the directory. See [Provisioning Directory Users with Teradata Schema Extensions](#).

---

## Process Overview

1. Review [About Teradata Schema Objects in the DIT](#) to become familiar with the hierarchical relationship among Teradata directory objects. Some objects have different object classes and DNs in the native directory schema environment, but the structure is similar.
2. Review the database objects that must exist in the directory, including the required hierarchy. See [About Required Teradata Objects](#).
3. Create the top level database objects in the directory. See [Creating the RootNode and System Objects](#).
4. Create containers for Teradata objects in the directory, then create Teradata Vantage user, role, and profile objects and insert them into the corresponding containers. See [Creating Containers and Inserting Database Objects](#).
5. Map directory users to Teradata Vantage user, role, and profile objects. See [Mapping Additional Directory Users to Vantage User, Role, and Profile Objects](#).

---

**Note:**

For information on using native directory schema to create security policies and assign them to users, see [Network Security Policy](#).

---

## Prerequisites

- Evaluate the readiness of the directory for management of Teradata Vantage users. See [Evaluating the System for Directory Management of Users](#).
- Develop a directory user management strategy. See [Working with Directory User Management Options](#).
- Create Vantage users, external roles, and profiles in the database before creating their counterparts in the directory. See [Creating Users and Granting Privileges](#).

## About Required Teradata Objects

The following table shows the Teradata directory objects required to manage database users, when using only native schema.

Teradata Object	Directory Object Type and Description	Class
RootNode	An organizationalUnit object that describes the parent object for all Teradata System objects. The ou attribute and its value must be used as the relative distinguished name (RDN).	Required
System	An organizationalUnit object that describes the parent object for a set of Teradata objects. ou=system name The system name can be a Teradata Vantage system or a Unity server. <b>Note:</b> The value of the LdapSystemFQDN mechanism property for a system must specify the FQDN of this object.	
Container	A groupOfNames object. Teradata user, role, or profile objects each require a separate container. <ul style="list-style-type: none"> <li>• ou=users</li> <li>• ou=profiles</li> <li>• ou=roles</li> </ul>	
User	A groupOfNames object that describes a Vantage user. cn=a database user name	Optional Directory users whose database privileges are authorized by the directory have only the privileges of the database objects to which they are mapped.
Role	A groupOfNames object that describes a Vantage external role. cn=a database external role name	
Profile	A groupOfNames object that describes a Vantage profile. cn=a database profile name	

## Creating the RootNode and System Objects

You must create the top-level Teradata root node and system objects to establish the hierarchical relationship between Teradata objects and other directory objects.

## RootNode Object

The RootNode is the primary Teradata object in the DIT. It contains all other Teradata directory objects.

### Example: Creating the RootNode Object

Create the RootNode object by entering the following information in the DIT:

```
dn: ou=tdat,dc=domain,dc=com
objectClass: top
objectClass: organizationalUnit
ou: tdat
```

## System Object

You must create a system object for each Teradata Vantage system that directory users access.

### Example: Creating a System Object

Create System objects by entering the following information in the DIT.

```
dn: ou=system1,ou=tdat,dc=domain,dc=com
objectClass: top
objectClass: organizationalUnit
ou: systemone
```

## Configuring System Objects

The requirements for System objects in the directory varies, depending on the configuration of Teradata Vantage systems:

- If the directory serves a single Teradata Vantage system, create a single System object.
- If the directory serves multiple Vantage systems:
  - If all systems maintain the same set of users, profiles, roles and IP filters, and they are mapped to directory users in the same manner, you only need a single System object for all systems.
  - If the users, profiles, roles, and IP filters diverge among database systems, then you should create a separate System object for each Vantage system.
  - If directory users for multiple Vantage systems log on through Unity, Teradata recommends that the LdapSystemFQDN property on all Vantage systems and connected Unity servers use the same authorization structure. If all Unity managed systems authenticate to the same directory,

they should point to the same tdatSystem object. If they authenticate to different directories, the mappings to child objects for the tdatSystem object in each directory should be the same in all directories. See *Teradata® Unity™ Installation, Configuration, and Upgrade Guide for Customers*, B035-2523 and *Teradata® Unity™ User Guide*, B035-2520.

## Creating Containers and Inserting Database Objects

Individual Teradata user, role, and profile objects can only exist in the directory in container objects. Create the required containers and then insert Teradata user, role, and profiles objects.

---

### Note:

Be sure to save all new directory entries.

---

## Users

### Example: Creating a Users Container

Use the following syntax to create a users container in the directory, to hold user objects.

```
dn: ou=users,ou=system1,ou=tdat,dc=domain,dc=com
objectClass: top
objectClass: organizationalUnit
ou: users
```

### Example: Inserting User Objects into a Users Container

Use the following syntax to create user objects that represent Teradata Vantage users, and insert them into the appropriate users container.

```
dn: cn=dbuserX,ou=users,ou=system1,ou=tdat,dc=domain,dc=com
objectClass: top
objectClass: groupOfNames
cn: dbuserX
member: directory_user_dn
-
```

The cn for each user object must match the name of a Vantage user.

You must specify a member attribute and value when the user object is created.

*directory\_user\_dn* specifies a directory user that you want to map to the database user object, for example:

```
member: uid=dirUser1,ou=people,dc=domain,dc=com
```

## Roles

### Example: Creating a Roles Container

Use the following syntax to construct a roles container in the directory to contain role objects.

```
dn: ou=roles,ou=system1,ou=tdat,dc=domain,dc=com
objectClass: top
objectClass: organizationalUnit
ou: roles
```

### Examples: Inserting Role Objects into a Roles Container

Use the following syntax to create role objects that represent Teradata Vantage external roles and insert them into the appropriate roles container.

```
dn: cn=adminrole,ou=roles,ou=system1,ou=tdat,dc=domain,dc=com
objectClass: top
objectClass: groupOfNames
cn: adminrole
member: group_dn
-
cn=userrole,ou=roles,ou=system1,ou=tdat,dc=domain,dc=com
objectClass: top
objectClass: groupOfNames
cn: userrole
member: group_dn
-
```

The cn for each role object must match the name of a Vantage external role.

At least one member attribute and associated value is required when the role object is created.

*group\_dn* is either:

- The dn of the roles container, if no individual role members are known
- The dn of a directory group that you want to map to the role, for example:



```
member: cn=administrators,ou=groups,dc=domain,dc=com
```

## Profiles

### Example: Creating a Profiles Container

The following syntax constructs a profiles container in the directory to contain profile objects.

```
dn: ou=profiles,ou=system1,ou=tdat,dc=teradata,dc=com
objectClass: top
objectClass: organizationalUnit
ou: profiles
```

### Examples: Inserting Profile Objects into a Profiles Container

Use the following syntax to create profile objects that represent Teradata Vantage profiles and insert them into the appropriate profiles container.

```
dn: cn=profile1,ou=profiles,ou=system1,ou=tdat,dc=domain,dc=com
objectClass: top
objectClass: groupOfNames
cn: profile1
member: directory_user_dn
-
dn: cn=profile2,ou=profiles,ou=system1,ou=tdat,dc=domain,dc=com
objectClass: top
objectClass: groupOfNames
cn: profile2
member: directory_user_dn
-
```

The cn for each profile object must match the name of a Vantage profile.

At least one member attribute and associated value is required when the profile object is created.

*directory\_user\_dn* is either:

- The dn of the profiles container, if no individual members are known.
- The dn of a user that is a member of the profile. You must use a separate member attribute for each profile member.

```
member: uid=dirUser1,ou=principals,dc=domain,dc=com
```

**Note:**

You must use a separate member attribute for each profile member

## Mapping Additional Directory Users to Vantage User, Role, and Profile Objects

After you create the needed containers and Teradata objects in the directory, you can map directory users to the objects that have privileges they need.

You must use a separate member attribute for each mapping, but there is no limit on the number of members you can add per entry.

## Mapping Directory Users to Database Users

To map a directory user to a database user, place the dn of the directory user in a member attribute for the database user object.

### Example: Mapped User

```
dn: cn=dbuser16,ou=users,cn=system1,cn=tdat,dc=domain,dc=com
changeType: modify
add: member
member: uid=dirUser2,ou=people,dc=domain,dc=com
```

## Mapping Directory Groups to Teradata Vantage External Roles

To map a Teradata Vantage external role to a directory group, specify the dn of the group object in a member attribute of the role object.

### Example: Mapped Roles

```
dn: cn=adminrole,ou=roles,cn=system1,cn=tdat,dc=domain,dc=com
changeType: modify
add: member
member: cn=dbas,ou=groups,dc=domain,dc=com

dn: cn=salesrole,ou=roles,cn=systemone,cn=tdat,dc=domain,dc=com
changeType: modify
add: member
member: cn=salesusers,ou=groups,dc=domain,dc=com
```

## Mapping Directory Users to Vantage Profiles

To map a Teradata Vantage profile to a directory user, place the dn of the user in a member attribute of the profile object.

### Example: Mapped Profiles

```
dn: cn=profile1,ou=profiles,cn=systemone,cn=tdat,dc=domain,dc=com
changeType: modify
add: member
member: uid=dirUser7,ou=people,dc=domain,dc=com
```

```
dn: cn=profile2,ou=profiles,cn=systemone,cn=tdat,dc=domain,dc=com
changeType: modify
add: member
member: uid=dirUser3,ou=people,cn=domain,cn=com
```

# Changing the TDGSS Configuration

You can customize the function of Teradata Vantage security mechanisms to meet site-specific authentication and authorization requirements, by editing the default TDGSS configuration to add mechanism properties or change property values.

## Prerequisites

- Implement a user authentication and authorization strategy. See [Implementing User Authentication and Authorization](#).
- For clients that run Java applications, see *Teradata JDBC Driver Reference*, available at <https://teradata-docs.s3.amazonaws.com/doc/connectivity/jdbc/reference/current/frameset.html>, for Java setup instructions.
- Configuration changes are associated with the implementation of specific security features, as documented in other parts of this publication. Do not initiate a configuration change until you fully understand the reasons for the change, the effects that can result, and all change requirements.
- Users who run configuration change procedures on database nodes must have tdtrusted or similar OS-level privileges. See [Working with OS-Level Security Options](#).

## Configuration Change Process

1. Make sure you understand the format and function of the configuration files. See [About the TDGSS Configuration Files](#).
2. Review the principles of editing and the configuration changes associated with common security strategies. See [About Editing Configuration Files](#).
3. Edit the TdgssUserConfigFile.xml on Teradata Vantage nodes, if required. See [Making Changes to TdgssUserConfigFile.xml on Database Nodes](#).
4. Edit the Unity configuration file TdgssUnityConfig.xml on Unity servers, if required. For information about Unity, see *Teradata® Unity™ Installation, Configuration, and Upgrade Guide for Customers*, B035-2523 and *Teradata® Unity™ User Guide*, B035-2520.
5. If the new configuration does not function as expected, you can return to an older configuration. See [Returning to an Old Configuration](#).

## About the TDGSS Configuration Files

Teradata provides two types of configuration TDGSS files:

- TdgssLibraryConfigFile.xml
- TdgssUserConfigFile.xml

A copy of each file is installed on every Teradata Vantage node as standard part of system implementation. The files are in plain text, stored in XML format.

## User and Library Configuration File Content

TDGSS Configuration File Type	Description
Library file	<p>Contains the default TDGSS configuration for the current Advanced SQL Engine release. The TdgssLibraryConfigFile.xml, defines the security elements, properties, and values available on the system.</p> <p>The TdgssLibraryConfigFile.xml file is located in the /opt/teradata/tdgss/etc directory on Unity and SQL Engine nodes.</p> <p><b>Note:</b> Do not edit the library configuration file.</p>
User file	<p>Contains the system-specific TDGSS configuration, including any user edits. Settings in this file override those in the library configuration file.</p> <p>The TdgssUserConfigFile.xml file is located in the /opt/teradata/tdgss/etc directory on Unity and SQL Engine nodes. Additionally, Unity (if used) copies the TdgssUserConfigFile.xml file to TdgssUnityConfig.xml in /etc/opt/teradata/config/unity and uses that file for its TDGSS configuration file.</p> <p>Editing of this file is permitted, but not required. You can edit the user configuration file to:</p> <ul style="list-style-type: none"> <li>• Control availability of mechanisms to all clients by setting the MechanismEnabled property</li> <li>• Control the source for authorization of user privileges for externally authenticated users by setting the AuthorizationSupported property</li> <li>• Customize LDAP properties, as required for directory user authentication and authorization</li> <li>• Change mechanism property values</li> <li>• Add new mechanisms and properties</li> </ul> <p>Property value settings must be the same for all nodes.</p> <p><b>Note:</b> If you are using Unity you must set up the PROXY mechanism on both Unity servers and the SQL Engine nodes.</p> <p>An upgrade to a new TDGSS release does not overwrite the user configuration file to allow customizations to be retained.</p> <p>For more information, see <a href="#">Changing the TDGSS Configuration</a>.</p>

## About TDGSS Configuration for the Teradata Viewpoint Server

Teradata Viewpoint is the primary Vantage administration tool, which runs as a Java application server. The Viewpoint server requires the same configuration as a Java-enabled Vantage client.

## Related Information

For information on...	See...
Viewpoint setup requirements	<i>Teradata® Unity™ Installation, Configuration, and Upgrade Guide for Customers</i> , B035-2523.

## Using the dumpcfg Utility to Check the Current Configuration

You can use the dumpcfg utility to retrieve the current TDGSS configuration. Run this utility and its options from the Vantage command prompt.

```
dumpcfg [-g TDGSSCONFIG | IPFILTER] [-f bin_file] [-?]
```

### -g TDGSSCONFIG

[Optional] Reads the IP filter GDO and outputs the currently active IP filters.

### -g IPFILTER

[Optional] Reads the tdgssconfig GDO (in the database) and outputs the current content of the TdgssUserConfigFile.xml

### -f bin\_file

[Optional] Reads the .bin file and returns the current TDGSS configuration.

### -?

[Optional] Displays the preceding option descriptions.

## Example: Sample dumpcfg Statements

```
dumpcfg
    (displays TDGSSCONFIG GDO)
dumpcfg -g TDGSSCONFIG
    (displays TDGSSCONFIG GDO)
dumpcfg -g IPFILTER
    (displays IPFILTER GDO)
dumpcfg -f v2r6.2.0.0/etc/tdgssconfig.bin
    (display old style binary configuration file)
```

## Example: Using dumpcfg

The following examples show typical output for *dumpcfg*:

```
dumpcfg
      (displays ./tdgssconfig.bin)

dumpcfg -f v2r6.2.0.0/etc/tdgssconfig.bin
      (displays the specified file)
```

This example of running *dumpcfg* shows partial output for the configuration:

```
dumpcfg
dumpcfg: Reading TDGSSCONFIG GDO file...
Header: Version 2
  48 Elements   at offset      2c (44)
 261 Attributes at offset     710 (1808)
  22 Data items at offset    f40 (3904)
Level 00: TdgssConfigFile (Element 0)
Level 01:   Header (Element 1)
           ATTR: "Version"="1"
           ATTR: "ConfigFileType"="User"
Level 01:   Mechanisms (Element 2)

[...]

Level 02:   Mechanism (Element 7)
           ATTR: "Name"="TD2"
           ATTR: "ObjectId"="1.3.6.1.4.1.191.1.1012.1.1.9"
           ATTR: "LibraryName"="gssp2td2"
           ATTR: "Prefix"="TD2"
           ATTR: "InterfaceType"="teradata"
Level 03:   MechanismProperties (Element 15)
           ATTR: "AuthenticationSupported"="no"
           ATTR: "AuthorizationSupported"="no"
           ATTR: "SingleSignOnSupported"="no"
           ATTR: "DefaultMechanism"="yes"
           ATTR: "MechanismEnabled"="yes"
           ATTR: "MechanismRank"="20"
           ATTR: "DelegateCredentials"="no"
           ATTR: "MutualAuthentication"="yes"
           ATTR: "ReplayDetection"="yes"
           ATTR: "OutOfSequenceDetection"="yes"
```

[illegible]

## About Editing Configuration Files

You can add new mechanisms and properties to the TDGSS user configuration file.

## Adding New Mechanisms and Properties to the TdgssUserConfigFile.xml

You can edit the TDGSS user configuration file to:

- Change the value of a mechanism property in an existing mechanism configuration.
- Add a new property to an existing mechanism, then edit the property value.
- Copy a new mechanism from the library configuration file into the user configuration file, then edit the mechanism property values.

**Note:**

If you can use the default property values for a new mechanism, you do not need to transfer the mechanism into the user configuration file. A mechanism can perform default functions from the library configuration file.

## Effects of Upgrade and Migration on TDGSS Configuration Changes

## On Teradata Vantage Nodes and Unity Servers

Teradata Vantage and Unity Server Upgrade: When you upgrade a database system or Unity server with a new release of Teradata Vantage, TDGSS does not overwrite the TdgssUserConfigFile.xml, but retains the existing configuration.



Teradata Operating System and Hardware Migration: When you migrate Teradata Vantage data to new hardware, or to the same hardware after a change of operating system, the following files and directories are automatically preserved:

- TDGSS site directory and its contents, including the LDAP configuration, if any.

---

**Note:**

If the LDAP configuration and certificates are outside of the TDGSS site directory, the migration script will detect this and preserve the files and the non-standard directory structure.

---

- Certificates for the PROXY mechanism.
  - If Kerberos is in use, the `/etc/krb5.conf` file and the file named in the KRB5 mechanism's `TeradataKeyTab` property are preserved.
- 

**Note:**

The differences between the Unity and Vantage configurations are: Unity uses the `tdgssconfig.bin` file found in the TDGSS `etc` directory, the `TdgssUserConfigFile.xml` is named `TdgssUnityConfig.xml` and located outside of the normal TDGSS configuration at `/etc/opt/teradata/config/unity`, and is managed by Unity independently of the database. For Unity configuration information, see *Teradata® Unity™ Installation, Configuration, and Upgrade Guide for Customers*, B035-2523.

---

## On Teradata Vantage Clients

Teradata recommends not configuring the clients. If configuration is required, an optional Teradata GSS Administrative Package may be installed on the client and configuration may be performed; see [Teradata GSS Administrative Package](#).

## General Rules for Editing the TDGSS Configuration

- You can individually enable or disable authentication mechanisms, but a logon fails if the operant mechanism for a session is disabled on the client, the Unity server (if used) or the database. This is true for both system-selected (default) and user-selected mechanisms.
- Security requirements may vary among Vantage clients. You may find it useful to configure and enable a different set of mechanisms or define a different default mechanism for different clients.
- You can designate only one mechanism as the default mechanism. The system automatically uses the default, so users do not need to specify the mechanism at logon.
- Before you edit the value of a mechanism property, review the editing guidelines for the property. See the topics beginning with [Mechanism Properties](#).
- Most mechanism properties are editable only on Vantage nodes, and on the Unity server, if used.
- On Vantage clients, only the `MechanismEnabled` and `DefaultMechanism` properties can be configured.

- If no mechanism is specified in a user logon, job script, or client application preset, the system uses the first configured DefaultMechanism it finds, in the following order:
  1. Client TdgssUserConfigFile.xml default
  2. Unity (if used) the configured Unity default. For information about the TDGSS configuration on Unity, see *Teradata® Unity™ Installation, Configuration, and Upgrade Guide for Customers*, B035-2523.
  3. Teradata Vantage TdgssUserConfigFile.xml default
- You must edit certain properties as part of implementing common security administration strategies, for example, directory authentication and authorization, or use of Teradata Unity.
- Some optional properties and mechanisms do not appear in the TdgssUserConfigFile.xml. You must manually copy them from the TdgssLibraryConfigFile.xml and add them to the TdgssUserConfigFile.xml before configuring them.

## Working with LDAP Mechanism Properties

Most of the configurable TDGSS mechanism properties include the word “Ldap” in the property name, for example, LdapServerName or UseLdapConfig, indicating that they are used in directory authentication or authorization of users. All of these properties apply to the LDAP mechanism and some can also be configured in the KRB5 and SPNEGO mechanisms, depending on your Teradata GSS implementation.

### Prerequisites

- You must use a supported LDAPv3-compliant directory. For a list of directories certified for use with Teradata Vantage, see [Evaluating the System for Directory Management of Users](#).
- Before proceeding with LDAP implementation, you should consider the directory options shown in [Directory Management of Database Users](#).
- Enable external authentication. See [About External Authentication Controls](#).
- If LDAP authenticated users log on through Unity, you must coordinate the configuration LDAP mechanism properties on Unity and connected Vantage systems. See *Teradata® Unity™ Installation, Configuration, and Upgrade Guide for Customers*, B035-2523 and *Teradata® Unity™ User Guide*, B035-2520.

## Configuring LDAP for Authentication Only

Teradata GSS provides a large number of LDAP properties to support various directory-based security strategies. Teradata recommends that you start by implementing LDAP authentication for a few users and then add other options, for example, authorization of user privileges in the directory, as needed.

If you only configure LDAP authentication, user privileges are authorized by the database. Authenticated users must have the same username in the database and the directory.

The following LDAP mechanism property settings are required for the authentication-only strategy:

- Make sure that the MechanismEnabled property is set to 'yes' (the default).
- Configure the LdapServerName property. See [LdapServerName](#).

The procedure configuring mechanism property values in the TdgssUserConfigFile.xml is shown in [Making Changes to TdgssUserConfigFile.xml on Database Nodes](#).

## Configuring LDAP for Directory Authentication and Authorization

To configure LDAP for authentication and authorization requires the following:

---

### Note:

Because the user is authorized by the directory the directory username does not need an equivalent database username, as is the case for an authentication-only implementation

- Make sure that the LDAP MechanismEnabled property is set to "yes" (the default).
- Configure the LdapServerName property. See [LdapServerName](#).
- Optionally configure directory user identification options to speed directory searches for user authorization mapping. See [Optimizing Directory Searches](#).

---

The procedure for configuring mechanism property values in the TdgssUserConfigFile.xml is shown in [Making Changes to TdgssUserConfigFile.xml on Database Nodes](#).

---

### Note:

You must also map directory users to other directory objects to define user database privileges in the directory. See [Provisioning Directory Users with Teradata Schema Extensions](#) or [Using Native Directory Schema to Provision Directory Users](#).

---

## Configuring LDAP Properties for Kerberos or Directory Authorizations

If the AuthorizationSupported property is set to no in the KRB5 or SPNEGO mechanism, user privileges are authorized in the database and LDAP properties do not apply.

If the AuthorizationSupported property is set to yes in the KRB5 or SPNEGO mechanism, user privileges are authorized in the directory. Some LDAP configuration is required.

- You must configure the LdapServerName property in the authenticating mechanism (KRB5 or SPNEGO).
- Configuration of all other LDAP properties in the authenticating mechanism is optional, using the same guidelines shown for using these properties in the LDAP mechanism.

## Setting Global LDAP Protection Properties

LDAP protection properties set Teradata GSS protection parameters for LDAP and Kerberos external authentication.

These properties are optional, but Teradata strongly recommends the use of TLS protection on all systems that use simple binds. Some or all of the following properties may apply, depending on implementation.

- LdapClientUseTls
- LdapClientTlsCACert
- LdapClientTlsCACertDir
- LdapClientTlsCert
- LdapClientTlsKey
- LdapClientTlsRandFile
- LdapClientTlsReqCert
- LdapClientTlsCipherSuite
- LdapClientTlsCRLCheck
- LdapServicePasswordProtected
- LdapClientSaslSecProps
- LdapAllowUnsafeServerConnect

The settings can apply to all external authentication methods, and must be set in each authentication mechanism for which the protection is used.

For information on protection, see [Using TLS with a Directory Server](#).

For configuration instructions, see the mechanism properties listed above in [TDGSS Configuration Files, Valid Settings, and Editing Guidelines](#).

## Making Changes to TdgssUserConfigFile.xml on Database Nodes

Consider these important factors when making configuration changes on the database nodes:

- Users who run configuration change procedures on database nodes must have tdtrusted or similar OS-level privileges. See [Working with OS-Level Security Options](#).
- Run tdsstestcfg to verify the new configuration is correct.

For information about the tdsstestcfg command, see [Working with tdsstestcfg](#).

- Run the run\_tdgssconfig utility to update the TDGSSCONFIG GDO.
- If run\_tdgssconfig indicates that a TPA reset is required, run tpareset to activate the changes to the TDGSS configuration.

For information about the tpareset command, see *Teradata Vantage™ - Database Utilities*, B035-1102.

For more information about the TdgssUserConfigFile.xml file see [TDGSS Configuration Files, Valid Settings, and Editing Guidelines](#).

## Changing the TDGSS Configuration

1. On the Teradata Vantage node with the lowest ID number, navigate to the directory that provides access to TdgssUserConfigFile.xml.

```
cd /opt/teradata/tdat/tdgss/site
```

2. Make a backup copy of the TdgssUserConfigFile.xml and save it according to your site standard backup procedures.
3. Open a text editor, such as vi, and bring up a working copy of the user configuration file:

```
vi TdgssUserConfigFile.xml
```

4. Edit the properties in the file by deleting the old values and entering new values in accordance with the editing guidelines for each property. For more information, see [About Editing Configuration Files](#).

---

### Note:

Most mechanism properties work best using their factory preset values. Make sure of your reason for wanting to change a property value before you edit it.

You can add optional LDAP properties to the KRB5, LDAP and SPNEGO mechanisms and edit their default values. Copy only the optional properties you want to use from the LDAP mechanism in the TdgssLibraryConfigFile.xml and paste them into the LDAP mechanism in the copy of the TdgssUserConfigFile.xml you are editing.

---

5. Verify the configuration is correct:
  - a. Run tdsstestcfg to test the configuration. It launches a test environment in a new shell that contains the updates to the configuration file.

```
/opt/teradata/tdgss/bin/tdsstestcfg
```

- b. Run tdgssauth utility to test the newly-configured LDAP properties for their effects on directory user authentication and authorization before you commit the configuration changes to the TDGSSCONFIG GDO.

```
/opt/teradata/tdgss/bin/tdgssauth -m ldap -u <dir_user>
```

See [Working with tdgssauth](#).

- c. Exit the test shell:

```
exit
```

- d. Continue editing and testing until the configuration is correct.
6. After you complete editing and any needed testing, run the `run_tdgssconfig` utility to update the TDGSSCONFIG GDO.

```
/opt/teradata/tdgss/bin/run_tdgssconfig
```

7. If `run_tdgssconfig` indicates a TPA reset is required, run `tpareset` to activate the changes to the TDGSS configuration.

```
tpareset -f "use updated TDGSSCONFIG GDO"
```

## About TDGSS Configuration Errors

If TDGSS initialization fails after changing the configuration due to a configuration error, diagnostic messages are written to the gateway log. The log is located at:

```
/var/opt/teradata/tdtemp/gtw
```

## Returning to an Old Configuration

If you make a TDGSS configuration change and it does not work the way you had planned, or if for any other reason you want to go back to the original configuration:

1. Retrieve the backup copy of the `TdgssUserConfigFile.xml`. Recall, the backup was made when you performed your particular configuration change procedure.
2. Use the backup file as is until you are ready to retry the change.

---

### Note:

If you revert to the backup configuration, you must rerun the configuration change procedure to replace the changed configuration with the backup.

---

# Testing Directory Authentication and Authorization Setup

You can use one or more of the following methods to check Teradata Vantage-related directory entries, either as a spot check upon completing directory setup, or later to investigate user problems.

Tool	Usage
BTEQ or other Teradata Vantage client application	Log on through a Vantage client application, such as BTEQ, to verify that their user credentials are set up correctly in the directory.
tdgsstestcfg	This tool allows the user to test the changes made to the user configuration file before making those changes live with run_tdgssconfig. This tool launches a test environment in a new shell that contains the updates made to the configuration file. From the new shell, run the appropriate test tools as needed, such as tdgssauth.
tdgssauth	Use tdgssauth to test TDGSS security mechanism configurations on Unity servers and Vantage nodes. tdgssauth is used before bringing new configurations live, to test and correct authentication, authorization, and policy failures offline. It can test LDAP, Kerberos, and TDNEGO configurations.
ldapsearch	Use ldapsearch to examine directory user attributes when investigating the source of user problems.

## Working with tdgsstestcfg

The tdgsstestcfg tool allows the user to test the changes made to the user configuration file (TdgssUserConfigFile.xml) before making those changes live with run\_tdgssconfig.

This tool launches a test environment in a new shell that contains the updates made to the configuration file. From the new shell, run the appropriate test tools as needed, such as:

- tdgssauth
- tdspasswd
- tdspolicy
- gethost
- dumpcfg

After running the tests, exit the shell by typing `exit`.

Then run `run_tdgssconfig`.

## Working with tdgssauth

The tdgssauth standalone tool is used to test TDGSS security mechanism configurations on Teradata Vantage nodes and Unity servers. tdgssauth is used to test and correct authentication, authorization, and policy failures offline:

- tdgssauth is used offline to minimize the number of server TPA resets when bringing external authentication deployments and configuration fixes live.
- tdgssauth tests the following mechanisms: TD2, LDAP, Kerberos, and TDNEGO.

This tool makes use of TDGSS itself to establish a pair of security contexts based on the user's input options. One context is established to simulate the client side of a secured connection. The other context simulates the server side of a secured connection.

Once the contexts are established, the server's context is probed to determine the outcome of the authentication attempt. The user's authentication properties are acquired and displayed in human readable form. The user's name is then used to probe security policy and the results of the probe are also displayed in human readable form.

The tool can also exercise confidentiality and integrity services offered by TDGSS. Exercising these services is controlled from security policy and from command line options.

---

### Note:

This tool can perform all tdsbind functions except using command line options to adjust LDAP configuration properties. It differs from tdsbind in that it uses TDGSS to perform an actual token exchange that leads to the establishment of real security contexts while tdsbind merely emulates how the database will use a directory service when presented with particular user names and passwords. tdgssauth is also capable of invoking the policy API based on the outcome of context establishment while tdsbind is not.

---

tdgssauth is not included with Unity servers.

You can use tdgssauth to:

- Verify a permanent user's authentication and authorization properties using LDAP. See [Example: tdgssauth Verifying a Permanent User's Authentication and Authorization Properties](#).
- Verify an unmapped directory user. See [Example: tdgssauth Verifying an Unmapped User's Authentication and Authorization Parameters Using LDAP](#).
- Verify a database users' security properties using TD2. See [Example: tdgssauth Verifying a Database User's Security Properties Using TD2](#).
- Debug LDAP. See [Example: tdgssauth Debugging LDAP](#).
- Debug Kerberos. See [Example: Using tdgssauth to Debug Kerberos](#).

## Using tdgssauth Syntax

Run tdgssauth from the Teradata Vantage command prompt. Enter the tdgssauth options needed for your test. For example:



```
su - teradata
tdgssauth -u username -m mechanism -i IP_address
```

The following rules apply when using `tdgssauth`:

- You can specify `tdgssauth` options in any order, but they must be separated by spaces.
- The input is case sensitive.
- Become the `teradata` user (`su - teradata`) if the system is configured for CA certificates as part of setting up TLS. This causes `tdgssauth` to run under the Linux user *teradata* instead of *root*, which accurately represents how the system processes authentication when CA certificates are present. Also see [Using TLS with a Directory Server](#).

## Using `tdgssauth` Options

Option	Description
<b>Initialization Options</b>	
<code>-t <i>directory</i></code>	Perform a test mode initialization. In this mode the user provides a directory path to the TDGSS package when the package is located outside of the normal TDGSS directory. If this option is not specified, a normal server mode initialization is performed. Mutually exclusive with <code>-v</code> .
<code>-v <i>version</i></code>	Specifies the TDGSS version to test when performing a server mode initialization. If the version is not specified, the currently active version of TDGSS is assumed. One can determine the available versions of TDGSS by examining the contents of the directory <code>/opt/teradata/tdat/tdgss</code> . May not be used with <code>-t</code> . May be used only with server-mode initialization.
<code>-b</code>	If the <code>tdgssconfig.bin</code> file is present, this option causes TDGSS to read its configuration information from the selected TDGSS's <code>tdgssconfig.bin</code> file. If this file is not present, the selected TDGSS's <code>tdgssconfig.bin.prebuilt</code> file is read. This option is useful for debugging Unity-based LDAP and Kerberos authentication on Unity nodes. If neither file is present, the command fails.
<b>Context Establishment Options</b> (see <a href="#">Rules for <code>tdgssauth</code> Options That Control Authentication</a> )	
<code>-m <i>mechanism</i></code>	Specifies the security mechanism to exercise. If a mechanism is not specified, the mechanism defaults to TD2.
<code>-n <i>target</i></code>	Specifies the target Service Principal Name (SPN). This name is required for Kerberos (KRB5) authentication. It is formed by concatenating the string <code>TERADATA</code> , a slash character, and the fully qualified primary DNS name of the node where the tool is running. For example, if the tool is being run from a node named <code>dbc1.example.com</code> , then the target name would be <code>TERADATA/dbc1.example.com</code> . The user has control of the target name to allow testing non-Teradata targets and to vary the primary node name when case-sensitivity is needed.
<code>-D</code>	Requests establishment of a security context capable of delegation. A context so established permits the server to impersonate the client in other services. Note that specifying this option is only a request; there is no guarantee that the context will be capable of delegation.

Option	Description
-M	Requests that mutual authentication occur during context establishment. Mutual authentication requires that the two participants in context establishment fully authenticate each other. Note that specifying this option is only a request; there is no guarantee that mutual authentication will occur.
-R	Request replay detection.
-S	Requests that the context be established to permit detection of out of sequence messages. Note that specifying this option is only a request; there is no guarantee that out of sequence detection will be possible.
-I	Requests establishment of a security context capable of guaranteeing message integrity. A context with integrity capability can generate message integrity codes suitable for the authentication mechanism to help the peer determine when a message tampering has occurred. Note that specifying this option is only a request; there is no guarantee that the context will be capable of guaranteeing message integrity.
-C	Requests establishment of a security context capable of confidentiality. A context capable of confidentiality will be capable of encrypting or decrypting network traffic. Note that specifying this option is only a request; there is no guarantee that the context will be capable of confidentiality.
-?	Requests that a usage message be displayed.
-u <i>username</i>	Specifies the username to authenticate. This option is required for mechanisms that do not support single sign on.
-w <i>password</i>	Specifies the user's password when a credential is required. If -w is omitted the tool prompts for the password and it is read securely from the user's terminal. Teradata recommends that passwords never be provided on a command line because command lines are visible to other users using tools like ps .
-a <i>additional-logdata</i>	Specifies additional authorization information other than authcid and password; for example, you can specify profile information: <code>tdgssauth -u user -w password -a profile=myprofile -m ldap</code>
-i <i>ipaddr</i>	The IPv4 or IPv6 address of the client to use in security policy tests. If this option is not included, security policy checks will not be performed. If -i is included but -u has not been specified and the mechanism is non-authenticating (such as TD2), security policy checks will not be performed.
<b>Message Exchange Options</b>	
-T <i>text</i>	Requests that the specified <i>text</i> be wrapped and unwrapped. Wrapping is the process where an integrity code is added to the message and the message is optionally encrypted. Unwrapping is the process where a wrapped message is decrypted (if it was encrypted) and the integrity of the message is verified. Four things happen for each message. First, the client's context is used to wrap the message and the resulting wrapped token is dumped in hex form. Second, the server's context is used to unwrap the message wrapped by the client's context and the results

Option	Description
	<p>of the unwrap operation are dumped. Third, the server's context is used to wrap the unwrapped message and the results of the wrapping are dumped. Finally, the client's context is used to unwrap the message wrapped by the server's context and the results of the unwrapping are dumped.</p> <p>The tool accepts zero or more -T options on the command line. The process described above is followed for each -T option.</p>
-e	<p>Requests encryption. Specifically, this option requests that the text specified in -T options have both confidentiality and a message integrity code applied. Normally, TDGSS will add a message integrity code.</p> <p>If policy enforcement requires confidentiality, then this option will have no effect. If policy enforcement requires integrity only or nothing at all, this option will add confidentiality by causing the text specified in the -T options to be encrypted.</p>
-q <i>qop</i>	<p>Request that wrapping be done with a particular QoP (Quality of Protection). Legitimate values are numbers in the range 0 through 3 or the mnemonic names of default, low, medium, and high.</p> <p>If policy enforcement requires more protection than the user has requested with -q and -e, policy enforcement's choice overrides the user's selection. If the user's selection is greater than the minimum required by policy enforcement, the user's selection is used.</p>
<b>Tracing Options</b>	
-V <i>level</i>	<p>Enable token dumps during context establishment and optionally add low level LDAP tracing depending on the value of <i>level</i>. The <i>level</i> argument is an integer value resulting from OR-ing the following bit values:</p> <ul style="list-style-type: none"> <li>• 0x0001 - Execution traces</li> <li>• 0x0002 - LDAP packets</li> <li>• 0x0004 - Passed arguments</li> <li>• 0x0008 - Connection information</li> <li>• 0x0010 - BER encoding</li> </ul> <p><b>Note:</b></p> <p>The output includes the user's password and the database service password. Both the character interpretation and dumped hex need to be changed before sharing this trace information.</p>
-l	Enables TDNEGO logging display.

### Rules for tdgssauth Options That Control Authentication

The tdgssauth options that control authentication are: -u, -w, and -a. The following rules apply for using these options:

- If the mechanism supports authentication then at least one of -u or -a must be specified.
- If the mechanism supports authentication and -u is specified, the user's name and password are hardcoded into a properly escaped and quoted User Principal Name (UPN). If -a has been specified, then a space character and the value provided in the -a option are appended to the UPN and the resulting string is used as mechdata (.logdata).

- If the mechanism supports authentication and -u is not specified, -a must be specified and the value passed must be complete mechdata, which includes the user's name and password (legacy mode).
- If the mechanism does not support authentication, then -u must be specified only if security policy checks are to be made.
- If the mechanism supports authentication but does not support single sign on, a password is required. The password is provided using the -w option (not recommended) or by allowing tdgssauth to securely prompt for the password.

## Example: tdgssauth Verifying a Permanent User's Authentication and Authorization Properties

The example shows how to verify a permanent user's authentication and authorization properties using LDAP from a given IP address. Run:

```
tdgssauth -u userconflow -m ldap -i 198.51.100.20
```

The user's name (-u) is the same as it is specified in a bteq .logon command. The -m option specifies the logon mechanism to use (LDAP in this case). The -i option specifies the IP address from which the user will connect.

Result:

```
1> Please enter a password:
2>                               Status: authenticated, not authorized
3>                               Database user: userconflow [permanent user]
4>                               Authenticated user: ldap://
dsa1.example.com:389/uid=userconflow,ou=principals,dc=example,dc=com
5>                               Audit trail identifier: userconflow
6>                               Authenticating service: dbssvc
7>                               Actual mechanism employed: ldap [OID 1.3.6.1.4.1.191.1.1012.1.20]
8>                               Mechanism specific data: userconflow
9>
10> Security context capabilities: replay detection
11>                               out of sequence detection
12>                               confidentiality
13>                               integrity
14>                               protection ready
15>                               exportable security context
16>
17> Minimum quality of protection: 1 (Low) with confidentiality and integrity
18>                               Options: none
```

The following explains the output from the command:

Line Number	Description
1> Enter a password	<p>When prompted, enter the user's password for the specified mechanism. In this example, enter the user's LDAP password because the specified mechanism is ldap. If KRB5 is the specified mechanism, enter the user's KRB5 password.</p> <p>Use -w and specify the user's password on the command line to avoid being prompted for the password.</p> <p><b>Note:</b></p> <p>It is not recommended to specify the user's password on the command line.</p>
2> Status: authenticated, not authorized	The user authenticated successfully, but the user does not have a mapping to an explicit Vantage user in the directory.
3> Database user: userconflow [permanent user]	The database user's name and the database user is a permanent user (the user was created in the database by the DBA).
4> Authenticated user: ldap://dsa1.example. com:389/uid=userconflow, . ..	The identity of the user in the directory server and the server that authenticated the user.
5> Audit trail identifier: userconflow	The user's audit trail identifier used in event logs caused by a session logged on as this user.
6> Authenticating service: dbssvc	The service name of the service used to authenticate the user. The service is configured in the <LdapConfig> section of the TdgssUserConfigFile.xml file.
7> Actual mechanism employed: ldap [OID 1.3.6.1. 4.1.191.1.1012.1.20]	The name and Object Identifier (OID) of the actual authentication mechanism used to authenticate the user. Note, the TDNEGO mechanism reports the actual mechanism that it selected to authenticate the user. Other explicitly named mechanisms report themselves here.
8> Mechanism specific data: userconflow	The mechanism specific data. This data is used by other parts of the system during the login process and is not used by TDGSS. In most, if not all cases, this provides the name of the user from the -u command line option.
10 - 15> Security context capabilities: replay detection out of sequence detection ... exportable security context	These lines show what a particular security context provides. The security context is the one established for the named user using the specified mechanism.
17> Minimum quality of protection: 1 (Low) ...	The minimum QoP that the user is required to use for the life of the session. In this example, the directory configuration shows that this user needs to use, at the very least, a low strength confidentiality QoP. The database will enforce this and if the session uses a less secure QoP than the one specified the user's session will be terminated.

Line Number	Description
18> Options: none	The connection options in effect for this user. In this example, the word none indicates that this is a normal connection. This value may contain has-policy or no-direct-connect. has-policy says that the user must use only a plaintext connection to the database and is used for very specialized purposes. no-direct-connect says that the user is not permitted to connect directly to the database, but must instead come through Unity.

## Example: tdgssauth Verifying an Unmapped User's Authentication and Authorization Parameters Using LDAP

The example shows how to verify an unmapped user's authentication and authorization properties using LDAP from a given IP address. Run:

```
tdgssauth -u drct01 -m ldap -i 198.51.100.20
```

The user's name (-u) is the same as it is specified in a bteq .logon command. The -m option specifies the logon mechanism to use (LDAP in this case). The -i option specifies the IP address from which the user will connect.

Result:

```
1> Please enter a password:
2>                               Status: authenticated, authorized
3>                               Database user: drct01 [unmapped user,
autoprovisioning candidate]
4>                               Profile: profxu1
5>                               External roles: extrole01xu1, extrole02xu1, extrole03xu1
6>                               Authenticated user: ldap://
dsa1.example.com:389/uid=drct01,ou=principals,dc=example,dc=com
7>                               Audit trail identifier: drct01
8>                               Authenticating service: dbssvc
9>                               Actual mechanism employed: ldap [OID 1.3.6.1.4.1.191.1.1012.1.20]
10>                              Mechanism specific data: drct01
11>
12> Security context capabilities: replay detection
13>                               out of sequence detection
14>                               confidentiality
15>                               integrity
16>                               protection ready
17>                               exportable security context
18>
```

```
19> Minimum quality of protection: none
20> Options: none
```

The following explains the output from the command:

Line Number	Description
1> Enter a password	<p>When prompted, enter the user's password for the specified mechanism. In this example, enter the user's LDAP password because the specified mechanism is ldap. If KRB5 is the specified mechanism, enter the user's KRB5 password.</p> <p>Use <code>-w</code> and specify the user's password on the command line to avoid being prompted for the password.</p> <p><b>Note:</b> It is not recommended to specify the user's password on the command line.</p>
2> Status: authenticated, authorized	The user authenticated and authorized successfully.
3> Database user: drct01 [unmapped user, autoprovisioning candidate]	The user is not mapped to a Vantage database user (an unmapped user) and this user is a candidate for autoprovisioning.
4> Profile: profxu1	The user has the <i>profxu1</i> profile associated with the session.
5> External roles: extrole01xu1, extrole02xu1, extrole03xu1	The user is permitted to occupy the three external roles, <i>extrole01xu1</i> , <i>extrole02xu1</i> , and <i>extrole03xu1</i> . The DBA must create those roles in the database and grant them rights.
6> Authenticated user: ldap://dsa1.example.com:389 /uid=drct01,ou=principals, dc=example,dc=com	The identity of the user in the directory server and the server that authenticated the user.
7> Audit trail identifier: drct01	The user's audit trail identifier used in event logs caused by a session logged on as this user.
8> Authenticating service: dbssvc	The service name of the service used to authenticate the user. The service is configured in the <LdapConfig> section of the TdgssUserConfigFile.xml file.
9> Actual mechanism employed: ldap [OID 1.3.6.1. 4.1.191.1.1012.1.20]	The name and Object Identifier (OID) of the actual authentication mechanism used to authenticate the user. Note, the TDNEGO mechanism reports the actual mechanism that it selected to authenticate the user. Other explicitly named mechanisms report themselves here.
10> Mechanism specific data: drct01	The mechanism specific data. This data is used by other parts of the system during the login process and is not used by TDGSS. In most, if not all cases, this simply provides the name of the user from the <code>-u</code> command line option.

Line Number	Description
12 - 17> Security context capabilities: replay detection, out of sequence detection ... exportable security context	These lines tell us what a particular security context provides. The security context is the one established for the named user using the specified mechanism.
19> Minimum quality of protection: None	The minimum QoP that the user is required to use for the life of the session. In this example, during the life of a session this user can use any QoP including no QoP at all.
20> Options: none	The connection options in effect for this user. In this case, the word none indicates that this is a normal connection. This value may contain has-policy or no-direct-connect. has-policy says that the user must use only a plaintext connection to the database and is used for very specialized purposes. no-direct-connect says that the user is not permitted to connect directly to the database, but must instead come through Unity.

## Example: tdgssauth Verifying a Database User's Security Properties Using TD2

The example shows how to verify a database user's security properties using the TD2 mechanism. Run:

```
tdgssauth -u userconfhigh -m td2 -i 198.51.100.20
```

The user's name (-u) is the same as it is specified in a bteq .logon command. The -m option specifies the logon mechanism to use (TD2 in this example). The -i option specifies the IP address from which the user will connect.

Result:

```
1>      Status: not authenticated, not authorized
2>      Actual mechanism employed: TD2 [OID 1.3.6.1.4.1.191.1.1012.1.1.9]
3>
4> Security context capabilities: replay detection
5>                                out of sequence detection
6>                                confidentiality
7>                                integrity
8>                                protection ready
9>                                exportable security context
10>
11> Minimum quality of protection: 3 (High) with confidentiality and integrity
12>                                Options: none
```

The following explains the output from the command:



Line Number	Description
1> Status: not authenticated, not authorized	The user has not authenticated nor authorized. There is only a security context. TD2 was used to establish the security context. TD2 does not authenticate, but instead requires the database to verify the user's name and password, so there is not a prompt asking for the user's password.
2> Actual mechanism employed: TD2 [OID 1.3.6.1.4.1.191.1.1012.1.1.9]	TD2 is the security mechanism.
4 - 9> Security context capabilities: replay detection, out of sequence detection ... exportable security context	These lines tell us what a particular security context provides. The security context is the one established for the named user using the specified mechanism.
11> Minimum quality of protection: 3 (High) with confidentiality and integrity	The minimum QoP that the user is required to use for the life of the session. In this example, the user must use a high strength confidentiality QoP.
12> Options: none	The connection options in effect for this user. In this case, the word none indicates that this is a normal connection. This value may contain has-policy or no-direct-connect. has-policy says that the user must use only a plaintext connection to the database and is used for very specialized purposes. no-direct-connect says that the user is not permitted to connect directly to the database, but must instead come through Unity.

## Example: tdgssauth Wrap and Unwrap

The example shows wrapping a string to protect it with a signature. Run:

```
tdgssauth -u userconfhigh -m td2 -i 198.51.100.20 -T "this is a test"
```

The user's name (-u) is the same as it is specified in a bteq .logon command. The -m option specifies the logon mechanism to use (TD2 in this case). The -i option specifies the IP address from which the user will connect. -T wraps and unwraps the string.

Result:

```
Status: not authenticated, not authorized
  Actual mechanism employed: TD2 [OID 1.3.6.1.4.1.191.1.1012.1.1.9]

Security context capabilities: replay detection
                             out of sequence detection
                             confidentiality
                             integrity
```

```

                                protection ready
                                exportable security context

Minimum quality of protection: 3 (High) with confidentiality and integrity
Options: none

String to be wrapped: this is a test
Requesting QoP: 3 (High)
Requesting Confidentiality: yes

Wrapped text (by client side):
00000000: ff 6e 79 ed f9 b3 fa 42 ee 79 3e c1 b0 7a af ab *.ny....B.y>..z..*
00000010: e9 ce 82 c5 3e bd 49 e3 a6 70 95 ab 42 9a 95 6b *....>.I..p..B..k*
00000020: d2 1b 2a 8e c6 f1 04 91 24 78 5d 50 e5 8b 69 8a *...*.....$x]P..i.*
00000030: 86 2a 0f ab 75 f6 3d 7e 19 14 3f d3 35 6d 77 74 *.*...u.=~...?.5mwt*
00000040: 03 07 04 03 00 00 00 40 00 00 00 00 00 00 01 *.....@.....*

Unwrapped text (by server side):
00000000: 74 68 69 73 20 69 73 20 61 20 74 65 73 74      *this is a test*

Actual QoP applied: 3 (High)
Confidentiality applied: yes

Wrapped text (by server side):
00000000: d9 f0 77 7b 1a 9c 75 2e 3e 65 6e 75 ee 9a 07 33 *..w{...u.>enu...3*
00000010: a5 b3 f0 8e 04 3e 24 15 a8 6e b8 29 97 68 43 c5 *.....>$..n.)..hC.*
00000020: 4f dc f3 d5 14 70 9d e1 27 38 9a de 50 3c 95 fd *0....p..'8..P<...*
00000030: 8d cf 2f e9 b1 ed 77 18 aa ca 53 7d 05 61 50 dc *.../...w...S}.aP.*
00000040: 03 07 84 03 00 00 00 40 00 00 00 00 00 00 01 *.....@.....*

Unwrapped text (by client side):
00000000: 74 68 69 73 20 69 73 20 61 20 74 65 73 74      *this is a test*

Actual QoP applied: 3 (High)
Confidentiality applied: yes

```

The -T option specifies a string to wrap and unwrap. Wrapping causes the text to be protected with a signature and to optionally be encrypted (see the -c and -e options in [Using tdgssauth Syntax](#)). The tool will use the QoP configured for the session when invoking the wrap function in TDGSS. In this case, the client side wrapped the message and turned it into an 80-byte string. The 80-byte string is passed to the server side of TDGSS and requests the server side to unwrap the string back to the original string. The server then wraps the string it unwrapped and generates a different 80-byte string. The client unwraps the string from the server side back to the original string.

The -T option can be used with any mechanism and any number of -T options may be specified.

## Example: tdgssauth Debugging LDAP

The example shows how to use the -V option to debug LDAP. The example shows all the message exchanges between TDGSS and the directory server. Run:

```
tdgssauth -u userconflow -m ldap -t -i 198.51.100.20 -V 2
```

The user's name (-u) is the same as it is specified in a bteq .logon command. The -m option specifies the logon mechanism to use (LDAP in this case). The -t option says to perform a test mode initialization. The -i option specifies the IP address from which the user will connect. The -V option requests verbose output.

First, a series of message dumps headed by ldap\_read and ldap\_write are output. These are the actual message exchanges between TDGSS and the directory server. These messages happen when TDGSS is initialized, once per process lifetime. The gateway will cause TDGSS to generate this message when coming up after a TPA reset. These are not per-session messages.

### Note:

The output includes the user's password and the database service password. Both the character interpretation and dumped hex need to be changed before sharing this trace information.

Result: Typically, this command produces pages of output, which is shortened here.

```
0000: 30 39 02 01 01 60 34 02 01 03 04 27 63 6e 3d 64 09...`4....'cn=d
0010: 62 73 73 76 63 2c 6f 75 3d 73 65 72 76 69 63 65 bssvc,ou=service
0020: 73 2c 64 63 3d 65 78 61 6d 70 6c 65 2c 64 63 3d s,dc=example,dc=
0030: 63 6f 6d 80 06 73 65 63 72 65
74 com..SECRETPASSWORD
ldap_read: want=8, got=8
0000: 30 0c 02 01 01 61 07 0a 0....a..
ldap_read: want=6, got=6
0000: 01 00 04 00 04 00 .....
ldap_write: want=165, written=165
0000: 30 81 a2 02 01 02 63 81 9c 04 00 0a 01 00 0a 01 0....c.....
0010: 00 02 01 00 02 01 00 01 01 00 87 0b 6f 62 6a 65 .....object
0020: 63 74 43 6c 61 73 73 30 7c 04 0b 6f 62 6a 65 63 ctClass0|..objec
0030: 74 43 6c 61 73 73 04 15 73 75 70 70 6f 72 74 65 tClass..supporte
0040: 64 43 61 70 61 62 69 6c 69 74 69 65 73 04 14 64 dCapabilities..d
0050: 65 66 61 75 6c 74 4e 61 6d 69 6e 67 43 6f 6e 74 efaultNamingCont
0060: 65 78 74 04 10 6e 65 74 73 63 61 70 65 6d 64 73 ext..netscapemds
0070: 75 66 66 69 78 04 12 73 75 70 70 6f 72 74 65 64 uffix..supported
0080: 45 78 74 65 6e 73 69 6f 6e 04 1a 63 6f 6e 66 69 Extension..confi
0090: 67 75 72 61 74 69 6f 6e 4e 61 6d 69 6e 67 43 6f gurationNamingCo
```

```

00a0:  6e 74 65 78 74                                ntext
ldap_read: want=8, got=8

.
.
.

```

Next, tdgssauth prompts for a password. After entering the password, a series of dumps, headed by "Client's token" and "Server's token," are output. These are the actual tokens that the client side and server side of TDGSS exchange to authenticate and authorize the user.

Please enter a password:

Client's token:

```

00000000: 01 01 01 00 00 00 00 74 00 00 00 0d 00 00 00 00 *.....t.....*
00000010: 01 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 *.....*
00000020: 00 00 00 00 00 00 00 34 00 00 00 00 00 00 00 00 *.....4.....*
00000030: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 *.....*
00000040: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 *.....*
00000050: e1 32 e2 18 d0 01 02 d3 01 01 d3 01 03 d4 01 04 *.2.....*
00000060: d5 02 00 80 d5 02 00 c0 d5 02 01 00 e2 03 d1 01 *.....*
00000070: 04 e2 03 d1 01 06 e2 03 d1 01 07 e2 07 d2 01 05 *.....*
00000080: d6 02 08 00 06 0c 2b 06 01 04 01 81 3f 01 87 74 *.....+.....?..t*
00000090: 01 14 46 08 00 01 81 00 03 00 00 00 01 00 00 00 *..F.....*
000000a0: 1e 01                                           *.*.

```

Server's token:

```

00000000: 03 02 01 01 00 00 03 a6 00 00 00 0d 00 00 00 00 *.....*
00000010: 01 00 00 00 00 00 01 00 00 00 01 00 00 00 01 00 *.....*
00000020: 00 00 00 00 00 00 00 66 00 00 00 00 00 00 00 00 *.....f.....*
00000030: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 *.....*
00000040: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 *.....*
00000050: 8a b3 f8 6e 8d 37 4b 78 2f 31 da d5 f2 7d 6a fd *...n.7Kx/1...}j.*
00000060: a3 01 50 c1 1a 20 cf 63 46 71 2a e2 d2 c6 b7 0a *..P.. .cFq*.....*
00000070: 5b 79 d4 5d 4c 0c 23 2a 06 5b 20 7b 12 1b 2c 33 *[y.]L.#*.[ {...,3*
00000080: e1 47 b5 98 3c 38 a1 08 7f 27 27 03 b0 b8 39 cb *.G...<8...'...9.*
00000090: a6 f7 1c 5d 0e b5 1e c8 90 93 4e ac f2 c7 dd 2a *...].....N....**

```

.  
.  
.

For the last token, more `ldap_read` and `ldap_write` dumps are output. These are the messages exchanged between the database and the directory server to authenticate and authorize the user. The user's password may appear in the dump if simple binding is used. Simple binding is used when the service or LDAP mechanism `LdapClientMechanism` property is set to the value "simple."

```

        Status: authenticated, not authorized
        Database user: userconflow [permanent user]
        Authenticated user: ldap://
10.25.67.159:389/uid=userconflow,ou=principals,dc=example,dc=com
        Audit trail identifier: userconflow
        Authenticating service: dbssvc
        Actual mechanism employed: ldap [OID 1.3.6.1.4.1.191.1.1012.1.20]
        Mechanism specific data: userconflow

Security context capabilities: replay detection
                             out of sequence detection
                             confidentiality
                             integrity
                             protection ready
                             exportable security context

ldap_write: want=59, written=59
 0000: 30 39 02 01 01 60 34 02 01 03 04 27 63 6e 3d 64 09...`4....'cn=d
 0010: 62 73 73 76 63 2c 6f 75 3d 73 65 72 76 69 63 65 bssvc,ou=service
 0020: 73 2c 64 63 3d 65 78 61 6d 70 6c 65 2c 64 63 3d s,dc=example,dc=
 0030: 63 6f 6d 80 06 73 65 63 72 65 74 com..secret

ldap_read: want=8, got=8
 0000: 30 0c 02 01 01 61 07 0a 0....a..

ldap_read: want=6, got=6
 0000: 01 00 04 00 04 00 .....

ldap_write: want=574, written=574
 0000: 30 82 02 3a 02 01 02 63 82 02 33 04 30 63 6e 3d 0.....c..3.0cn=
 0010: 70 6f 6c 69 63 79 2c 63 6e 3d 54 65 72 61 64 61 policy,cn=Terada
 0020: 74 61 20 50 6f 6c 69 63 69 65 73 2c 64 63 3d 65 ta Policies,dc=e
 0030: 78 61 6d 70 6c 65 2c 64 63 3d 63 6f 6d 0a 01 02 xample,dc=com...
 0040: 0a 01 00 02 01 00 02 01 00 01 01 00 a1 82 01 db .....

.
.
.
```

## Example: Using tdgssauth to Debug Kerberos

The example shows how to use tdgssauth to debug a Kerberos configuration:

```
$ kinit jdoe
Password for jdoe@EXAMPLE.COM:

$ tdgssauth -m KRB5 -n TERADATA/dbc1.example.com -i 10.0.1.195
      Status: authenticated, not authorized
      Database user: jdoe [permanent user]
      Authenticating domain: EXAMPLE.COM
      Actual mechanism employed: KRB5 [OID 1.2.840.113554.1.2.2]

Security context capabilities: mutual authentication
                             confidentiality
                             integrity
                             protection ready
                             exportable security context

Minimum quality of protection: none
Options: none
```

The example first uses kinit to establish a credential for user jdoe in the default domain (example.com in this case).

The next command is the tdgssauth command specifying the KRB5 mechanism (with the -m option), the service principal name consisting of the string “TERADATA/” followed by the fully qualified primary DNS name of the node being tested (-n option) and the IP address of the client (the -i option).

If you omit the -i option, you get the following output:

```
$ kinit jdoe
Password for jdoe@EXAMPLE.COM:

$ tdgssauth -m KRB5 -n TERADATA/dbc1.example.com
      Status: authenticated, not authorized
      Database user: jdoe [permanent user]
      Authenticating domain: EXAMPLE.COM
      Actual mechanism employed: KRB5 [OID 1.2.840.113554.1.2.2]

Security context capabilities: mutual authentication
                             confidentiality
                             integrity
                             protection ready
```

```
exportable security context
```

```
*** WARNING: Policy checks will not be made; IP address and/or user name was
not specified.
***           Please include -i and -u options if policy checks are to be made.
```

## Working with tdsbind

### Note:

tdsbind is deprecated. Teradata recommends using the tdgssauth tool instead of tdsbind. tdgssauth can test more security mechanisms than tdsbind and it more accurately validates security mechanism configurations because it uses actual TDGSS services while performing the offline test of the new configuration. See [Working with tdgssauth](#).

You can use tdsbind to test the setup of various user management parameters, for example:

- Authenticate an unmapped directory user. See [Example: Tdsbind Output for a Directory User not Mapped to a Database User](#).
- Authenticate a mapped directory user and check user mappings to Teradata directory objects. See [Example: Tdsbind Output for a Directory User Mapped to a Database User](#).
- Check directory users for applicable IP access restrictions, based on mappings to database users. See [Testing Directory-Based IP Restrictions](#).

The TDGSS software installed on all Teradata Vantage nodes includes a copy of tdsbind.

### Note:

You can also use tdsbind to test other setup parameters. See:

- [Testing XML-Based IP Restrictions](#)
- [Testing Directory-Based IP Restrictions](#)
- The testing step for each operation system dependent configuration change procedure in [Making Changes to TdgssUserConfigFile.xml on Database Nodes](#)

## Using Tdsbind Syntax

Run tdsbind from the Teradata Vantage command prompt. Enter the tdsbind options needed for your test, for example:

```
su - teradata
tdsbind [-u dir_user | -U td_user ] option_spec [...]
```

*option\_spec**-option value***Syntax Elements***option  
value*

For Tdsbind options and their values, see [Using Tdsbind Options](#).

**Using Tdsbind Options**

Tdsbind Option	Description
<i>-B base_ fqdn</i> [Deprecated]	<p>The FQDN of a directory object containing directory user and group objects. By default, tdsbind uses the value of the <code>LdapBaseFQDN</code> property.</p> <p><b>Note:</b> Although this option continues to function, it is deprecated for future use. See <a href="#">LdapBaseFQDN [Deprecated]</a>.</p>
<i>-c</i>	<p>Causes the system to initialize TDGSS as if it were a configured client. This attribute is for future use only, and is not currently valid. You cannot use this option if you use either the <code>-s</code> or <code>-t</code> option.</p>
<i>-D referral_method</i>	<p>Specifies the how referrals are chased. If this property is omitted, Tdsbind uses the value of the <code>LdapClientDeref</code> property from the TDGSS user configuration file.</p> <p><b>Note:</b> Teradata recommends that you do not use referral chasing. See <a href="#">LdapClientDeref</a>.</p>
<i>-d ldap_realm</i> [Deprecated]	<p>The name of the SASL realm for DIGEST-MD5 binding of the directory user.</p> <p><b>Note:</b> The DIGEST-MD5 authentication protocol used by LDAP is deprecated. Teradata strongly recommends you use simple binding with TLS protection, and stop using DIGEST-MD5.</p> <p>This option is meaningful only when both of the following are true:</p> <ul style="list-style-type: none"> <li>• Binding is set to use DIGEST-MD5</li> <li>• The directory service offers more than one realm</li> </ul> <p>By default, tdsbind uses the value configured for the <code>LdapServerRealm</code> property in the LDAP mechanism.</p> <p><b>Note:</b> This option is deprecated. If you need to specify a non-default value, specify a value for the <code>LdapServerRealm</code> in the <code>-O</code> option. See <a href="#">LdapServerRealm [Deprecated]</a>.</p>



Tdsbind Option	Description
<code>-f file name</code>	The name of a file generated using the ipxml2bin utility, which defines a set of IP logon restrictions. For information about XML IP restrictions, see <a href="#">Creating XML-Based IP Restrictions</a> .
<code>-G groupbase_fqdn</code>	The FQDN of any object in the directory that is the base of a subtree which contains group objects. If you do not specify an FQDN for -G, tdsbind uses the value of the LdapGroupBaseFQDN property. If the LdapGroupBaseFQDN property does not contain an FQDN, the system uses the value for the -B option (not recommended). See <a href="#">LdapGroupBaseFQDN</a> .
<code>-h ldap_host</code> [Deprecated]	The name of the LDAP directory server. By default, tdsbind uses the value of the LdapServerName property. <b>Note:</b> The -h option is deprecated. If you need to specify a value other than the default, use the -O option to specify an LdapServerName. See <a href="#">LdapServerName</a> for naming options.
<code>-l ip_add</code>	Specifies an IP address. Tdsbind tests the IP address against any configured IP restrictions to determine whether any of the restrictions denies the user access to the database from the IP address. <ul style="list-style-type: none"> <li>If you do not specify -U, tdsbind initiates a bind of the user.</li> <li>If you specify the -U option, tdsbind skips the bind operation and tests the named user and IP address for restrictions.</li> </ul> <b>Note:</b> To test a new IP restriction, before you create or change the system IP configuration, also use the -f option to specify a test file. For further information on setting up IP restrictions, see <a href="#">Restricting Logons by IP Address</a> .
<code>-O property=value</code>	Specifies one or more alternate values for LDAP property settings, and supersedes the values in the TdgssUserConfigFile.xml, with these constraints: <ul style="list-style-type: none"> <li>For each LDAP property, if a -O specification is not present, the system uses the value found in the TdgssUserConfigFile.xml.</li> <li>You can specify any LDAP mechanism properties.</li> <li>You cannot specify non-LDAP properties, identity map, or identity search.</li> <li>If you specify more than one LDAP property, the specifications must be space separated, and the -O must precede each one, for example: <pre>tdsbind -O LdapServerName=name -O LdapGroupBaseFQDN=fqdn</pre> </li> <li>If you specify an option more than once in a tdsbind command, for example, if you specify both -S and -O LdapSystemFQDN=fqdn, the command fails.</li> </ul> You can use -O to test new configurations. See <a href="#">Making Changes to TdgssUserConfigFile.xml on Database Nodes</a> . For detailed information on using LDAP properties, see the topics beginning with <a href="#">Directory Identification and Search Properties</a> .

Tdsbind Option	Description
<code>-p ldap_port</code> [Deprecated]	<p>Specifies the LDAP service port. The <code>-p</code> option is deprecated. The system defaults to the port designation associated with the naming convention specified for the <code>LdapServerName</code> property.</p> <p><b>Note:</b> If you need to specify a port other than that associated with the <code>LdapServerName</code> property, use the <code>-O</code> option to change the <code>LdapServerName</code> to include the optional port designation. See <a href="#">LdapServerName</a>.</p>
<code>-q</code>	<p>Specifies that tdsbind run in “quiet” mode, that is, suppress the display of LDAP properties and values, and show only user-specific information.</p>
<code>-R referral_setting</code>	<p>Specifies whether referral chasing is enabled or disabled. If you do not specify this option, tdsbind uses the value of the <code>LdapClientReferrals</code> property, which is set to off by default.</p> <p><b>Note:</b> Teradata recommends that you do not use referral chasing. See <a href="#">LdapClientReferrals</a>.</p>
<code>-r random_device</code>	<p>Specifies the name of a device, FIFO, or pipe that provides random bits when the default <code>/dev/[u]random</code> (the built-in Linux random number generator) is not available, or if an alternate source is preferred.</p> <p>If you do not specify a value for this option, the system defaults to <code>/dev/[u]random</code>, or to the value of the <code>LdapClientRandomDevice</code> property, if it is configured.</p>
<code>-S system_fqdn</code>	<p>Specifies the FQDN of the directory object that defines the Teradata Vantage server (the <code>tdatSystem</code> object).</p> <p>By default, tdsbind uses the value of the <code>LdapSystemFQDN</code> property. See <a href="#">LdapSystemFQDN</a>.</p>
<code>-s</code>	<p>If you use this option, the system initializes TDGSS as if it were a configured database node, and is the default if the tdsbind statement does not define other TDGSS initializing criteria.</p> <p>You cannot use this option if you use either the <code>-c</code> or <code>-t</code> option.</p>
<code>-t directory_name</code>	<p>Specifies a directory containing a different version of the TDGSS bin and etc directories. This argument causes the system to initialize TDGSS in a test environment instead of the normal default location.</p> <p>You cannot use this option if you use either the <code>-c</code> or <code>-s</code> option.</p>
<code>-U td_user</code>	<p>Specifies a Vantage username, which tdsbind uses, along with the IP address specified in <code>-l ip_add</code>, to evaluate whether a user logon is restricted.</p> <p>If you use this option, the bind process does not take place, because it is not required to test IP restrictions. Tdsbind ignores any specified bind options, for example, the database user password.</p> <p>When this option is specified, the <code>-l</code> option is required.</p>
<code>-u dir_user</code>	<p>The authentication identifier for the directory user; a valid directory user authcid.</p>

Tdsbind Option	Description
	<p>You must specify this option if you are binding a directory user, for example, when you test directory user authentication and authorization characteristics against a new TdgssUserConfigFile.xml. There is no default.</p> <p><b>Note:</b></p> <p>This option is not required when you use tdsbind to test user IP restrictions. Instead, use the -U option to specify a database user.</p>
-V	<p>Specifies the debug flags to be passed to the OpenLDAP client API. If this property is omitted, tdsbind uses the value of the LdapClientDebug property from the TDGSS user configuration file. The default is no.</p> <p>You can use the LdapClientDebug property to assist the Teradata Support Center in debugging LDAP directory issues, but this property is not user settable.</p> <p><b>Note:</b></p> <p>Do not use this option without Teradata Support Center assistance. Values other than the default may cause system malfunction.</p>
-v <i>version</i>	Initializes a specific version of TDGSS. Tdsbind defaults to the current TDGSS version. Like -t, you cannot use -v with the -c or -s option.
-w <i>password</i>	<p>The password for the directory user specified in the -u option.</p> <p>By default, tdsbind interactively prompts the user for a password and securely reads the submitted password.</p>
-X <i>user_base_fqdn</i>	<p>The fully qualified distinguished name of any object in the directory that is the base of a subtree which contains the user objects.</p> <p>If you omit this property, tdsbind uses the value of the LdapUserBaseFQDN property. See <a href="#">LdapUserBaseFQDN</a>.</p> <p>If the value of the LdapUserBaseFQDN property is not set, tdsbind uses the value for the tdsbind -B option.</p>

## Usage Notes

The following rules apply when using tdsbind:

- Become the teradata user (su - teradata) if the system is configured for CA certificates as part of setting up TLS. This causes tdsbind to run under the Linux user teradata instead of root, which accurately represents how the system processes authentication when CA certificates are present. Also see [Using TLS with a Directory Server](#).
- You can specify tdsbind options in any order, but they must be separated by spaces.
- The input is case sensitive, for example, the meaning of -u is completely different than -U.
- Options that have a corresponding LDAP mechanism property, for example -B (LdapBaseFQDN) are deprecated, although still usable. Where possible, use -O to specify the corresponding LDAP property values in a space-separated list instead of using the individual deprecated options.

**Note:**

If you simultaneously specify an LDAP property value in the -O option, for example LdapBaseFQDN, and also specify the corresponding standalone option (-B), tdsbind fails.

## Diagnosing Logon Failure Due to Incorrect Realm Information

Directory users may receive the generic error message, “SSO logon failed by gateway.” This message is often related to entry of (or defaulting to) an invalid directory server realm name.

To help diagnose the problem, you can run the same tdsbind -u input shown in [Example: Tdsbind Output for a Directory User Mapped to a Database User](#). If the command produces the following error message, the LdapServerRealm property in the TDGSS user configuration file contains an invalid realm name.

```
tds_bind: Directory error - Invalid Credentials
additional info: SASL(-1): generic failure: realm changed:
authentication aborted
```

You can correct this error by editing the value of the LdapServerRealm property. See [LdapServerRealm \[Deprecated\]](#).

Once the value of the LdapServerRealm is correct, run tdsstestcfg to verify the configuration is correct, run \_tdgssconfig to update the TDGSSCONFIG GDO, and run tpareset if run \_tdgssconfig indicates to do so, which restarts the server and enables the change. If you cannot restart the server, instruct users to enter the correct realm information as part of the logon string. See [Logging on Using LDAP Authentication and Authorization](#).

## About tdsbind Error Output

When tdsbind returns an error, it often appears around output line 8. The errors are usually related to the server configuration. Examine the LDAP configuration to help identify the root cause of any bind failures.

## Example: Tdsbind Output for a Directory User not Mapped to a Database User

```
1 # tdsbind -u drct01
2 Enter LDAP password:
3     LdapGroupBaseFQDN: ou=groups,dc=domain1,dc=com
4     LdapUserBaseFQDN:
5     LdapSystemFQDN: ou=system1,ou=tdat,dc=domain1,dc=com
6     LdapServerName: _ldap._tcp.domain1.com
7     LdapServerPort: 389
8     LdapClientUseTls: yes
```

```

9      LdapClientTlsCACert: /opt/teradata/tdgss/site/certs/ server.pem
10     LdapClientTlsReqCert: demand
11     LdapClientMechanism: simple
12     LdapServiceFQDN: cn=dbssvc,ou=services,dc=domain1,dc=com
13 LdapServicePasswordProtected: yes
14     LdapServicePassword: configured
15     LdapServiceBindRequired: yes
15     LdapClientTlsCRLCheck: none
16 LdapAllowUnsafeServerConnect: yes
17     UseLdapConfig: yes
18     AuthorizationSupported: yes
19
20     FQDN: uid=drct01,ou=principals,dc=domain1,dc=com
21     AuthUser: ldap://dsa1.domain1.com:389/
uid=drct01, ou=principals,dc=domain1,dc=com
22     DatabaseName: drct01
23     Service: domain1
24     Profiles: profile1
25     Roles: extrole01, extrole02, extrole03

```

The table explains the example:

Line Number	Description
<b>tdsbind Input</b>	
1 tdsbind -u drct01	Requests that LDAP authenticate directory user drct01.
2 Enter LDAP password	The example does not specify the -w option (user password) so tdsbind prompts for a password. The user running the command enters the password, and then presses the Enter key.
<b>tdsbind Output</b> <b>Note:</b> Where a line of output is the value of an LDAP property, tdsbind uses the currently configured value for the corresponding property. If the tdsbind command uses a variable that corresponds to an LDAP property (for example, -B, -S, and -h, or a -O list), the command line value overrides the configured value.	
3 LdapGroupBaseFQDN: ou=groups, dc=domain1,dc=com	The FQDN of the group object when directory groups are mapped to Teradata Vantage roles.
4 LdapUserBaseFQDN	The FQDN of a directory object that contains user objects.
5 LdapSystemFQDN: ou=system, ou=Standard Systems,dc=domain1,dc=com	The FQDN of the tdatSystem object that is the parent of the structure used for LDAP user authorization
6 LdapServerName: _ldap._tcp. domain1.com	The name of the LDAP directory server.

## 14: Testing Directory Authentication and Authorization Setup

Line Number	Description
7 LdapServerPort: 389 [Deprecated]	<p>The tdsbind command sets the LDAP server port (-p) to the default. When the command contains a value, it override the default.</p> <p><b>Note:</b> Separate specification of the LDAP server port is deprecated and should not be used. Instead, you can include the port designation as part of specifying the LDAPServerName value. See <a href="#">LdapServerName</a>.</p>
8 LdapClientUseTls: yes	By default, tdsbind uses the value configured for the corresponding LDAPClientUseTls mechanism property. The yes value indicates that TLS is used to establish the connection to the directory server.
9 LdapClientTlsCACert:/opt/teradata/tdgss/site/certs/server.pem	Identifies a file containing the directory server CA certificate.
10 LdapClientTlsReqCert: demand	Specifies what checks the system performs on directory server certificates (if any), in a TLS-protected session. Demand specifies that Vantage asks the directory server for a certificate. If it does not provide a certificate, or if it provides an invalid certificate, the connection terminates.
11 LdapClientMechanism: simple	The LDAP binding style enabled on the system.
12 LdapServiceFQDN: cn=dbssvc, ou=services,dc=domain1,dc=com	The DN of a bindable object in the directory, which represents the service or application that requires binding.
13 LdapServicePasswordProtected: yes	Indicates whether the password for the service bind was stored in encrypted form during configuration.
14 LdapServicePassword: configured	The password for the service bind, if required
15 LdapServiceBindRequired: yes	Indicates whether LDAP requires the service, that is Teradata Vantage, must authenticate itself to the directory
16 LdapClientTlsCRLCheck: none	<p>Indicates how the authentication mechanism should use the Certificate Revocation List (CRL) of the CA to verify that the server certificates are not revoked.</p> <p>None specifies that no checks are performed.</p>
17 LdapAllowUnsafeServerConnect: yes	Indicates whether Vantage is allowed to operate with a directory server running software that does not support IETF RFC 5746-compliant connections.
18 UseLdapConfig: yes	Indicates whether TDGSS uses mechanism properties configured in the alternate <LdapConfig> section rather than the base mechanism property values.
19 AuthorizationSupported: yes	Determines whether the LDAP user is authorized database privileges in the directory.
20	Not applicable

Line Number	Description
21 FQDN: uid=drct01,ou=principals,dc=domain1,dc=com	The directory user FQDN. If the bind operation is successful, the output displays the FQDN. If the bind operation was unsuccessful, an error message appears at this point and tdsbind exits
22 AuthUser: ldap://dsa1.domain1.com:389/uid=dirUser1,ou=principals,dc=domain1,dc=com	The directory user global unique identifier (GUID), if the directory supports GUIDs
23 DatabaseName: dirUser1	The AuditTrailId for the directory user. The name of the user after rewriting it based on the configuration in the <Canonicalization> section of the <LdapConfig>.
24 Service: local	The name of the service that authenticated the user. This field is blank if the mechanism authorizes the user.
25 Profiles: profxu1	The value for this attribute appears only if the directory user is mapped to one or more profiles.
26 Roles: extrole01xu1, extrole02xu1, extrole03xu1	The value for this attribute appears only if the group that contains the directory user is mapped to one or more roles.
27 Users: perm01	The database user object to which the directory principal is mapped.

## Example: Tdsbind Output for a Directory User Mapped to a Database User

Assumptions:

```

1 # tdsbind -u diperm02
2 Enter LDAP password:
3     LdapGroupBaseFQDN: ou=groups,dc=domain1,dc=com
4     LdapUserBaseFQDN:
5     LdapSystemFQDN: ou=system,ou=tdat,dc=domain1,dc=com
6     LdapServerName: _ldap._tcp.domain1.com
7     LdapServerPort: 389
8     LdapClientUseTls: yes
9     LdapClientTlsCACert: /etc/openssl/certs/server.pem
10    LdapClientTlsReqCert: demand
11    LdapClientMechanism: simple
12    LdapServiceFQDN: cn=dbssvc,ou=services,dc=domain1,dc=com
13 LdapServicePasswordProtected: yes
14    LdapServicePassword: configured
15    LdapServiceBindRequired: yes
16    LdapClientTlsCRLCheck: none
17 LdapAllowUnsafeServerConnect: yes
18    UseLdapConfig: yes
19    AuthorizationSupported: yes
20
21    FQDN: uid=drct02,ou=principals,dc=domain1,dc=com
22    AuthUser: ldap://
dsa1.domain1.com:389/uid=diperm01,ou=principals,dc=domain1,dc=com

```

```

23      DatabaseName: diperm01
24      Service: local
25      Profiles: prof01
26      Roles: extrole01, extrole02, extrole03
27      Users: perm01

```

For a mapped directory user, lines 1 through 12 have meanings similar to those for the unmapped directory user shown in [Example: Tdsbind Output for a Directory User not Mapped to a Database User](#). For a mapped directory user, tdsbind returns line 13. If the directory user maps to a permanent Teradata Vantage user, the permanent user name appears on line 13.

tdgssauth can also be used to test the LDAP settings:

```
tdgssauth -m ldap -u diperm02
```

## Using BTEQ to Verify Directory User Mapping

To test the new directory configuration using BTEQ, set up a script that contains a valid directory logon, for example:

```

.logmech ldap
.logon tdpid/dirusername,dirpassword
.quit

```

### Note:

To test maps to multiple Vantage users or profiles, also include the associated .logdata syntax.

See [Logging on Using LDAP Authentication and Authorization](#) for logon format options.

If all directory integration tasks are complete and correct, a properly formatted logon should succeed. If the logon fails, one of the following may be the cause of the failure:

- One or more of the logon parameters are not in the directory information tree.
- One or more of the logon parameters are not mapped to the directory user.
- The directory failed to authenticate the authcid and password.
- The LDAP mechanism configuration is not correct.

## Common BTEQ Error Messages and Related Directory Setup Problems

Error Message	Problem	Solution
CLI error: External authentication failed by gateway.	LDAP cannot authenticate the user with the information provided in the logon.	A common cause of failure is that the LDAP logon contains a Teradata Vantage username and /or password.



Error Message	Problem	Solution
Gateway error: User, account, or password information is invalid.	The user, account, or password is invalid.	Confirm that the directory username and password are entered correctly, and that the account specification, if used, is valid. If the logon appears to be correct, you can use <code>tdgssauth</code> to identify other possible problems. See <a href="#">Working with tdgssauth</a> .
DBS error: User identification is not authorized.	The Vantage user ( <code>tdatUser</code> object) mapped to the directory user does not exist in the database.	Do either one of the following: <ul style="list-style-type: none"> <li>• Remap the directory user.</li> <li>• Create the user in the database.</li> </ul>
DBS error: Profile 'xxx' does not exist.	The mapped Vantage profile ( <i>profile_name</i> ) does not exist. <ul style="list-style-type: none"> <li>• The profile may not exist in the database or the directory.</li> <li>• The profile may be misnamed in the directory.</li> <li>• The logon user may not be mapped to the profile.</li> </ul>	Take one of these actions depending on the details of the problem, if the: <ul style="list-style-type: none"> <li>• Profile does not exist, create it in the database.</li> <li>• Profile directory object does not exist, create the object in the directory.</li> <li>• Profile directory object is misnamed, rename the profile object in the directory.</li> <li>• User is not mapped to the profile, remap the user to the profile.</li> </ul>

## Working with Ldapsearch

Ldapsearch is a standard LDAP tool, which allows you to explore the directory assignments for a user to help resolve questions about failed logon attempts or difficulties with data access.

### Note:

TDGSS includes a Teradata version of Ldapsearch based on the OpenLdap standard. Non-Teradata versions of Ldapsearch may not function as described in this document.

## Running Ldapsearch

You can run Ldapsearch from the Teradata Vantage command prompt by entering the following Ldapsearch attributes, in any order:

### Syntax for Use of DIGEST-MD5 Binding [Deprecated]

```
ldapsearch -Y DIGEST-MD5 -U user -w password -s scope [-b basedn] [-H scheme://host:port/] [filter] [ attribute [...] ]
```

**Note:**

The DIGEST-MD5 authentication protocol used by LDAP is deprecated. Teradata strongly recommends you use simple binding with TLS protection, and stop using DIGEST-MD5.

## Syntax for Use of Simple Binding

```
ldapsearch -x -D user -w password -s scope [-b basedn] [-H scheme://host:port/]  
[filter] [ attribute [...] ]
```

## Working with Ldapsearch Options

Option or Argument	Description
-x	Specifies that the search uses simple binding, if offered by the directory. You cannot use the -x option with the -Y option.
-D <i>user</i>	Passes the user identity when you specify -x (simple binding). The name format you use may depend on the directory type. You can use: <ul style="list-style-type: none"> <li>The FQDN for the user on all certified directories.</li> <li>user@domain on Active Directory</li> <li>The contents of the userPrincipalName attribute for the user on ADAM or AD LDS</li> </ul>
-Y DIGEST-MD5 [Deprecated]	Specifies that the search uses DIGEST-MD5 binding, if offered by the directory. You cannot be use the -Y option with the -x option.  <b>Note:</b> The DIGEST-MD5 authentication protocol used by LDAP is deprecated. Teradata strongly recommends you use simple binding with TLS protection, and stop using DIGEST-MD5.
-U <i>user</i> [Deprecated]	Passes the user identity when the search uses a DIGEST-MD5 bind, that is, when you specify -Y DIGEST-MD5.  <b>Note:</b> The DIGEST-MD5 authentication protocol used by LDAP is deprecated. Teradata strongly recommends you use simple binding with TLS protection, and stop using DIGEST-MD5.
-w <i>password</i>	Specifies the directory user password in the ldapsearch command.
-W	Specifies that the ldapsearch prompts the user for a password.
-R <i>realm</i>	Specifies a SASL realm offered by the directory server. This option is valid only when the directory server offers more than one realm. Set up the -R option similarly to the LdapServerRealm property. See <a href="#">LdapServerRealm [Deprecated]</a> .

Option or Argument	Description
<code>-b basedn</code>	<p>The FQDN of the of the directory object that constitutes the search base, that is, the starting point for the directory search.</p> <p>If you omit this option, the search uses the directory defaults in the <code>ldaprc</code> or <code>.ldaprc</code> file. For more information, go to: <a href="http://www.openldap.org">http://www.openldap.org</a>.</p>
<code>-s scope</code>	<p>Use this option to specify the scope of the search.</p> <ul style="list-style-type: none"> <li>• <code>one</code>: Searches the children of the object identified by the search base (<code>-b</code> option).</li> <li>• <code>base</code>: Searches only the object identified by the search base (<code>-b</code> option).</li> <li>• <code>sub</code>: Specifies a subtree search (or deep search). A deep search includes any object names contained in the <code>-b</code> option, and any other objects included in the subtree named by the search base.</li> </ul> <p><b>Note:</b></p> <p>If you use the root node as the search base (the usual default) with a scope of <code>sub</code>, the command searches the entire directory.</p>
<code>-H scheme://host:port/</code>	<p>Identifies the URI for the LDAP directory server.</p> <p>See <a href="#">LdapServerName</a> for valid settings.</p>
<code>-Z</code>	<p>Requests that the search use TLS protection for the search authentication token exchange. If TLS is not available, <code>-Z</code> returns an error message, but the search continues (without protection).</p>
<code>-ZZ</code>	<p>Same as <code>-Z</code>, but if TLS is not available, the search aborts.</p>
<code>filter</code>	<p>Specifies the filter for the search, and is approximately equivalent to an SQL WHERE clause.</p> <p>You must use a unique syntax to specify a filter, in accordance with IETF RFC 2254. Go to: <a href="http://www.faqs.org/rfcs/rfc2254.html">http://www.faqs.org/rfcs/rfc2254.html</a>.</p> <p>If you do not specify a filter, the search uses <code>'(objectClass=*)'</code>.</p> <p><b>Note:</b></p> <p>All search filters must begin with a <code>'('</code> character, which is not legal in an attribute name.</p>
<code>attr1 [attr2 ...]</code>	<p>A space separated list of one or more optional arguments that tell the server the names of attributes it must return. If you do not specify any attributes, the search returns all user defined attributes for each object that matches the search criteria, for most directory types. Specify:</p> <ul style="list-style-type: none"> <li>• <code>'*'</code> to include all normal attributes.</li> <li>• <code>'+'</code> to include all operational attributes, that is, those attributes the server uses to manage the object.</li> <li>• <code>'1.1'</code> to return no attributes.</li> </ul> <p>For some directory types, such as OpenLDAP, you can use <code>'+'</code> and <code>'*'</code> to request all user attributes and all system attributes, respectively.</p> <p>A search always returns the FQDN of the object.</p>

## About Ldapsearch Attributes and System Names

### Attributes

Some supported directories make a distinction between operational and user attributes. Servers that draw such distinctions frequently specify special attribute names that mean “all user attributes” and “all operational attributes.” Common operational attributes may include creation and modification times.

Sun Java Directory server, Active Directory, ADAM, and AD LDS do not support special attribute names.

Other servers may allow the name “\*” to mean “all user attributes” and “+” to mean “all operational attributes.”

When you use `ldapsearch` to examine an entire OpenLdap RootDSE object, you must use the following commands:

When run from this operating system	Use this command
Windows	<code>ldapsearch -H ldap://host:port/ -b "" -s base *</code>
Linux	<code>ldapsearch -H ldap://host:port/ -b "" -s base \*</code>

### System Names [Deprecated]

When you use DIGEST-MD5 binding and specify the DNS name of the directory server in an `ldapsearch` command, the system that runs the `ldapsearch` command must resolve the DNS name of the directory in the same way the directory server resolves the name, for example:

If directory server `dirsvr1` resolves to `dirsvr1.teradata.com` on the directory server system, the `dirsvr1` must also resolve to `dirsvr1.teradata.com` on the system where the `ldapsearch` runs. If the Teradata Vantage server does not resolve it in the same way, the directory authentication fails.

#### Note:

The DIGEST-MD5 authentication protocol used by LDAP is deprecated. Teradata strongly recommends you use simple binding with TLS protection, and stop using DIGEST-MD5.

## Finding User Information with Ldapsearch

For directory server `esroot` running Active Directory, ADAM, or AD LDS, you can run `ldapsearch` to find information about a configured directory user, `drc01`.

#### Note:

`ldapsearch` returns output in standard LDIF format, in accordance with IETF RFC 2849.

For information on the LDIF format, go to: <http://www.faqs.org/rfcs/rfc2849.html>.

## Step 1: Obtain the defaultNamingContext

Before you can use `ldapsearch` to gather information about a directory user, you must first determine the directory `defaultNamingContext`, to help define the location of the `tdatUsers` container that contains user `drct01`. You can find the `defaultNamingContext` in the `RootDSE` object.

## Search Input

```
C> ldapsearch -x -b "" -s base -H ldap://k1/ defaultNamingContext
```

## Search Output

```
dn:
defaultNamingContext: DC=esrootdom,DC=esdev,DC=tdat
```

## Explanation of the defaultNamingContext Search

Search Criteria	Description
<b>ldapsearch Input</b>	
-b ""	Specifies a search base (-b) and a scope (-s) for the RootDSE object.
-s base	
-H ldap://host:port/	Identifies the URI for the Ldap server. For details, see <a href="#">Running Ldapsearch</a> and <a href="#">LdapServerName</a> .
defaultNamingContext	Specifies an attribute name to limit the search. If you do not specify an attribute name, the search returns values for all available attributes.
<b>ldapsearch Output</b>	
dn:	The FQDN of the object. Since the search base is "" and the scope is base, dn returns the RootDSE object, the name for which is a zero length string.
defaultNamingContext: DC=esrootdom, DC=esdev, DC=tdat	The value of the schemaNamingContext attribute.

## Step 2: Search for User *drct01*

You must search the Directory users container to find user *drct01*.

### Search Input [Deprecated]

The filter in this example is based on Active Directory security account manager (SAM) objects. This and other search criteria may vary depending on the directory you search.

```
C> ldapsearch -H ldap://host:port/ -Y DIGEST-MD5 -U drct01 -b
"CN=Users,DC=esrootdom,DC=esdev,DC=tdat" -s one
"(sAMAccountName=drct01)"
```

#### Note:

The DIGEST-MD5 authentication protocol used by LDAP is deprecated. Teradata strongly recommends you use simple binding with TLS protection, and stop using DIGEST-MD5.

### Search Output

The search returns information on user *drct01*:

```
Password:
dn: CN=John Doe,CN=Users,DC=esrootdom,DC=esdev,DC=tdat
objectClass: top
objectClass: person
objectClass: organizationalPerson
objectClass: user
cn: John Doe
sn: Doe
givenName: John
distinguishedName: CN=John Doe,CN=Users,DC=esrootdom,DC=esdev,DC=tdat
instanceType: 4
whenCreated: 20040605220928.0Z
whenChanged: 20040728221734.0Z
displayName: Directory User 1
uSNCreated: 50268
memberOf: CN=xu1,OU=groups,OU=testing,DC=esrootdom,DC=esdev,DC=tdat
uSNChanged: 315083
name: Directory User 1
objectGUID: f?-å=çAÆ!¶S++§
```

```

userAccountControl: 512
badPwdCount: 0
codePage: 0
countryCode: 0
badPasswordTime: 127337313454062500
lastLogoff: 0
lastLogon: 127355266545781250
pwdLastSet: 127309469682812500
primaryGroupID: 513
objectSid: ?
accountExpires: 9223372036854775807
logonCount: 140
sAMAccountName: drct01
sAMAccountType: 805306368
userPrincipalName: drct01@esrootdom.esdev.tdat
objectCategory:
CN=Person,CN=Schema,CN=Configuration,DC=esrootdom,DC=esdev,DC=tdat
lastLogonTimestamp: 127355266545781250
tdatProfileMemberOf:
CN=profxu1,CN=profiles,CN=end2end,CN=tdat,OU=testing,DC=esrootdom,DC=esd
ev,DC=tdat

```

## Explanation of the Search for User drct01

Search Criteria	Description
<b>ldapsearch Input</b>	
-H ldap://server:port/	Identifies the URI for the LDAP server. For details, see <a href="#">Running Ldapsearch</a> .
-U drct01	Names the directory user authenticated in the search.
-b "CN=Users, DC=esrootdom, DC=esdev, DC=tdat"	Identifies the search base. In the example, the users container appears in the default naming context. User drct01 and all Active Directory users are all children of this container.
-s one	Requests a search of only children of the object named in the -b option.
"(sAMAccountName=drct01)"	The search filter. Limits the search to the object where the sAMAccountName attribute contains drct01.
<b>ldapsearch Output</b>	
Password:	Prompts for the directory password of the user named in the -u option.

Search Criteria	Description
dn: CN=John Doe CN=Users, DC=esrootdom,DC=esdev,DC=tdat	The distinguished name of the user drct01. This object is returned as a result of the search filter, not the bind of user drct01.
objectClass: top	These are common directory user entries, shown for reference, which may or may not appear in your directory.
objectClass: person	
objectClass: organizationalPerson	
objectClass: user	
cn: John Doe	
sn: Doe	
givenName: John	
distinguishedName: CN=John Doe,CN=Users, DC=esrootdom, DC=esdev,DC=tdat	
instanceType: 4	
whenCreated: 20040605220928.0Z	
whenChanged: 20040728221734.0Z	
displayName: Directory User1	
uSNCreated: 50268	
memberOf: CN=xu1,OU=groups, OU=testing, DC=esrootdom, DC=esdev, DC=tdat	Lists the groups in which the user has membership. The data contained in this attribute can help you to search the group for roles assigned to the user, that is, any role that appears in a tdatRoleMemberOf attribute in the group object identified by the data in this attribute. The tdatRoleMemberOf attribute in the group object is specific to Active Directory.
uSNChanged: 315083	These are common directory entries, shown for reference, that may or may not appear in your directory.
name: Directory User 1	
objectGUID: £?=å=çAÆ ¶S++§	
userAccountControl: 512	
badPwdCount: 0	
codePage: 0	
countryCode: 0	



Search Criteria	Description
badPasswordTime: 127337313454062500	
lastLogoff: 0	
lastLogon: 127355266545781250	
pwdLastSet: 127309469682812500	
primaryGroupID: 513	
objectSid:?	
accountExpires: 9223372036854775807	
logonCount: 140	
sAMAccountName: drct01	
sAMAccountType: 805306368	
userPrincipalName: drct01@esrootdom.esdev.tdat	
objectCategory: CN=Person, CN=Schema, CN=Configuration, DC=esrootdom, DC=esdev, DC=tdat	
lastLogonTimestamp: 127355266545781250	
tdatProfileMemberOf: CN=profxu1, CN=profiles, CN=end2end, CN=tdat, OU=testing, DC=esrootdom, DC=esdev, DC=tdat	<p>Directly locates the Teradata profile objects that describe the mapped user profiles. This attribute only appears in Active Directory.</p> <p>If a directory user is mapped to a Vantage user, a row containing tdatUserMemberOf attribute is always present. This attribute identifies the tdatUser object that defines the Vantage user to which the directory user is mapped.</p>

## Determining the schemaNamingContext Value

Before you install Teradata directory schema extensions on Active Directory, ADAM, and AD LDS, you must first determine the value to use for the schemaNamingContext.

### Search Input

```
C> ldapsearch -b "" -s base -H ldap://host:port/k1/ schemaNamingContext
```

## Search Output

```
dn:
schemaNamingContext:
CN=Schema,CN=Configuration,DC=k1dns,DC=systemone,DC=com
```

## Explanation of Search for a schemaNamingContext Value

Search Criteria	Description
<b>ldapsearch Input</b>	
-b ""	Specifies a search base (-b) of "" and a scope (-s) of base to obtain the RootDSE object.
-s base	
-H ldap://host:port/K1	Specifies the URI of the directory server.
schemaNamingContext	Specify an attribute name to limit the attribute values the search can return, otherwise it returns values for all available attributes.
<b>ldapsearch Output</b>	
dn:schemaNamingContext	Indicates that the output is the distinguished name of the schema naming context.
schemaNamingContext CN=Schema, CN=Configuration, DC=k1dns, DC=k1dns, DC=systemone, DC=com	The value for the schemaNamingContext attribute.

## About Other LDAP Tools

In addition to `tdgssauth` and `ldapsearch`, TDGSS includes several other commonly available directory tools, customized by Teradata to help you administer directories that interface with Teradata Vantage. Even though these tools are OpenLdap-compliant, they may differ slightly from tools with similar names supplied with your LDAP-compliant directory.

LDAP Tool	Description
<code>ldapcompare</code>	Compares directory objects to a search filter and returns a true/false assessment of whether the object satisfies the filter criteria.
<code>ldapdelete</code>	Removes objects from the directory.
<code>ldapmodify</code>	Adds new objects and modifies existing objects in the directory.
<code>ldapmodrdn</code>	Renames objects in the directory.

LDAP Tool	Description
ldappasswd	Changes directory passwords. <b>Note:</b> This tool works only with directories that support the PASSWORD-MODIFY extended operation. All supporting directories contain 1.3.6.1.4.1.4203.1.11.1 for the value in the RootDSE supportedExtension attribute.
ldapsearch	Finds objects in the directory.
ldapwhoami	Binds to the directory and discovers a user directory identity. All supporting directories contain 1.3.6.1.4.1.4203.1.11.3 for the value of the RootDSE supportedExtension attribute.

You can find the LDAP tools in the TDGSS bin directory (/opt/teradata/tdgss/bin).

Because the LDAP tools are industry standards, this document provides minimal guidance on how to use them. For additional usage information, go to the OpenLdap website at: <http://www.openldap.org>.

## Directory Tool For Active Directory, ADAM, and AD LDS

You can also use the LDIFDE tool supplied with Windows for directory maintenance on Active Directory, ADAM, and AD LDS.

# LDAP Binding Options

Teradata binding features are optional for the default Teradata authentication and DIGEST-MD5 binding.

---

**Note:**

The DIGEST-MD5 authentication protocol used by LDAP is deprecated. Teradata strongly recommends you use simple binding with TLS protection, and stop using DIGEST-MD5.

---

## Evaluation and Implementation Process

1. Determine the need to implement simple binds in place of the default DIGEST-MD5 binding scheme. See:
  - [Using DIGEST-MD5 Binds \[Deprecated\]](#)
  - [Using Simple Binds](#)
2. Change the binding scheme if required. See [Implementing Simple Binds](#).
3. Evaluate the need for service binds, that is, where the directory authenticates the database system or Unity server before LDAP authenticates the user. See [Using Service Binds](#).
4. Implement service binds, if required. See [Working with Service Binds](#).
5. For sites that allow the use of anonymous service binds, you can optionally configure anonymous service binds, although this is not recommended. See [Using Anonymous Binds](#).

## Using DIGEST-MD5 Binds [Deprecated]

### About DIGEST-MD5 Binds

DIGEST-MD5 is the default binding scheme for LDAP authentication.

---

**Note:**

The DIGEST-MD5 authentication protocol used by LDAP is deprecated. Teradata strongly recommends you use simple binding with TLS protection, and stop using DIGEST-MD5.

---

### Implementing DIGEST-MD5 Binds

To use DIGEST-MD5 binding, set the `LdapClientMechanism` property to “sasl/digest-md5” (the default). See [LdapClientMechanism](#).

## Using Simple Binds

### About Simple Binds

You may need to use simple binding, the alternative to the default DIGEST-MD5 binding, if:

- Site policy requires simple binding.
- The directory does not support DIGEST-MD5 binding.
- Your site uses an LDAPv3-compliant directory that is not certified by Teradata, for example, ApacheDS, IBM z/OS, or Oracle Internet Directory.

### Implementing Simple Binds

1. Set the `LdapClientMechanism` property to “simple” on Teradata Vantage nodes and on the Unity server, if used. See [LdapClientMechanism](#).

---

**Note:**

If you use [Option 3: Non-LDAP External Authentication with Directory Authorization](#), TDGSS ignores the setting for this property and automatically implements a simple bind.

---

2. When you implement simple binds and the directory does not support the use of simple user names in the logon string for the bind DN, do one of the following:
  - If LDAP can create the user DN from the simple user name without searching the directory, you can employ an identity map as a template to substitute data from the simple username and create the DN. See [Using Identity Mapping](#).
  - If the directory contains a divergent user location scheme, implement the identity search option, which allows Vantage to search the directory for the DN. See [Using Identity Searches](#).

---

**Note:**

The identity search option also requires configuration of service binding. See [Using Service Binds](#).

---



---

**Note:**

Teradata strongly recommends that sites using simple binds also configure TLS protection, as shown in [Using TLS with a Directory Server](#).

---

## Using Service Binds

### About Service Binds

A service bind requires that the directory authenticate a directory client (or service), such as Teradata Vantage or the Unity server, if used, before the directory authenticates and/or authorizes a directory user to access the service.

Service binds are sometimes required by site security policy, or when the authentication mechanism specifies use of:

- Directory authorization, when authentication is by a non-LDAP mechanism. See [Option 3: Non-LDAP External Authentication with Directory Authorization](#).
- Identity searches (LDAP mechanism only). See [Using Identity Searches](#).

If you do not configure the LDAP mechanism properties that support service binds, LDAP creates an anonymous bind. See [Using Anonymous Binds](#).

---

#### Note:

Anonymous binds fail if the directory disallows anonymous binds.

---

### Working with Service Binds

Service binds occur between the username submitted by the service (either Teradata Vantage or Teradata Unity) and a bindable directory object. All LDAPv3-compliant directories support bindable objects. Requirements for bindable objects vary among directories, but all bindable objects include a userPassword attribute.

- You can use Teradata system-level objects for service bindable objects, for example:
  - Directories that use Teradata schema extensions can use a tdatSystem object.
  - Directories that use only native directory schema can use a system object.

Do not use existing Teradata system-level objects, the parents of other Teradata directory objects, as service bindable objects. Create new system-level objects for use in service binds, or use other bindable directory objects, according to your directory policy.

- For sites with multiple Teradata Vantage systems:, create a bindable object for each system, and for the Unity server, if users log on through Unity.
  - If site policy requires unique authentication of each service, create a bindable object for each system, and for the Unity server, if used. The LdapServiceFQDN property on each system names its unique bindable object.
  - If site policy views multiple Vantage systems and the Unity server as a single network service, then the individual systems can all point to a single common bindable object in the directory.
  - If systems are authenticated in different directories, then each directory must contain the necessary bindable object(s).

- Teradata requires only that the service DN for the object is bindable.

## LDAP Mechanism Properties that Support Service Binds

Evaluate all the LDAP mechanism properties that support service binds. You may need to configure some or all of them when implementing service binds on your system.

### Note:

Configure the TdgssUserConfigFile.xml on each Teradata Vantage system served by the directory and the TdgssUnityConfig.xml on the Teradata Unity server, if used.

Property	Setting	Property Value Setting
LdapServiceBindRequired	Yes/No	Sets the requirement for a service bind. <ul style="list-style-type: none"> <li>• A yes value means that TDGSS always performs a service bind.</li> <li>• A no value (the default) means that TDGSS performs a service bind only if IdentitySearch elements are present in the configuration.</li> </ul>
LdapServiceFQDN	Distinguished name	Identifies the bindable object in the directory that represents the service identity, that is, a Teradata Vantage system or Unity server.
LdapServicePassword	String	If your site security policy requires a password for the service FQDN, configure a password as the value of this property.
LdapServicePasswordFile	String (fully qualified file name)	Name of a file that contains a list of encrypted, base64 encoded service passwords, one per line. Allows for changing the LDAP service password without requiring a database restart. See <a href="#">LdapServicePasswordFile</a> .
LdapServicePasswordProtected	Yes/No	Enables encryption for the LDAP service password, if used. <ul style="list-style-type: none"> <li>• Yes (the default) means that TDGSS stores the LdapServicePassword is stored in encrypted form.</li> <li>• No means that TDGSS stores the LdapServicePassword in plain text.</li> </ul>

For detailed configuration information, see [LDAP Binding Properties \[Deprecated\]](#).

## Service Bind Configuration Process

1. Create a tdatSystem object, or other bindable object, in the directory to represent the service. See:
  - [Creating a Bindable Object on Sun Java Directory Server and OpenLdap](#)

- [Creating a Bindable Object on Active Directory, ADAM, or AD LDS](#)
2. Edit the `TdgssUserConfigFile.xml`. See [Editing TdgssUserConfigFile.xml for Service Binds](#).

**Note:**

For service binds resulting from an identity search, the identity LDAP uses to authenticate the service must have the permission to search the portion of the directory that contains user definitions.

## Creating a Bindable Object for the Service

### Creating a Bindable Object on Sun Java Directory Server and OpenLdap

Directory configuration for a bindable object for the service is identical on Sun Java Directory Server and OpenLdap, because both support the `simpleSecurityObject` class of directory objects.

Use the following example to configure a bindable directory object:

```
dn: cn=dataone,cn=tdat,ou=production,dc=division,dc=company,dc=com
objectClass: top
objectClass: tdatSystem
objectClass: simpleSecurityObject
cn: dataone
userPassword: password
```

**Note:**

The directory configuration requires only the DN and user password for the object. The `objectClass` information can be in any form that accurately represents a bindable object.

### Creating a Bindable Object on Active Directory, ADAM, or AD LDS

The configuration for a bindable directory object for the service is identical for Active Directory, ADAM, and AD LDS, because these directory types support the `msDS-BindableObject` class of directory objects.

Create a service user in the directory and assign a password to the user. The FQDN of the service is used for the value of the `LdapServiceFQDN` property when configuring the LDAP mechanism. See [Editing TdgssUserConfigFile.xml for Service Binds](#).



**Note:**

The directory configuration requires only the DN and user password for the object. The objectClass information can be in any form that accurately represents a bindable object, and may vary among directories.

## Editing TdgssUserConfigFile.xml for Service Binds

You must edit the TdgssUserConfigFile.xml to enable service binds.

**Note:**

The following procedure includes steps that generates an encrypted version of the service password for use in configuring the TdgssUserConfigFile.xml, which avoids storing the value of LdapServicePassword property in plain text.

1. Modify the TdgssUserConfigFile.xml and set the LdapServiceFQDN property with the bind account DN.
2. Update TDGSSCONFIG.GDO. Run:

```
run_tdgssconfig
```

3. Generate a protected password using the tdspasswd command:
  - a. At the Teradata command prompt, enter:

```
$ tdspasswd -m mechanism
```

where *mechanism* is the authentication mechanism for which you are editing the TdgssUserConfigFile.xml; for example, ldap.

- b. The system prompts you to enter the new password.

```
Enter New password:
Confirm New password:
```

**Note:**

The system does not display the password when you enter it.

- c. After the system confirms the new password, it generates and displays an encrypted version of the password, for example:

```
$ tdspasswd -m ldap
Enter New password:
Confirm New password:
```

```
AV8Jeq2cvjmAjiHgcSrAUoE=
$
```

**Note:**

Only the mechanism you specify in the -m option can use the encrypted password, and only for service binds.

**Note:**

If the LdapServiceFQDN bind account DN is changed, the above steps must be run again, even if the bind account plain text password remains the same.

4. Edit the mechanism to specify a service user and password for the service bind:

```
<Mechanism Name="ldap">
  <MechanismProperties
    ...
    LdapServiceBindRequired="yes"
    LdapServiceFQDN="cn=service_id,ou=services,dc=domain,dc=com"
    LdapServicePassword="encrypted_password"
    LdapServicePasswordProtected="yes"
    ...
  />
</Mechanism>
```

**Note:**

The LdapServicePasswordProtected property is only an indicator of password protection status, and does not enable the protection.

**service\_id**

The CN of the service user object in the directory.

**encrypted\_password**

The encrypted password generated in step 1c (sub-step c of step 1.).

5. Verify the configuration is correct:
  - a. Run `tdgsstestcfg` to test the configuration. It launches a test environment in a new shell that contains the updates to the configuration file.

```
/opt/teradata/tdgss/bin/tdgsstestcfg
```

- b. Run a utility, such as `tdgssauth`, to test the new configuration before you commit the changes to the TDGSSCONFIG GDO.

See [Working with tdgssauth](#).

- c. Exit the test shell:

```
exit
```

- d. Continue editing and testing until the configuration is correct.

- 6. Run the `run_tdgssconfig` utility to send the changes to the TDGSSCONFIG GDO:

```
run_tdgssconfig
```

- 7. If `run_tdgssconfig` indicates that a TPA reset is required, then run `tpareset` from the Teradata Vantage node with the lowest ID number, to activate the changes to the TDGSS configuration.

```
tpareset "use updated TDGSSCONFIG GDO"
```

- 8. Repeat this procedure on the Teradata Unity server, if used. For information about Unity, see *Teradata® Unity™ Installation, Configuration, and Upgrade Guide for Customers*, B035-2523 and *Teradata® Unity™ User Guide*, B035-2520.

---

**Note:**

Service binds configured for LDAP apply it to all external authentication mechanisms.

---

## Using Anonymous Binds

### About Anonymous Binds

Any bind that does not carry a username and password is an anonymous bind. Security policy for some sites may allow anonymous binds, but they are not recommended.

### Configuring Anonymous Binds

Sites that allow anonymous binds and anonymous reads of the directory can use the following example to enable directory authorization without a service ID for LDAP (shown below), KRB5 and SPNEGO.

---

**Note:**

When the service bind is automatic, as shown in [Using Service Binds](#), and the bind is anonymous because the `LdapServiceFQDN` and `LdapServicePassword` are not configured, you do not need to configure the `LdapServiceBindRequired` property.

```
<Mechanism Name="ldap">
  <MechanismProperties
    ...
    LdapServiceBindRequired="yes"
    LdapServiceFQDN=""
    LdapServicePassword=""
    ...
  />
</Mechanism>
```

---

# TLS Protection Options

See:

- [Using TLS with a Directory Server](#)
- [Using TLS with Client to Database Connections](#)

## Using TLS with a Directory Server

If your site policy requires simple binds for directory authentication of database users, as shown in [LDAP Binding Options](#), consider that simple binding introduces security risks:

- Network connections to the database are vulnerable to man-in-the-middle attacks.
- Communications between the directory and the database transmit user identities and passwords across the network in plain text.

To eliminate these security risks, Teradata strongly recommends that you configure TLS protection for all systems that use simple binding.

## Protection Implementation Process

1. Enable TLS protection. See [Using Basic TLS Protection](#).
2. Review advanced protection options. See [About Advanced Protection Options](#).
3. If you decide to use any of the advanced options, complete the preparation tasks before configuring the options. See [Preparing to Configure Advanced Protection](#).
  - Check for the presence of the required security certificates. See [Checking the Directory Server Certificates](#).
  - Test the CN value. See [Testing the CN Value](#).

---

**Note:**

You must complete the configuration of basic TLS functions before configuring any of the advanced protection options.

---

4. Configure advanced protection options, if needed:

---

**Note:**

You can use advanced options with either TLS protection.

- [Verifying the Directory Server Certificate Chain](#).
  - [Using Mutual Authentication Between the Directory Server and Teradata Vantage](#).
-

5. Correct protection setup problems. See [Troubleshooting TLS Setup with a Directory Server](#).

## Prerequisites

- Configure the system for directory authentication and/or authorization of users. See [Directory Management of Database Users](#).
- If the system requires simple binding, you must complete simple binding setup before implementing TLS protection options. See [Using Simple Binds](#).

## X.509 Certificates Ownership and Permissions

When you create X.509 certificates or private keys, you must be logged on as root.

To prevent unauthorized overwriting of X.509 certificates and private key files, set the ownership and permissions as follows:

- The certificates and private key files are owned by root and the group is tdtrusted.
- The permissions are set to 640.

For example:

```
-rw-r----- 1 root      tdtrusted    0 May 21 15:07 cert
```

## Using Basic TLS Protection

### About TLS Protection

TLS protection encrypts the directory user ID and password during a bind to an LDAPv3-compliant directory, to prevent man-in-the-middle attacks and other security threats.

Teradata recommends TLS protection when:

- LDAP authentication uses simple binding.
- Kerberos authenticates users, while the directory authorizes user privileges in the database, resulting an automatic service bind (a type of simple bind).

You can configure LDAP protection properties in the LDAP, Kerberos, and SPNEGO mechanisms on Teradata Vantage nodes and on the Unity server, if the AuthorizationSupported property is set to yes. Also see [LDAP Protection Properties](#).

---

#### Note:

If you implement TLS protection you should check, and if necessary reset, the AllowUnsafeServerConnection property in each mechanism that you configure for protection, to ensure optimal database operation with directories that supports IETF RFC 5746. See [LdapClientTlsReqCert](#).

---

For configuration requirements when authentication is set for multiple directory services, see [Creating the <LdapConfig> Section in the TdgssUserConfigFile.xml](#).

## Configuring SSL Protection [Deprecated]

### Note:

*Do not* set the LdapServerPort property to 636. LdapServerPort is deprecated. See [LdapServerName](#) and [LdapServerPort \[Deprecated\]](#).

Configure the LdapServerName property, using an ldaps prefix, to simultaneously enable SSL protection and specify the required port number, 636:

```
<Mechanism Name="ldap">
  <MechanismProperties
    LdapServerName="ldaps://myserver/"
    ...
  />
</Mechanism>
```

## Configuring TLS Protection

To activate TLS protection:

- Set the LdapClientUseTls property to “yes”
- Set the LdapServerName property using an ldap prefix

```
<Mechanism Name="ldap">
  <MechanismProperties
    LdapServerName="ldap://myserver/"
    ..
    LdapClientUseTls="yes"
  />
</Mechanism>
```

## Related Information

For information on valid property settings, see [LDAP Protection Properties](#).

## About Advanced Protection Options

In addition to basic protection of logon information, TDGSS also offers the following advanced options for TLS protected systems:

- Authentication of the directory server by Advanced SQL Engine.
- Authentication of the Advanced SQL Engine nodes, and Unity (if used), by the directory server. This option is useful when site policy requires mutual authentication.

## Preparing to Configure Advanced Protection

Before you configure any of the advanced TLS authentication options, you must prepare the system.

### Preparation Process

1. Check the directory for the presence of security certificates required for advanced protection options. See [Checking the Directory Server Certificates](#).
2. Test the CN value of the certificate subject, to make sure it comes from the indicated IP address. See [Testing the CN Value](#).

### Checking the Directory Server Certificates

Advanced TLS options require the presence of one or more security certificates on the directory server. You can consult the directory manufacturer documentation for certificate installation procedures and requirements.

---

#### Note:

The OpenLDAP standard supports only PEM formatted certificates, and does not allow locking the PEM certificate with a pass phrase.

---

A certificate offered by the directory server must include the fully qualified DNS name of the directory server as a value for the CN attribute of its subject. This requirement is enforced by OpenSSL and OpenLDAP. If the directory server certificate does not conform to this requirement it is not be usable by TDGSS.

### Example: Using openssl to Examine a Certificate

You can examine the certificate to ensure that it conforms, using OpenSSL:

```
openssl s_client -connect server_name[:port] </dev/null
```



**server\_name**

The directory server DNS name.

**port**

[Optional] The port where SSL listens.

Default: 636

This command produces output similar to the following example.

```
dlopldap:/etc/openldap/ssl/certs # openssl s_client -connect localhost:636
</dev/null
CONNECTED(00000003)
depth=0 /C=US/CN=dlopldap.td.teradata.com/emailAddress=dl160010@teradata.com
verify error:num=18:self signed certificate
verify return:1
depth=0 /C=US/CN=dlopldap.td.teradata.com/emailAddress=dl160010@teradata.com
verify return:1
---
Certificate chain
 0 s:/C=US/CN=dlopldap.td.teradata.com/emailAddress=dl160010@teradata.com
  i:/C=US/CN=dlopldap.td.teradata.com/emailAddress=dl160010@teradata.com
---
Server certificate
-----BEGIN CERTIFICATE-----
MIIC5DCCAK2gAwIBAgIJAMvJ4ZlaGSiNMA0GCSqGSIb3DQEBBQUAMFYxCzAJBgNV
BAYTA1VTMSEwHwYDVQQDEExhbkG9wbGRhcC50ZC50ZXJhZGF0YS5jb20xJDAiBgkq
hkiG9w0BCQEFWRsMTYwMDEwQHRlcmFkYXRhLmNvbTAeFw0wODA1MTQxOTA4NDJa
Fw0wOTA1MTQxOTA4NDJaMFYxCzAJBgNVBAYTA1VTMSEwHwYDVQQDEExhbkG9wbGRh
cC50ZC50ZXJhZGF0YS5jb20xJDAiBgkqhkiG9w0BCQEFWRsMTYwMDEwQHRlcmFk
YXRhLmNvbTCBnzANBgkqhkiG9w0BAQEFAAOBjQAwYkCgYEA0QT6CP33QKHsxUWq
jetyHWftS2rnLPmpDK/tKj+05Crv0pMlIXertRHy68swSBLm0w//xiVywWQkuA2w
se8Q80lQlBUJkf19etuNZCrMqjusL3fvSaQlp0pLZLFdICuN+xxuGCqOKuARyISd
1UkwcQ6r9hlPCGHxXrKlghRYRiCawEAAa0BuTCBtjAdBgNVHQ4EFgQUamJoMI9/
TTS59BUTF1EwoEseNAwwgYYGA1UdIwR/MH2AFGpiaDCPf000ufQVExdRFqBLHjQM
oVqkWDBWMQswCQYDVQQGEwJVUzEhMB8GA1UEAxMYZGxvcGxkYXAudGQudGVyYWRh
dGEuY29tMSQwIgYJKoZIhvcNAQkBFhVkbDE2MDAxMEB0ZXJhZGF0YS5jb22CCQDL
yeGZWhkojTAMBgNVHRMEBTADAQH/MA0GCSqGSIb3DQEBBQUAA4GBAIUbbcrUG3Y0
VXdhAj1AKq95qryTeHE1wiDBmEe1UIC5KyyarGW9tA/sxaJ+9X/zrAwP1ymLn5n9
kIJt3gH7HjjrG1qzC7jRVoI0Yl/z+7QUKejGp0ph1gVl4VwFoRzxv+I2vIUuzyF3
dabR1Q0+lqgc1CHC001VEHEAK8v9k6q1
-----END CERTIFICATE-----
subject=/C=US/CN=dlopldap.td.teradata.com/emailAddress=dl160010@teradata.com
```

```

issuer=/C=US/CN=dlopldap.td.teradata.com/emailAddress=dl160010@teradata.com
---
No client certificate CA names sent
---
SSL handshake has read 906 bytes and written 340 bytes
---
New, TLSv1/SSLv3, Cipher is AES256-SHA
Server public key is 1024 bit
Compression: NONE
Expansion: NONE
SSL-Session:
    Protocol   : TLSv1
    Cipher     : AES256-SHA
    Session-ID:
1AC1C0A2959387177910D40DBC9EC81887C4A233D907F31BB8BA7EFA7E7E76D3
    Session-ID-ctx:
    Master-Key:
7C6DE241910B1820882D0833976FE4BF4704F163905C7540569C07D5708218A00C542D1E6846DB6
5E2DE04FD6F0CEC1A
    Key-Arg    : None
    Start Time: 1210794467
    Timeout    : 300 (sec)
    Verify return code: 18 (self signed certificate)
---
DONE

```

Explanation of the output:

- The example shows one certificate and includes the surrounding text.
- The output always shows certificates offered by a directory server between the BEGIN CERTIFICATE and END CERTIFICATE statements. A directory may offer more than one certificate, but only the first certificate is important to the SSL or TLS configuration
- Immediately following the END CERTIFICATE statement are two lines describing the issuer and the subject.
  - The issuer is the identity of the certificate used to sign the certificate.
  - The subject is the identity of the certificate:

```
/C=US/CN=dlopldap.td.teradata.com/emailAddress=dl160010@teradata.com
```

The CN attribute containing the value dlopldap.td.teradata.com identifies the verified certificate.

- When the issuer and subject of a certificate are the same, as in this example, the certificate is self-signed.

- The list of certificates offered by the directory server is called the certificate chain.

## Testing the CN Value

To verify that the CN value is the DNS name of the directory, run the following test.

1. Resolve the value in the CN attribute with the name service on the Teradata Vantage node, and Unity server, if used, by running the following commands:

- `dig tcp -t A directory_DNS_name`

- `nslookup directory_DNS_name`

The commands return the IP address associated with the directory DNS name.

2. Use the IP address to connect to the directory server indicated by the *directory\_DNS\_name*. If the IP address does not connect to the correct directory server, the certificate is unusable.

## Verifying the Directory Server Certificate Chain

You must verify the existence of the proper certificates to avoid the risk of having SSL certificate verification failures when the database tries to verify the authenticity of the directory certificates.

### Verification Process

1. Obtain all certificates in the certificate chain. See [Obtaining the Directory Server Certificate Chain](#).
2. Catenate the certificates into a single file or hash them in a directory. See [Creating the CA Certificate Symlinks](#).
3. Test the connection between the database and the directory or the Unity and the directory. See [Testing the Connection](#).
4. Configure TDGSS to verify the certificates. See [Configuring TDGSS to Use Advanced Binding Options](#).

## Obtaining the Directory Server Certificate Chain

### Preparation

Create an SSL directory structure on each Teradata Vantage node, and on the Unity server, if used. For the database, the structure must be identically named on all nodes.

Teradata recommends using the TDGSS site directory.

The resulting SSL structure consists of two directories:

- site/ssl/cacerts
- site/ssl/certs

These locations correspond to the values required for the following SSL/TLS LDAP mechanism properties, which must search the directories to find needed certificate and key information:

- For LdapClientTlsCACertDir: site/ssl/cacerts
- For LdapClientTlsCert and LdapClientTlsKey: site/ssl/certs

## When the Directory Supports TLS Only

If the directory server supports only TLS, then you must obtain a copy of the certificates directly from the site security administrator and place the copies in:

- the TDGSS site/ssl/cacerts directory on all Teradata Vantage nodes.
- See *Teradata® Unity™ Installation, Configuration, and Upgrade Guide for Customers*, B035-2523 for the recommended location on Unity servers.

Be sure to retain the BEGIN CERTIFICATE and END CERTIFICATE statements, which identify the beginning and end of each certificate.

## When the Directory Supports SSL

If the directory server supports either SSL only, or SSL and TLS, you can use the following procedures to obtain the server certificate chain from each directory server instance that Teradata Vantage, and Teradata Unity (if used) must access. The procedure collects the certificates into a directory and then creates hashes for them.

For each directory server to which Teradata Vantage must connect, run the following command from the Vantage command prompt:

```
openssl s_client -connect server_name[:port] -showcerts </dev/null
```

### ***server\_name***

The directory server DNS name.

### ***port***

[Optional] The port where SSL listens.

Default: 636

The command produces output that shows the certificate chain.

## Example: Certificate Chain

The example shows how to obtain a self-signed certificate from the directory server.

```
openssl s_client -connect server_name:port -showcerts </dev/null
```

Results:

```
CONNECTED(00000003)
depth=1 /C=US/CN=YaST Default CA (dlopldap)/emailAddress=postmaster@site
verify error:num=19:self signed certificate in certificate chain
verify return:0
---
Certificate chain
 0 s:/C=US/CN=dlopldap.site/emailAddress=postmaster@site
  i:/C=US/CN=YaST Default CA (dlopldap)/emailAddress=postmaster@site
-----BEGIN CERTIFICATE-----
MIIEUTCCAzmGAwIBAgIBATANBgkqhkiG9w0BAQUFADBSMQswCQYDVQQGEwJVUzEj
MCEGA1UEAxMaWWFTVCBEZWZhdWx0IENBICkbbG9wbGRhcCkxHjAcBgkqhkiG9w0B
CQEWd3Bvc3RtYXN0ZXJAc2l0ZTAeFw0wNzA4MDMxMjQwNTdaFw0wODA4MDIxMjQw
NTdaMEUxCzAJBgNVBAYTA1VTMRYwFAYDVQQDEw1kbG9wbGRhcC5zaXR1MR4wHAYJ
KoZIHvcNAQkBFg9wb3N0bWZzdGVyQHNpdGUwggEiMA0GCSqGSIb3DQEBAQUAA4IB
DwAwggEKAoIBAQDyBetFqZE5FFaeKc5n220KWcXFa1ys9ZOyHDwjVtOvm2Rgd/77
KM8TOsFq5HyUCUYzYwdC07VuBU3u2qgAleY/0nN+pKP89ohY/rYQbsblHA8QrXVP
LmBuKM3kN/MGyN0bKFYrnfAfY7bU1hreyNbcLdUZAmfunYX9YUk4lX8eOHRuMDgp
VMw+L8I0c80PtjwvQLDpjw3HRTch3HQLSK3+vCA9WQFmJmHuj5aZiFqqUiCRnOmW
zdlbJ9bolStln1luK3+ZYs5vGriR0k303TEZgRdBXcoPFAjVzLY3YjXg4PLwePQq
eeZjzWNgQyBasYoKnZ6L/ysspzrUlgBTvbTtAgMBAAGjggE9MIIBOTAJBgNVHRME
AjaAMADAGCWCgsAGG+EBDDQjFiFZYVNUIEdlbmVyYXRlZCBTZXJ2ZXIga2VydGlm
aWNhdGUwEQYJYIZIAYb4QgEBBAQDAgZAMAsGA1UdDwQEAwIFoDAdBgNVHQ4EFgQU
GuSiFwDmaqtRK2fNeqoVFSRd2aEwgYIGA1UdIwR7MHMAFGEVkfLO9/CKNHdY/FQ8
Smu+cF2uoVakVDBSMQswCQYDVQQGEwJVUzEjMCEGA1UEAxMaWWFTVCBEZWZhdWx0
IENBICkbbG9wbGRhcCkxHjAcBgkqhkiG9w0BCQEWd3Bvc3RtYXN0ZXJAc2l0ZTAaBgNVHRIEEzAR
gQ9wb3N0bWZzdGVyQHNpdGUwDQYJKoZIhvcNAQEFBQADggEBACiEMDMFZq2ZfBHh
9POKRJ9867B0b0Tgmzk/Iw0WT+MbFR5m0a04Ke36tQclukYSkXxXryrwk+pwDdo4
SXXXf6+cgUT+/te1ifCmjEq3o7zTi1E7wYoXz8q93j8PUjNhhfNfKlfVa4tapwhS
z0woT9ie+Bltt8zdbwtBx13iYolHqgxEv2jnKhFSKnW01XlPSyGEMpY/ZNitwA4N
5Qs3i0j6U+oo3xHkB/dDHgSP6DGurc+jo9doLiFyVX9AUUCfLb0/U3sCBneVVe1J
d31X/QEV/5Njrpj1ZrrnpF+XgUbSLqN5vYhL6pDbuk8Kv90ZPtSKjhobNYD5k1lt
nmqVdrc=
-----END CERTIFICATE-----
1 s:/C=US/CN=YaST Default CA (dlopldap)/emailAddress=postmaster@site
```

```

i:/C=US/CN=YaST Default CA (dlopldap)/emailAddress=postmaster@site
-----BEGIN CERTIFICATE-----
MIIEaDCCA1CgAwIBAgIJAONqNX/09CYAMA0GCSqGSIb3DQEBAQUAMF1xIzA1BgNV
BAYTA1VTMSMwIQYDVQDEExpZYVNUIERlZmF1bHQgQ0EgKGRsb3BsZGFwKTEeMBWw
CSqGSIb3DQEJARYPCG9zdG1hc3R1ckBzaXRlMB4XDTA3MDgwMzEyNDA1NloXDTE3
MDczMTEyNDA1NlowUjElMAkGA1UEBhMCVVMxIzAhBgNVBAMTG1lhU1QgRGVmYXVs
dCBDQSAoZGxvcGxkYXApMR4wHAYJKoZIhvcNAQkBFg9wb3N0bWFzdGVyQHNpdGUw
ggEiMA0GCSqGSIb3DQEBAQUAA4IBDwAwggEKAoIBAQCshFlzTizSuhvcWfBwHjhy
X9PgH3C3fFXLW7n0jgtmGmQ5PqUBpe8ZtyLVvaTW+F7G836arjGUFHdcNmQgDCDD
JU48UyNCa+koBqrjD8YCbEGnz8IpNnbIrxFD4XDwt01EKPCcChtP0SoifkN31SDQ
k0yJMGIEukbF7Ns25+rF+booxvgVeQ4/KQG5NU1U5Parq+G+sFEGTB6q2kpjpCMB
M80F730u2H2r1uBdAgUB+wAUTp+P+KqNKXI0mZuKI9cid167tz5u3IybWwZ1Nj8I
kH8TDRT+Cne8XSfwJtND2jm2JnK9vHfVxgdxkCpD53bVvcSScf6Id0x5/KePSCvF
AgMBAAGjggE/MIIBOzAPBgNVHRMBAf8EBTADAQH/MCwGCWCGSAGG+EIBDQqFfH1Z
YVNUIEDlbnVYXRlZCB0QSB0ZDZ0ZDZ0ZDZ0ZDZ0ZDZ0ZDZ0ZDZ0ZDZ0ZDZ0ZDZ0
CwYDVROpBAQDAgEGMB0GA1UdDgQWBBrhFZHyzvfwijR3WPxUPEprvnBdrjCBggYD
VR0jBHsweYAUyRWR8s738Io0d1j8VDxKa75wXa6hVqRUMF1xIzA1BgNVBAYTA1VT
MSMwIQYDVQDEExpZYVNUIERlZmF1bHQgQ0EgKGRsb3BsZGFwKTEeMBWwGCSqGSIb3
DQEJARYPCG9zdG1hc3R1ckBzaXRlBgkA42o1f/T0JgAwGgYDVROpBBMwEYEPcG9z
dG1hc3R1ckBzaXRlMB0GA1UdEgQTMGBD3Bvc3RtYXN0ZXJAc2l0ZTANBgkqhkiG
9w0BAQUFAAOCAQEARSQFVo0CQZk80bRF39Ikj6FfV8Ex1vYaDsebYsmEeoK/lP0g
jII091fw0S/80Sd096B76txHMAZLA0BJKD1wsZH4PMYxVbw/J5S8CeQmspUT1yVo
qJyqQB45/JNpyagIdMN/gI0MT/E0v+AVwaPXdh2yNqRGVo6w3/jd4Pj7H8mp43U2
pARRyeJCNeOSMI5DLOyQpmU2REDNcrfRdhjVHBLIqavENZVqGHJ2xWobjJ61lr+b
NF+Vw+HrPnEuRFimqCuRs9yRb0xNX90vJhUQdp1PctvB48WmeBNcUeE+7Ymy2H9J
DEKomkn6L5hzQCA6RD3FR6qROBOuNNSWwWwEyGQ==
-----END CERTIFICATE-----
---
Server certificate
subject=/C=US/CN=dlopldap.site/emailAddress=postmaster@site
issuer=/C=US/CN=YaST Default CA (dlopldap)/emailAddress=postmaster@site
---
No client certificate CA names sent
---
SSL handshake has read 2406 bytes and written 468 bytes
---
New, TLSv1/SSLv3, Cipher is AES256-SHA
Server public key is 2048 bit
Compression: NONE
Expansion: NONE
SSL-Session:
    Protocol    : TLSv1
    Cipher      : AES256-SHA
    Session-ID: 60DA3D90FD3D716C2C47BC71D3B08A3932288ABA01B02BCEF9D08F06E0035A38

```

```

Session-ID-ctx:
Master-Key: 33F4F8CF6112475A88501239FB4D4BA80D53E5B89848482AA81A58894FAB1C99
05137F3AD15E94EFD276CD84B7C7EF38
Key-Arg      : None
Start Time: 1210860425
Timeout      : 300 (sec)
Verify return code: 19 (self signed certificate in certificate chain)
---
DONE

```

The example includes two certificates and surrounding text. The certificates offered by the directory server occur between BEGIN CERTIFICATE and END CERTIFICATE statements.

Copy the certificates into their own files, and be sure to include the BEGIN CERTIFICATE and END CERTIFICATE statements. The name of the file that contains the certificate is not significant, but the file name should clarify which certificate is contained in the file.

Immediately following each END CERTIFICATE statement are two lines describing the issuer and the subject.

- The first certificate in the chain has the subject:

```
/C=US/CN=dlopldap.site/emailAddress=postmaster@site
```

The CN attribute contains the name dlopldap.site. You can store the certificate in a file with a name similar to: site/ssl/cacerts/dlopldap.pem

Note that the extension .pem indicates the certificate format. LDAP supports only PEM-formatted certificates.

- The second certificate in the chain has the subject:

```
/C=US/CN=YaST Default CA (dlopldap)/emailAddress=postmaster@site
```

You can store the certificate in a file with a name similar to: site/ssl/cacerts/YaST-CA.pem

## Creating the CA Certificate Symlinks

You can create symlinks to help OpenSSL locate certificates faster. If a directory server provides a certificate, OpenSSL hashes the subject and serial number of the certificate, and then opens symlinks having the same hash code until it finds the certificate. The symlinks provide an indexing criteria that eliminates the need for OpenSSL to sift through every available certificate.

The previous [Example: Certificate Chain](#) example contains two certificates. You must store each of these certificates in a file before beginning the hashing process that creates the symlinks. OpenSSL cannot use these files directly. Instead, it uses a file with a filename that is the hash code (rendered in readable hexadecimal) and an extension that is a decimal number starting with 0 to be created, in the form:

*hashcode.number*

This scheme allows certificates that have identical hashes to be represented as separate symlinks.

## Creating Symlinks Using the Certlink Utility

The certlink utility provides an automated method of creating symlinks based on certificate hashes. TDGSS software includes the certlink utility in:

- On Teradata Vantage: /opt/teradata/tdgss/bin/certlink
- On Unity: See *Teradata® Unity™ Installation, Configuration, and Upgrade Guide for Customers*, B035-2523 for the recommended location on Unity servers.

To create symlinks based on certificate hashes using the certlink utility:

1. From the site/ssl/cacerts directory that contains the two pem files, enter the following command to hash the files:

```
certlink .
```

2. The command produces output similar to:

```
dlopldap:~ # certlink .
dlopldap.pem => c5fc7afc.0
YaST-CA.pem => 9a135280.0
dlopldap:~ #
```

The example output shows that certlink created two symlink files:

- c5fc7afc.0
- 9a135280.0

These symlinks point to the .pem files:

- dlopldap.pem
- YaST-CA.pem

See [Example: Certificate Chain](#) for an example.

## Testing the Connection

Use tdgssauth to establish and test a connection between Vantage or the Unity server and the directory. For information on using tdgssauth, see [Working with tdgssauth](#).

- If the tdgssauth test succeeds, go to [Configuring TDGSS to Use Advanced Binding Options](#) and configure the TdgssUserConfigFile.xml on Teradata Vantage or the TdgssUnityConfig.xml on Unity. Then restart Vantage or Unity to enable the configuration change.
- If tdgssauth fails and does not provide a clear indication of the failure, you can add the -V 2 option to request that tdgssauth provide additional hexadecimal dumps, which contain hints about the cause



of failure. The [Troubleshooting TLS Setup with a Directory Server](#) topic describes the most common errors and solutions for them.

---

**Note:**

Certificates usually have expiration dates. If a certificate in the chain expires, the connection to the directory ceases to function until you install a new set of certificates in the CA certificate directory.

---

## Configuring TDGSS to Use Advanced Binding Options

To use advanced binding options that require certificates, you must configure the LDAP mechanism properties that request and locate the certificates.

### Prerequisites

Enable basic protection before configuring LDAP certificate properties. See [Using Basic TLS Protection](#).

### Example Configuration

Make the following changes to the `TdgssUserConfigFile.xml` file in the TDGSS site directory on database nodes or in Unity's copy of the file on Unity servers (for Unity configuration, see *Teradata® Unity™ Installation, Configuration, and Upgrade Guide for Customers*, B035-2523).

See [Changing the TDGSS Configuration](#).

- Add the `LdapClientTlsCACertDir` property, and specify the full path to the `site/ssl/cacerts` directory for the property value. This property points to the absolute path of the directory where the two PEM files and the two symlinks are located.

---

**Note:**

If all the CA certs are contained in a single file, you can alternately use the `LdapClientTlsCACert` property to specify the file name.

---

- Add the `LdapClientTlsReqCert` property and set the property value to “demand”. This value causes Teradata Vantage or the Unity server to ask the directory server for a certificate each time a directory user logs on to the database. If the directory does not provide a certificate, or it provides an invalid certificate, TDGSS terminates the connection.

For configuration information, see [LDAP Protection Properties](#).

The following example shows an LDAP mechanism `TdgssUserConfigFile.xml` that includes configured certificate properties. This example also applies to KRB5 or SPNEGO if `AuthorizationSupported` is set to “yes”.

```
<Mechanism Name="ldap">
  <MechanismProperties
    ...
    LdapServerName="ldap://someserver/"
    LdapClientUseTls="yes"
    LdapClientTlsCACertDir="/opt/teradata/tdat/tdgss/site/ssl/cacerts/"
    LdapClientTlsReqCert="demand"
  />
</Mechanism>
```

**Note:**

For configuration requirements when authentication is set for multiple directory services, see [Creating the <LdapConfig> Section in the TdgssUserConfigFile.xml](#).

## Using Mutual Authentication Between the Directory Server and Teradata Vantage

### About Mutual Authentication

Once you configure Teradata Vantage or the Unity server to authenticate the directory server, you can also setup the database to be authenticated by the directory server.

Site security policy may require the directory server to authenticate the Teradata Vantage nodes. Site policy determines how and when a network service (such as the database) must provide client certificates. You can configure Vantage system components to conform to these policies.

### Working with Certificates and Private Keys

When security policy requires mutual authentication of the database, the Unity server, and the directory, you must install certificates and associated keys on each Teradata Vantage node and on the Unity server. These certificates are known as client certificates because for LDAP purposes, the database and Unity server are clients of the directory server. Client certificates must be in PEM format and conform to the requirements of the directory server being used. Coordinate with the directory administrator or the directory vendor to determine detailed certificate requirements.

Store the certificates and keys in separate files. Protect the file containing the private key very carefully. Anyone with this key can assume the identity of the database

**Note:**

You can store the certificates and keys in the same file, but it is not recommended.

## Mutual Authentication Implementation Process

1. Install the Private Key. See [Installing the Private Key](#).
2. Install the certificate. See [Installing the Certificate](#).
3. Ensure that the key and certificate pair are installed on each database node. See [Installing Keys and Certificates for the Entire Teradata Vantage System](#).
4. Make the necessary updates to the TdgssUserConfigFile.xml. See [Updating the TDGSS Configuration](#).

## Installing the Private Key

Execute the following procedure on each Teradata Vantage node and on the Unity server, if used.

1. Obtain a certificate and key in pem format, according to your site security policy.
2. Create a directory called site/ssl/cacerts in the:
  - TDGSS site directory on database nodes.
3. From within the new directory you created in step 1, create an empty file named clientkey.pem, using the following commands:

Enter: touch clientkey.pem

Then enter: chmod 0600 clientkey.pem

4. Place the key in this file using an editor or the Posix cat command.
5. Secure the clientkey.pem file to be read-write for the file owner.

The resulting clientkey.pem file looks similar to:

```
-----BEGIN RSA PRIVATE KEY-----
MIICXAIBAAKBgQDRBPoI/fdAoezFRaqN63IdYW1Laucs+akMr+0qP47kKu/SkyUh
d6u1EfLryzBIEubTD//GJXLDBCS4DbCx7xDzSVCUFQmR+X16241kKsyq06wvd+9J
pCwk6ktskV0gK437HG4YKo4q4BHIj13VSRZxDqv2GU8IYfFesqWAdFhEhwIDAQAB
AoGAiw0Am01tvwroV5R9K1tmQYMK/vCoX6RmMth1nvYVkjGZEejW+yvEQZMG93+V
UyDIUHCIZcP14LobJjo1fUEnyDag37P7FE9JDXr7I3QRNA0keR+w0egNpMcQMIDE
Bgj7UCycCxuz0FX1UuvcnCMJH7QfBLb3p01BgK6W2ENfxLECCQD5PMSfs+ogS7Bb
fch1thBJA3576PyBeBURrcz/03lmUTKz0vAzXBPWTxNCV/tLn1HUvEYuiZ2pyun3
0zjcr2UFAkEA1rDZpCMZ4woUkvYX+BwkffG8HXnZNGROd4zu1tbQEgeBjOSVx299
s/FSxEgtMRSGv6vPwDMCQFyy+teDJ7Im8isJTDNbF19HTv+qzYdRDMRPUEZqPB4W
7FMz/PlpoOmeGj1gTID5Hfjw7kPvHfi5GwJBA083aik2j8LLostNmqsV4e+SUPYx
GxpQ3TgIrrdSqCSSTq3WCgHhoJCTeRK2S1W75tje1CXao97yCTp6GxuFpNkCQDLv
wKNlxJW0ZbU8eBFgs/PBr80ahMMebV0F94C3dKRibYU9EqA/vpOcZgBG0J557w3w
66sz2d5P4q71EBDcWE05DsFE9fqwAR5xcowqGPYiuh0=
-----END RSA PRIVATE KEY-----
```

6. OpenSSL does not accept a key in a globally readable file. Use the following commands to prevent unauthorized persons from obtaining the key.
  - a. Enter `chmod 0400 clientkey.pem`
  - b. Enter `chown gtw-or-unity-user clientkey.pem`
  - c. Substitute the user name of the Vantage gateway or Unity user for `gtw-or-unity-user`.

## Installing the Certificate

Create a file named `clientcert.pem` in the `site/ssl/cacerts` directory. Make it writable by the owner and universally readable. Place the certificate in the file.

### Sample of a Certificate File Installed on a Database Node

```
-----BEGIN CERTIFICATE-----
MIIC5DCCAk2gAwIBAgIJAMvJ4ZlAGSiNMA0GCSqGSIb3DQEBBQUAMFYxCzAJBgNV
BAYTA1VTMSEwHwYDVQQDEExhkbG9wbGRhcC50ZC50ZXJhZGF0YS5jb20xJDAiBgkq
hkiG9w0BCQEFWRsMTYwMDEwQHRlcmFkYXRhLmNvbTAeFw0wODA1MTQxOTA4NDJa
Fw0wOTA1MTQxOTA4NDJaMFYxCzAJBgNVBAYTA1VTMSEwHwYDVQQDEExhkbG9wbGRh
cC50ZC50ZXJhZGF0YS5jb20xJDAiBgkqhkiG9w0BCQEFWRsMTYwMDEwQHRlcmFk
YXRhLmNvbTCBnzANBgkqhkiG9w0BAQEFAAOBjQAwgYkCgYEA0QT6CP33QKHsxUWq
jetyHWFtS2rnLPmpDK/tKj+O5Crv0pMlIXertRHy68swSBLm0w//xiVywWkuA2w
se8Q80lQlBUJkf19etuNZCrMqjusL3fvSaQlp0pLZLFdICuN+xxuGCq0KuARyI5d
1UkwcQ6r9hlPCGHxXrKlghRYRiCawEAAa0BuTCBtjAdBgNVHQ4EFgQUamJoMI9/
TTS59BUTF1EwoEseNAwwgYYGA1UdIwR/MH2AFGpiaDCPf000ufQVExdRFqBLHjQM
oVqkWBWMQswCQYDVQQGEwJVUzEhMB8GA1UEAxMYZGxvcGxkYXAudGQudGVyYWRh
dGEuY29tMSQwIgYJKoZIhvcNAQkBFhVkbDE2MDAxMEB0ZXJhZGF0YS5jb22CCQDL
yeGZWhkojTAMBgNVHRMEBTADAQH/MA0GCSqGSIb3DQEBBQUAA4GBAIUbbcrUG3Y0
VXdhAj1AKq95qryTeHE1wiDBmEe1UIC5KyyarGW9tA/sxaJ+9X/zrAwP1ymLn5n9
kIJt3gh7HjjrG1qzC7jRVoI0Yl/z+7QUKejGp0ph1gVl4VwFoRzxv+I2vIUuzyF3
dabR1Q0+lqgc1CHC001VEHEAK8v9k6q1
-----END CERTIFICATE-----
```

## Installing Keys and Certificates for the Entire Teradata Vantage System

Repeat the key and certificate installation process on each node of the Teradata Vantage system and on the Unity server, if used. Make sure to keep the keys and certificates for each node paired properly, because each pair is uniquely matched. The file must be *named and located identically* on every node of the system even though the contents of each file are different.

## Updating the TDGSS Configuration

After you store a certificate in `site/ssl/certs/clientcert.pem` and a paired key in `site/ssl/certs/clientkey.pem` for each LDAP client (on each database node or on each Unity server), update the TDGSS configuration to include values for the `LdapClientTlsCert` and `LdapClientTlsKey` properties. See [Configuring TDGSS to Use Advanced Binding Options](#).

---

### Note:

The Linux user under which Teradata Vantage runs (user `teradata`) must have read access to the directory specified in the `LdapClientTlsCert` and `LdapClientTlsKey` properties. For sites that configured this property before Release 14.0, the permission is granted automatically by a script upon upgrade to Release 14.0. For sites that configure these properties on Release 14.10 or later, you must grant the permission manually. See [Working with OS-Level Security Options](#).

---

## Sample Configuration for Mutual Authentication

The following example shows a typical `TdgssUserConfigFile.xml` update to support TLS mutual authentication on the LDAP mechanism. Configuration of the KRB5 or SPNEGO mechanism is similar.

```
<Mechanism Name="ldap">
  <MechanismProperties
    ...
    LdapClientTlsCert="/opt/teradata/tdat/tdgss/site/ssl/certs/clientcert.pem"
    LdapClientTlsKey="/opt/teradata/tdat/tdgss/site/ssl/certs/clientkey.pem"
  />
</Mechanism>
```

After you add the client certificate and key to the `TdgssUserConfigFile.xml`, and run the `run_tdgssconfig` utility in the TDGSS bin directory, you can test the setup with `tdgssauth`. See [Working with tdgssauth](#).

---

### Note:

Make sure to verify the configuration on each Vantage node and on the Unity server, if used. Failure to adequately test the configuration can result in loss of connectivity for Vantage clients using LDAP authentication.

---

After you verifying the results, restart Teradata Vantage to enable the new configuration.

## Troubleshooting TLS Setup with a Directory Server

The topics that follow present common errors that may occur when configuring TLS options, along with suggested corrective action.

## Hostname Does Not Match CN in Peer Certificate

This error is caused by a lack of agreement between the CN in the directory server certificate and the network name service when `LdapClientTlsReqCert` is set to a value other than “never”.

### Error Message Text

```
hostname does not match CN in peer certificate
```

### Corrective Action

1. Obtain the certificate from the directory with the `openssl` command:

```
openssl s_client -connect server_name:port
```

#### ***server\_name***

The directory server DNS name.

#### ***port***

The port where SSL listens.

2. In the output from this command, find the line that begins with `subject`. This string should contain a CN attribute. The CN attribute value, a name, must resolve in DNS to the IP address of the directory server. The error message occurs because the name is either unresolved, or resolves to the wrong IP address. The error is related to either a DNS problem or a problem with the name in the server certificate.
3. Check the following items to determine the problem and then fix it.
  - a. If the `LdapServerName` property names the directory server explicitly, make sure the name in the property value matches the name in the subject for the directory server certificate. For example, if the subject CN attribute contains:

```
dlopldap.td.teradata.com
```

then make sure the `LdapServerName` property contains either the TLS specification:

```
ldap://dlopldap.td.teradata.com/
```

or the SSL specification:

```
ldaps://dlopldap.td.teradata.com/
```

- b. Make sure that the name in the CN attribute is resolvable and returns the correct IP address. Fix any errors and try again.
- c. If the name in the CN attribute cannot be resolved or resolves to the wrong IP address, and cannot be changed in DNS, you must install a new certificate on the directory server. See [Checking the Directory Server Certificates](#).

The CN attribute must meet these requirements:

- The subject for the certificate must contain the DNS name (preferably, the fully qualified DNS name) that resolves to the IP address where the server is listening.
- The DNS name must correctly resolve on the Teradata Vantage nodes or Unity server.
- If the LdapServerName attribute is configured to explicitly name directory servers, the value in the subject's CN attribute must be used in the configured LDAP or LDAPS URI.

## SSL: Certificate Verify Failure

This error message usually indicates a problem with the server certificate.

### Error Message Text

```
error:14090086:SSL routines:SSL3_GET_SERVER_CERTIFICATE:certificate
verify failed
```

### Corrective Action

1. Obtain the needed certificate(s). See [Obtain the Needed Certificate](#).

---

#### Note:

The process shown in the cross-referenced topic is for a simple SSL deployment. You may not be able to get all of the certificates you need this way, and may need to contact the site security administrator to obtain the certificates you need to authenticate the directory.

The certificates must be in PEM format. If not, you must convert them to PEM format using the `openssl x509` command before going on to the next step.

---

2. Install the certificate(s). See [Installing the Certificate](#).
3. Run the `c_rehash` utility. See [Creating Symlinks Using the Certlink Utility](#).

**Note:**

This error may indicate a security breach. Consider the following actions to check for a possible security threat.

- Check the access logs to detect any suspicious activity.
- Get a copy of the directory server certificate chain to you to make sure that the database has the officially trusted set of certificates.

---

## **TLS: Self-signed Certificate Offered or Is Part of the Certificate Chain**

This error message may not represent a real security issue. It is an alert that the directory server is offering a self signed certificate. This error usually occurs in the:

- Message traces, when you use the `tdgssauth -V2` option. See [Working with tdgssauth](#).
- Diagnostic output from the ldap tools, when you use the `-d2` option.

A more serious form of this error is: `certificate verify failed`

### **Error Message Text**

```
TLS certificate verification: Error, self signed certificate
```

### **Corrective Action**

If your site allows self-signed certificates, you can eliminate the error message by installing the self-signed certificate. See [Installing the Certificate](#).

## **TLS: Unable to Get Local Issuer Certificate**

This error occurs when you use a certificate signed by a third party (like VeriSign or Thawte) in the directory. In such cases, the directory server may not offer the complete certificate chain, prevents certificate verification.

### **Error Message Text**

```
verify error:num=20:unable to get local issuer certificate
```



## Using openssl to Identify the Certificates Not Verified

When you encounter the verify error:num=20, you can use the openssl command to display the certificate chain. The output shows a chain that ends with an issuer for which there is no certificate, for example:

```
depth=1 /O=VeriSign Trust Network/OU=VeriSign, Inc./OU=VeriSign International
Server CA - Class 3/OU=www.verisign.com/CPS Incorp.by Ref. LIABILITY LTD.
(c)97 VeriSign
verify error:num=20:unable to get local issuer certificate
verify return:0
CONNECTED(00000003)
---
Certificate chain
 0 s:/C=US/ST=California/L=El Segundo/O=Teradata/OU=Domain
Controllers/CN=sussan140.td.teradata.com
   i:/O=VeriSign Trust Network/OU=VeriSign, Inc./OU=VeriSign International
Server CA - Class 3/OU=www.verisign.com/CPS Incorp.by Ref. LIABILITY LTD.
(c)97 VeriSign
-----BEGIN CERTIFICATE-----
...snipped...
-----END CERTIFICATE-----
 1 s:/O=VeriSign Trust Network/OU=VeriSign, Inc./OU=VeriSign International
Server CA - Class 3/OU=www.verisign.com/CPS Incorp.by Ref. LIABILITY LTD.
(c)97 VeriSign
   i:/C=US/O=VeriSign, Inc./OU=Class 3 Public Primary Certification
Authority-----BEGIN CERTIFICATE-----
...snipped...
-----END CERTIFICATE-----
Server certificate
subject=/C=US/ST=California/L=El Segundo/O=Teradata/OU=Domain
Controllers/CN=sussan140.td.teradata.com
issuer=/O=VeriSign Trust Network/OU=VeriSign, Inc./OU=VeriSign International
Server CA - Class 3/OU=www.verisign.com/CPS Incorp.by Ref. LIABILITY LTD.
(c)97 VeriSign
---
Acceptable client certificate CA names /C=US/O=VeriSign, Inc./OU=Class 1
Public Primary Certification Authority - G2/OU
=(c)1998 VeriSign, Inc.-For authorized use only/OU=VeriSign Trust Network
/C=US/O=VeriSign, Inc./OU=Class 4 Public Primary Certification Authority - G2/
OU=(c) 1998 VeriSign, Inc. - For authorized use only/OU=VeriSign Trust Network
/C=US/O=VeriSign, Inc./OU=Class 3 Public Primary Certification Authority
/C=US/O=VeriSign, Inc./OU=Class 2 Public Primary Certification Authority
```

```

/C=US/O=VeriSign, Inc./OU=Class 1 Public Primary Certification Authority
/C=US/O=VeriSign, Inc./OU=Class 3 Public Primary Certification Authority - G2/
OU=(c) 1998 VeriSign, Inc. - For authorized use only/OU=VeriSign Trust Network
/OU=Copyright (c) 1997 Microsoft Corp./OU=Microsoft Corporation/CN=Microsoft
Root Authority/DC=com/DC=microsoft/CN=Microsoft Root Certificate Authority---
SSL handshake has read 5299 bytes and written 312 bytes
---
New, TLSv1/SSLv3, Cipher is RC4-MD5
Server public key is 1024 bit
Compression: NONE
Expansion: NONE

```

The error occurs at a depth of 1, that is, one certificate down the certificate chain, openssl cannot verify the certificate. This error indicates that openssl could not find the issuer certificate or an acceptable client certificate.

## Corrective Action

Use the following procedure to provide the missing local issuer certificate.

1. Obtain the needed certificate.
2. Install the certificate. See [Installing the Certificate](#).
3. Run the `c_rehash` utility. See [Creating Symlinks Using the Certlink Utility](#).

## Obtain the Needed Certificate

Typically, a certificate can be acquired from the site security administrator. For systems running Linux, with openssl installed, some or all of these certificates can be found in the `/etc/site/ssl/certs` directory. Do the following to obtain a certificate.

1. Go to the directory where certificates are stored, for example: `/etc/site/ssl/certs`.
2. List the files. The files look similar to:

```

dlopldap:~ # cd /etc/ssl/certs
dlopldap:/etc/ssl/certs # ls
1e49180d.0  7a9820c1.0      a3c60019.0  demo          thawteCb.pem
2edf7016.0  843b6c51.0      aad3d04d.0  eng1.pem      thawteCp.pem
56e607f4.0  878cf4c6.0      argena.pem   eng2.pem      vsign1.pem
594f1775.0  Equifax-root1.pem argeng.pem   eng3.pem      vsign3.pem
6adf0799.0  ICP-Brasil.pem  c33a80d4.0  eng4.pem      vsignss.pem
6f5d9899.0  RegTP-5R.pem    cdd7aee7.0  eng5.pem      webgo.pem
714aceac.0  RegTP-6R.pem    d4e39186.0  expired
7651b327.0  YaST-CA.pem     ddc328ff.0  f73e89fd.0
dlopldap:/etc/ssl/certs # openssl x509 -inform pem -in vsign3.pem -subject

```

```

subject= /C=US/O=VeriSign, Inc./OU=Class 3 Public Primary Certification
Authority-----BEGIN CERTIFICATE-----
MIICPDCCAaUCEHC65B0Q2Sk0tjjKewPMur8wDQYJKoZIhvcNAQECBQAwXzELMAkG
A1UEBhMCVVMxZAVBgNVBAoTDlZlcm1TaWduLCBjbMuMTcwNQYDVQQLZy5DbGFz
cyAzIFB1YmxpYyBQcm1tYXJ5IENlcnRpZmljYXRpb24gQXV0aG9yaXR5MB4XDk2
MDEyOTAwMDAwMFoXDTI4MDgwMTIzNTk1OVowXzELMAkGA1UEBhMCVVMxZAVBgNV
BAoTDlZlcm1TaWduLCBjbMuMTcwNQYDVQQLZy5DbGFzcyAzIFB1YmxpYyBQcm1t
YXJ5IENlcnRpZmljYXRpb24gQXV0aG9yaXR5MIGfMA0GCSqGSIb3DQEBAQUAA4GN
ADCBiQKBgQDJXfme8huKARS0EN8EQNvjV69qRUCPhAwL0TPZ2RHP7gJYHyX3KqhE
BarsAx94f56TuZoAqiN91qyFomNFx3InzPRMxnVx0jnvT0Lwdd8KkMa0IG+YD/is
I19wKTakyYbnsZogy10lhec9vn2a/iRFM9x2Fe0PonFkTGUgWhFpwIDAQABMA0G
CSqGSIb3DQEBAQUAA4GBALtMEivPLCYATxQT3ab7/AoRhIzzKBxnki98tsX63/Do
lbwdj2wsqFHMci9kwFPwTtYmwHYBV4GSXiHx0bH/59AhWM1pF+NEHJwZRDmJXNyc
AA9WjQKZ7aKQRUzkuXcKpFAyAw7xzvjyVGM5mKf5p/AfbdynMk20mufTqj/ZA1k
-----END CERTIFICATE-----
dlopldap:/etc/ssl/certs #

```

**Note:**

The files that end in a non-numeric suffix are certificate files. On this system, all certificates are stored in PEM format.

3. Use the `openssl x509` command, with a `-subject` option, to examine the subject of each certificate. Run the `openssl x509 -subject common` to each of the `.pem` files until it finds a file containing a certificate with a subject that matches the missing certificate.

The subject shown in bold text in the example (`subject= /C=US/O=VeriSign, Inc./OU=Class 3 Public Primary Certification Authority`) matches the subject of the missing certificate identified in the search shown in [Using openssl to Identify the Certificates Not Verified](#).

## Using TLS with Client to Database Connections

The TLSv1.2 protocol provides confidentiality and data integrity for network traffic transmitted between clients and the SQL Engine Gateway.

- HTTPS port 443 is the default port for TLS connections.
- Port 1025 is the default port for legacy non-TLS connections.
- TTU 17.10 client drivers and interfaces connect to SQL Engine 17.00, 16.20, 16.10 and 15.10 without using TLS.
- TTU Release 17.00 and older client drivers and interfaces connect to SQL Engine 17.10 without using TLS.
- If the Gateway is configured to require TLS, TTU Release 17.00 and older client drivers and interfaces cannot connect to SQLE 17.10.

- TLS is supported by network-connected TTU drivers and interface products, including: JDBC, ODBC, .NET, and CLIV2.
- Certificates must be in .PEM format.

TLS can be configured to require clients to use only TLS, but the default configuration allows clients without TLS to connect to the database.

### Prerequisites for TLSv1.2 Configuration

- SQL Engine 17.10 or later.
- TTU 17.10 or later.
- Port 443 must be open on the firewall.
- Certificate management is essential for TLSv1.2 enablement. How the certificates are managed is the responsibility of the customer according to their security policies and security requirements.

### TLSv1.2 Considerations

- TLS not available for the mainframe channel.
- Performance varies by TLS cipher choice and workload.

### Client Configuration

The default data transfer encryption setting for CLIV2 is SSLMODE=ALLOW which means prefer legacy port (and TDGSS) but optionally use TLS.

The default data transfer encryption setting for the drivers (JDBC, ODBC, and .NET) is SSLMODE=PREFER which means prefer TLS port but optionally use legacy port.

For detailed information about client TLS configuration, see the appropriate manual for your client.

## Configuring Signed Certificates and Private Keys

TLS requires each database and Unity server to have a signed certificate and private key.

Generating the signed certificates requires the customer's private key which is not available on the database servers.

Two command line tools are provided for configuring and installing certificates and keys:

- The `tlsutil` tool generates the Certificate Signing Requests (CSRs) and installs the certificates and private keys. See [About tlsutil](#).
- The `nodenames` tool is invoked by `tlsutil` to obtain a list of the names that a node is known by for inclusion in the CSR and signed certificate. `nodenames` is also useful for troubleshooting some DNS issues. See [About nodenames](#).

The process is to create CSRs and private keys for a single node or for each node in the database and collect the CSR files at the node that is running `tlsutil`. By default, the CSR files are placed in the directory `/opt/teradata/tadat/tgtw/site/tls/tmpdir/newcsrs`.

Each private key is saved in a temporary location and is not installed at this time. The private key never leaves the database server it was created for.

## Prerequisites

- DNS must be configured for `tlsutil` and `nodenames` to work.

## Generate and Install Signed Certificates and Private Keys

Perform the following steps to generate and install signed certificates and private keys.

### Note:

Signed certificates cannot be created on the Teradata nodes at a customer site.

1. Log in as root on an SQL Engine node. You may perform this process from any node. The node used for creating the CSRs must be the same node used to install the signed certificates.
2. Create CSR(s) and private key(s). You may create a unique CSR for each node or a single CSR that is used on all nodes in a system. Perform one of the following:

- Create a unique CSR for every node using the default 2048 bit RSA key:

```
# tlsutil -c mydb.example.com
```

- Create a unique CSR for every node using an elliptic curve key:

```
# tlsutil -c -k ec:secp384r1 mydb.example.com
```

- Create a single CSR for all nodes in the system:

```
tlsutil -c -s mydb.example.com
```

3. Move the CSR(s) to a customer system and generate a signed certificate for each CSR.

This step is not performed on the SQL Engine nodes. The customer site administrator uses the CSR(s) to create signed certificates using a customer-defined procedure.

### Note:

Teradata recommends that you do not use self-signed certificates.

4. Place the signed certificates on the same SQL Engine node that `tlsutil` was run from earlier.

### Note:

The signed certificates must be in .PEM format.

The signed certificates can be:

- Placed in `/opt/teradata/tdat/tgtw/site/tls/tmpdir/signedcerts`. The file names of the signed certificates are arbitrary.

- Zipped into a single ZIP file and placed in /opt/teradata/tdat/tgtw/site/tls/tmpdir/zipfiles. This requires that `tlsutil -c` was run with the `-z` option.
  - Zipped into a single ZIP file and placed in a directory of your choice
5. Install the signed certificates and private keys on each node. Installation is done with the `-i` option.

**Note:**

This must be run on the same database server from which `tlsutil` was run with the `-c` option.

Perform one of the following steps:

Option	Command
Install the signed certificates and private keys.	<code>tlsutil -i</code>
If <code>tlsutil</code> was initially run with the <code>-d</code> option to specify a different directory, the same directory must be specified in the install command.	<code>tlsutil -i -d <i>directory</i></code>
Use the <code>-z</code> option if you want <code>tlsutil</code> to get the signed certificates from a zipped archive at location /opt/teradata/tdat/tgtw/site/tls/tmpdir/zipfiles/all_certs.tgz. To use your own location for the zipped archive file, add the <code>-f</code> option along with the full path to the file.	<code>tlsutil -i -z</code>  OR <code>tlsutil -i -z -f <i>zip_file_path</i></code>

6. Remove the temporary files created from previous steps on each node. Perform one of the following steps:

Option	Command
Remove the temporary files from the default temporary directory on all the nodes. The default temporary directory is here: /opt/teradata/tdat/tgtw/site/tls/tmpdir.	<code>tlsutil -r</code>
Clean up the temporary directory on the local node only.	<code>tlsutil -r -l</code>
If you specified your own directory use the <code>-d</code> option to clean up.	<code>tlsutil -r -d <i>directory</i></code>

7. Optional. Test that the certificates are valid:

```
tlsutil -t
```

## Example: Create Signed Certificates and Private Keys on All Nodes Using ZIP Archives

1. Generate CSRs:

```
# tlsutil -c -z mydb.example.com
```

Your result will be similar to the following:

```
CSRs have been generated for all nodes (4 generated):
/opt/teradata/tdat/tgtw/site/tls/tmpdir/newcsrs/gtwcsr.mydb1.csr
/opt/teradata/tdat/tgtw/site/tls/tmpdir/newcsrs/gtwcsr.mydb2.csr
/opt/teradata/tdat/tgtw/site/tls/tmpdir/newcsrs/gtwcsr.mydb3.csr
/opt/teradata/tdat/tgtw/site/tls/tmpdir/newcsrs/gtwcsr.mydb4.csr
CSRs have been archived in:
/opt/teradata/tdat/tgtw/site/tls/tmpdir/zipfiles/all_csrs.tgz
```

2. The customer signs the certificates using a customer-defined process.
3. Obtain the signed certificates. Run "tar" to archive them and place the archive here: /opt/teradata/tdat/tgtw/site/tls/tmpdir/zipfiles/all\_certs.tgz
4. Install the signed certificates and private keys:

```
# tlsutil -i -z
```

Result:

```
Signed certificate and private key installed on 4 node(s).
```

5. Remove the temporary files created from previous steps on each node.

```
# tlsutil -r
```

6. Optional. Test that the certificates are valid:

```
# tlsutil -t
```

## Example: Update Only Invalid Signed Certificates

Use the `tlsutil -u` option to create signed certificates on a subset of database servers. This option is used with the `-c` option only.

When used with the `-c` option, update mode checks the signed certificates and private keys on all database servers and creates CSRs only for those that do not have a valid certificate and key.

Update mode used with -c reports that all certificates are valid if none fail the validity test. In that case, no further action is required.

For example, as root, run the following commands to update invalid signed certificates:

1. Generate CSRs:

```
# tlsutil -c -u mydb.example.com
```

Result: If all certificates are valid, no further action is required.

2. If some certificates are invalid, sign the certificates using a customer-defined process.
3. Install the signed certificates and private keys:

```
# tlsutil -i
```

## Signed Certificates and Private Keys Default Locations

TLS requires each SQL Engine node to have a signed certificate and private key.

Description	Default Location
Signed certificate on each node	/opt/teradata/tdat/tgtw/site/tls/certs/gtwcert.pem
Private key on each node	/opt/teradata/tdat/tgtw/site/tls/private/gtwkey.pem
Directory that holds the new CSRs	/opt/teradata/tdat/tgtw/site/tls/tmpdir/newcsrs
Directory where customer places the signed certificates	/opt/teradata/tdat/tgtw/site/tls/tmpdir/signedcerts
Zipped archive files	/opt/teradata/tdat/tgtw/site/tls/tmpdir/zipfiles

## About tlsutil

The tlsutil utility is used to obtain and install signed certificates and private keys for use with TLS.

### tlsutil Syntax

```
tlsutil -c [-s | -l | -u [-e expire_time]] [-d directory] [-v]
          [-k rsa[:keylength] | ec[:named_curve]]
          [-g "genpkey_parameters"]
          [-z] database_name ...
```

```
tlsutil -i [-d directory] [-v] [-z [-f filename]]
```

```
tlsutil -r [-l] [-d directory] [-v]
```



```
tlsutil -t [-l] [-d directory] [-v] [-e expire_time]
```

```
tlsutil -h
```

## tlsutil Syntax Elements

The following table contains descriptions of the `tlsutil` command arguments.

Command Arguments	Description
-c	Create one or more Certificate Signing Requests(CSR's).
-d	Directory to hold certificates, keys and temporary storage. The directory must start with "/".
-e	Validity threshold until certificate expiration in days.
-f	File (in ZIP format) containing all signed certificates.
-g	The -g option allows a quoted string of parameters to be passed to openssl genpkey to generate private keys using genpkey. Do not include "openssl genpkey" or the "-out" parameter.
-h	Displays the help text and lists the valid values for named curves.
-i	Installs all signed certificates and private keys.
-k	The -k option provides parameters for rsa and ec private key generation. For example: <ul style="list-style-type: none"> <li>• <b>rsa key:</b> Optionally specify keylength. Default is 2048.</li> <li>• <b>ec key:</b> Optionally specify named curve. Default is secp384r1.</li> </ul>
-l	Local node only. Note, the default is to perform operations on all nodes.
-r	Remove temporary directories and other subdirectories from default locations. If the -d option is used, -r will remove <directory>/tmpdir and all subdirectories
-s	The same private key and signed certificate are installed on all nodes. The -s option is used with <code>tlsutil -c</code> (create CSR mode). This option creates a single CSR which can be used on any node in the system. When the -s option is used, instead of using the output of <code>nodenames</code> (which may include node-specific names), only the list of database names intended to be passed to <code>nodenames</code> is used. A single CSR is created. The user is responsible for using the CSR to generate a signed certificate. When <code>tlsutil -i</code> is run to install the signed certificate, the single signed certificate is installed on all nodes, along with the same private key.
-t	Test mode. Used to confirm that signed certificates are valid.
-u	Update mode. Only create CSRs for nodes where the installed private key or certificate is missing, invalid, or the certificate is at or near expiration.
-v	Verbose mode.

Command Arguments	Description
-z	Zipped file used to hold all CSRs and signed certificates. -z has no effect when running in local mode.

**directory**

The name of the directory to hold certificates, keys, and temporary storage. The directory must start with "/".

**database\_name**

Name of the database. Teradata recommends using the fully qualified name of the database. For example: xyz.example.com.

**expire\_time**

Number of days until a certificate expires.

**filename**

Name of the ZIP file that contains all of the signed certificates.

**genpkey\_parameters**

genpkey is an OpenSSL command that generates a private key.

There are several parameters for genpkey. For details on genpkey parameters, see the web. The "openssl genpkey" and "-out *key\_file\_name*" arguments are not allowed in the -g option, because tlsutil supplies those.

**named\_curve**

The name of the elliptical curve encryption cipher you want to use.

`tlsutil -h` lists the valid named curves.

## tlsutil Usage

The default is for all files created by tlsutil to be placed in or under the `/opt/teradata/tdat/tgtw/site/tls` directory.

For Unity servers, specify the -d option to use a different directory. Once an alternate directory is selected, the -d option with the same directory must be used for all subsequent commands. It is strongly recommended that the directory chosen not be used for any other purpose. tlsutil does not allow the root directory ("/") to be specified.

You must be logged in as root to run tlsutil.

## tlsutil Examples

Run the following examples from the command line.

tlsutil is located in /opt/teradata/tdgss/bin.

---

### Note:

You must be logged in as root to run tlsutil.

---

### Example: Create a CSR for every node

```
# tlsutil -c mydb.example.com
```

### Example: Install signed certificates on nodes

```
# tlsutil -i
```

### Example: Create a single CSR for use on every node

```
# tlsutil -c -s mydb.example.com
```

### Example: Create a CSR for the local node only

```
# tlsutil -c -l mydb.example.com
```

### Example: Create CSRs only for nodes where the certificates are invalid or will expire within 30 days

```
# tlsutil -c -u -e 30 mydb.example.com
```

### Example: Create a CSR for every node using a 4096 bit RSA key

```
# tlsutil -c -k rsa:4096 mydb.example.com
```

### Example: Create a CSR for every node using an elliptic curve key

```
# tlsutil -c -k ec:secp384r1 mydb.example.com
```

## About nodenames

The nodenames utility displays a list of network interfaces on the node where it is run. It is used internally by the tlsutil utility and can also be run from the command line.

The information provided by `nodenames` is helpful when you generate a Certificate Signing Request (CSR) because it provides the common name (CN) and subject alternative names (SANs) that are used in the CSR.

Use the first name produced by `nodenames` as the CN attribute. Include all the names produced by `nodenames` in the SANs, including the CN name.

## **nodenames Syntax**

```
nodenames [-v] database_name ...
```

## **Syntax Elements**

**-v**

Optional. Displays verbose output.

**database\_name**

One or more names of the database. Teradata recommends using the fully qualified name of the database. For example: `mydb.example.com`.

---

**Note:**

The fully qualified database name must be in DNS for `nodenames` to recognize it.

---

## **nodenames Example**

From the command line, run:

```
#/opt/teradata/tdgss/bin/nodenames mydb.example.com
```

Result:

```
mydb1.example.com
mydbcop1.example.com
```

## **Cipher Suites and Overriding the Configuration File**

The gateway TLS configuration file contains the configured cipher suites from which the gateway parses and loads the secure cipher suites. The configuration file is in standard OpenSSL format.

The configuration file is located here: `/usr/tgtw/etc/gtwtls.cfg`.

To override the settings to add or remove ciphers, copy and paste the configuration file into a local file called `/usr/tgtw/etc/localgtwtls.cfg` and make your edits there.

The default cipher suite list contains the following ciphers:

- TLS\_AES\_256\_GCM\_SHA384
- TLS\_CHACHA20\_POLY1305\_SHA256
- TLS\_AES\_128\_GCM\_SHA256
- ECDHE-ECDSA-AES256-GCM-SHA384
- ECDHE-RSA-AES256-GCM-SHA384
- DHE-RSA-AES256-GCM-SHA384
- ECDHE-ECDSA-CHACHA20-POLY1305
- ECDHE-RSA-CHACHA20-POLY1305
- DHE-RSA-CHACHA20-POLY1305
- ECDHE-ECDSA-AES128-GCM-SHA256
- ECDHE-RSA-AES128-GCM-SHA256
- DHE-RSA-AES128-GCM-SHA256
- AES256-GCM-SHA384
- AES128-GCM-SHA256

## Updating Cipher Suites and Versions

If you updated the version or cipher suites, use gtwglocal "Reread TLS Config" to update the configuration from the gateway's TLS configuration file. This requires a "SEt HOSt" command before rereading the configuration file.

To update, run the following commands:

```
tdatcmd
gtwglobal
set host host_number
> reread TLS config
```

## Updating TLS Certificate and Private Key

If you installed a new certificate and key pair, use gtwglocal "Reread TLS Certificate" to update the TLS certificate and private key. This requires a "SEt HOSt" command before it can be used.

To update, run the following commands:

```
tdatcmd
gtwglobal
set host host_number
> reread TLS certificate
```

## Using gtwcontrol to Enable or Disable TLS and to Set the Port

TLS can be disabled and later re-enabled by using the gtwcontrol TLS option.

### Note:

By default TLS is enabled, but Gateway cannot accept TLS connections until the TLS certificates are deployed.

After your system is configured you can use gtwcontrol to change the settings on how you want Gateway to listen on the ports; for example, to listen on both TLS and HTTPS ports or to listen on one port only.

```
gtwcontrol --TLS [disable|enable|require|nolegacy][,trace=yes|no|all]
```

1. To disable TLS:

```
gtwcontrol --TLS disable
```

2. Restart the database for the change to take effect:

```
tpareset -f "Disabling TLS"
```

### Listen on both HTTPS port and Legacy port. Accepts connection on both ports.

```
# gtwcontrol --TLS enable,trace=no
```

### Listen on both HTTPS port and Legacy port and only accepts connections from HTTPS port

```
// Listen on both HTTPS port and Legacy port and only accept
// connections from HTTPS port.
// Return an error for connections from the Legacy port.
# gtwcontrol --TLS require,trace=yes
```

### Listen on Legacy port only

```
# gtwcontrol --TLS disable,trace=yes
```

### Listen on HTTPS port only

```
# gtwcontrol --TLS nolegacy,trace=all
```

**Note:**

The database has to be restarted for the changes to take effect.

For more information, see *Teradata Vantage™ - Database Utilities*.

## Troubleshooting TLS

### TLS Logging

Use the `gtwcontrol` trace option to turn on tracing for TLS:

```
gtwcontrol --TLS require,trace=all
```

The debug traces are logged to `/var/opt/teradata/tdtemp/gtw/*.log`.

### Test the Signed Certificates and Private Keys on all Nodes

```
tlsutil -t
```

### Test the Signed Certificates and Private Keys on all Nodes and Show Which Are Expiring within 30 Days

```
tlsutil -t -e 30
```

### Check the Log File for Warnings About Expiring Certificates

Check `/var/log/messages`. For example, the following sample shows the TLS certificate is expiring in 11 days.

```
INFO: Teradata: 6210 #Event number 34-06210-00 (severity 0, category 10),
occurred on Thu Nov  5 13:44:25 2020 at 00
way, version
PDE:17.10c.00.28,TDBMS:17.10c.00.28,PDEGPL:17.10c.00.28,TGTW:17.10c.00.47cert,TD
GSS:17.10c.00.28
gtwTLSContext.cpp @290 (83900744): Thu Nov  5 13:44:25 2020
    The TLS certificate will expire after 11 days.
```

### Test the Signed Certificates and Private Keys on a Single Database

```
tlsutil -t -l
```

### Display Detailed Information to Help Diagnose Issues

```
tlsutil -c -v mydb.example.com
```

## Display Detailed Information to Help Diagnose Issues

```
tlsutil -c -v mydb.example.com
```

## Use the Data Dictionary Views to Troubleshoot Issues

The data dictionary stores information about TLS connections.

For example: Query the client confidentiality type in DBC.SessionInfoV:

```
select clientconftype from dbc.sessioninfov;
```

The type is determined by the client and represents the connection between client and the gateway or Unity:

Type	SSLMODE	Gateway Require Confidentiality	Client Data Encryption	Description
T	ALLOW or higher	ON or OFF	ON or OFF	TLS used for encryption.
E	DISABLE or ALLOW	ON (for Gateway Require Confidentiality) and/or ON (for Client Data Encryption)		TLS was not attempted because SSLMode was DISABLE or ALLOW. Connection was made to a legacy port. TDGSS used for encryption, and the application does not have the option to change this during the session.
U	DISABLE or ALLOW	OFF	OFF	TLS was not attempted. Unencrypted, and the application does not have the option to change this during the session.
O	DISABLE or ALLOW	OFF	ON or OFF	TLS was not attempted because SSLMode was DISABLE or ALLOW. May be encrypted using TDGSS or unencrypted, and the application has the option of changing this at any time. This situation primarily refers to BTEQ, which lets the user turn encryption on and off during the session. Other drivers don't permit this.
F	ALLOW or PREFER	ON (for Gateway Require Confidentiality) and/or ON (for Client Data Encryption)		TLS was attempted, but the TLS handshake failed because of TLS version mismatch, cipher mismatch, or certificate problem. Connection was made to a legacy port. This is a fallback to using TDGSS for encryption.



Type	SSLMODE	Gateway Require Confidentiality	Client Data Encryption	Description
R	PREFER	ON (for Gateway Require Confidentiality) and/or ON (for Client Data Encryption)		SSLMODE was set to PREFER, but a legacy port socket connection was made. TDGSS is used for encryption. This situation occurs because the client was unable to connect to port 443 for reasons such as blocked by the firewall.
S	PREFER	OFF	OFF	SSLMODE was set to PREFER, but a legacy port socket connection was made. Unencrypted. This situation occurs because the client was unable to connect to port 443 for reasons such as blocked by the firewall.

See LogOnOffV, QryLogClientAttrV, and SessionInfoV in *Teradata Vantage™ - Data Dictionary*.

# Optimizing Directory Searches

When a directory user logs on to Teradata Vantage, the authentication mechanism initiates a search of the directory to identify the user and validate user credentials. Depending on how the directory is configured, LDAP may encounter difficulties locating the directory user. TDGSS provides several options to help optimize directory searches.

## About Search Optimization Options

- If Teradata Vantage users are authenticated or authorized externally, evaluate the need to configure directory search properties to optimize search efficiency. See [Configuring LDAP Properties to Narrow the Search Base](#).

---

**Note:**

Teradata recommends that you configure the search properties if you manage Teradata Vantage users in a directory.

---

- For directories that require use of complete DNs in the user logon, you can configure an identity map to allow use of simple usernames in place of the DN. See [Using Identity Mapping](#).
- You can configure an identity search to efficiently find a user in the directory, regardless of the user location or whether the directory has a strict hierarchical structure. See [Using Identity Searches](#).

## Configuring LDAP Properties to Narrow the Search Base

You can configure certain LDAP properties on database nodes, and on the Unity server, if used, to help narrow the search base for directory objects to the children of specified parent objects, rather than searching the entire directory.

---

**Note:**

This feature is not dependent upon bind type.

1. Make changes to the TdgssUserConfigFile.xml as shown in [Making Changes to TdgssUserConfigFile.xml on Database Nodes](#).
2. Edit the LDAP needed search properties to enhance searches.

---

where:

Property	Description
LdapGroupBaseFQDN	Contains the FQDN of the directory object that contains group objects.

Property	Description
	When you authorize database users in a directory, you have the option to create role objects in the directory, and then map them to groups with user members. You can configure the <code>LdapGroupBaseFQDN</code> property to enhance the search for directory groups and speed user authorization. See <a href="#">LdapGroupBaseFQDN</a> .
<code>LdapUserBaseFQDN</code>	Contains the FQDN of a directory group object that contains directory user objects. You can configure this property to narrow the search base for directory users to enhance user authentication. See <a href="#">LdapUserBaseFQDN</a> .

## Working with Directory User Identification Options

### About Directory User Identification

When the directory authenticates a database user, TDGSS searches for user information in the directory based on the directory username specified in the logon. Directories use distinguished names (DNs) to uniquely name each directory user object, for example:

```
cn=ab111222,ou=northamerica,ou=useraccounts,dc=div,dc=corp,dc=com
```

However, requiring users to enter the entire DN can result in logon errors. In addition, the database may be able to log only part of the DN, due to object name length limitations.

To avoid having to enter the entire DN, it is common practice to allow users to specify the simple form of the username in a logon string, for example:

```
ab111222
```

The authentication process links the simple username to the DN in the directory.

Although it is generally good practice, allowing the use of simple usernames in the database logon string can present problems:

- Some directories do not allow a simple username in the logon string and force users to enter the entire DN at logons.
- Directories that do allow simple usernames may not efficiently bind them to the correct DNs.

### Comparing User Identification Options

You can configure user identification options to:

- Increase the directory efficiency of binding the simple username to the DN.

- Provide a link between the simple username and the DN in directories where the link does not exist, to allow directory users to logon with a simple username.

Identification Option	Description
<a href="#">Using Identity Mapping</a>	<p>An identity map defines a pattern and a rule that identifies the structure of the DN and how it relates to the simple user name.</p> <p>Identity maps:</p> <ul style="list-style-type: none"> <li>• Are easier to configure</li> <li>• Do not require a service bind</li> <li>• Are useful if the map can construct the user DN with only the information in the simple username, which is sometimes not possible.</li> </ul>
<a href="#">Using Identity Searches</a>	<p>An identity search describes search criteria that LDAP uses to locate the user in the directory, and also allows you to specify the search scope and a filter, for a more precise search.</p> <p>Identity searches:</p> <ul style="list-style-type: none"> <li>• Require a more complex configuration than identity maps</li> <li>• Can find any user regardless of its location in the directory</li> <li>• Require a service bind, which you must configure</li> <li>• Are useful when the directory does not have a strict hierarchical structure</li> </ul>

You can configure both identification options concurrently, if needed. See [Using Multiple IdentityMap and IdentitySearch Elements in Combination](#).

## Using Identity Mapping

An identity map defines a pattern relationship between the authcid expressed in the logon, and the full DN of the directory object that represents the user. The binding process uses an identity map to extrapolate the user DN from the information in a simple username. You can also define how the authcid is logged in the database.

### Prerequisites

- Identify all username formats used for external authentication logons. You should create either an identity map or identity search for each format used. See the topics beginning with [LDAP Logon Format Examples](#) for some common username forms.
- Determine whether or not LDAP can canonically create the directory user DN from the user authcid without consulting the directory. If not, identity mapping is not useful, and you should consider using an identity search.

### Identity Mapping Implementation Process

1. Make sure you understand the format and function of the TDGSS configuration files. See [About the TDGSS Configuration Files](#).

2. Create the identity map in a text editor such as vi or Notepad. See [Configuring an Identity Map](#).
3. Add the identity map to the TdgssUserConfigFile.xml on Teradata Vantage nodes. See [Making Changes to TdgssUserConfigFile.xml on Database Nodes](#).
4. If the configuration does not function properly or is not useful, you can revert to the previous configuration. See [Returning to an Old Configuration](#).

## Configuring an Identity Map

Identity maps require that you add <IdentityMap> elements to the TdgssUserConfigFile.xml on database nodes and to the TdgssUnityConfig.xml on Unity servers.

If you use identity mapping, you must configure an identity map (or identity search) for each variation of simple username format that your site allows. For an example that configures several simple username format variations, see [Use Case for Combined IdentityMap and IdentitySearch](#).

Create each identity map in the TdgssUserConfigFile.xml for each authentication mechanism employed by directory users for logging on to Vantage, for example:

```
<Mechanism Name="ldap">
  <MechanismProperties
    ...
  />
  <IdentityMap
    Match="Posix regular expression-encoded matching rule"
    Pattern="Substitution rule"/>
    DatabaseName="ab111222@div.corp.com"/>
</Mechanism>
```

## Sample Identity Map for Logging on with a UPN

You can configure the LDAP mechanism to create an identity map for usernames that logon with a FQDN, such as user@dom1.dom2.dom3, for example:

```
<Mechanism Name="ldap">
  <MechanismProperties
    ...
  />
  <IdentityMap
    Match="match"
    Pattern="pattern"/>
    DatabaseName="database_name"/>
</Mechanism>
```

**match**

A Posix regular expression representing a matching rule that shows how the username is divided into sub-strings. The individual substrings are enclosed by ( ).

Example: `([^\@]+)@([^\.]+)\.([^\.]+)\.([^\.]+)`

**pattern**

The substitution rule that determines how the map extrapolates a DN from the username substrings defined in the Match attribute.

Example: `uid=${1},ou=users,dc=${2},dc=${3},dc=${4}`

**database\_name**

Defines how the system rewrites the username so that the database can identify the user in a particular form.

The value `${1}` identifies the user in the database using only the uid portion of the logon, and drops the `${2}`, `${3}`, and `${4}` portions of the username.

Example: `${1}`

**Note:**

The identity map does not require a service bind.

## Explanation of the Pattern Substitution Schema

The Pattern attribute contains a substitution schema. The values refer to the four substrings created from the username by the Match attribute. The term `${1}` pertains to the userid, which is located in the first substring of the username. Since a UPN includes domain information, the Pattern attribute does not require complete values for the terms `${2}`, `${3}`, and `${4}`.

An identity map substitutes corresponding substrings from the logon username, for example:

For the username...	<code>\${1}</code>	<code>\${2}</code>	<code>\${3}</code>	<code>\${4}</code>	Result
ab111222@div.corp.com	ab111222	div	corp	com	uid=ab111222,ou=users,dc=div,dc=corp,dc=com
adminuser@sales.corp.com	adminuser	sales	corp	com	uid=adminuser,ou=users,dc=sales,dc=corp,dc=com
jsmith@div.corp.com	jsmith	div	corp	com	uid=jsmith,ou=users,dc=div,dc=corp,dc=com

## Sample Identity Map for Simple User Names

You can use the following identity map for any simple username specified in a valid logon, for example, jsmith. You must include domain information sufficient to construct the DN as part of the Pattern attribute, for example:

```
<Mechanism Name="ldap">
  <MechanismProperties
    ...
  />
  <IdentityMap
    Match="(.*)"
    Pattern="cn=${1},ou=people,dc=div,dc=corp,dc=com"/>
    DatabaseName="svc1_${0}"/>
</Mechanism>
```

### Note:

For sites using multiple directory services, where users normally log on using only simple uids, you can use the DatabaseName attribute to affix a string that represents the service to the authcid, to differentiate among possible duplicate usernames that may appear in the various services. Using the value shown in the IdentityMap example:

```
DatabaseName="svc1_${0}"
```

subsequent logons using the simple user name jsmith, identify the user as "svc1\_jsmith" in the database, where \${0} causes the database to use the entire authcid (jsmith).

## Sample of an Identity Map for NT-Styled Logons

You can use the following identity map for usernames expressed in Windows NT logon format, for example: td\ab111222.

```
<Mechanism Name="ldap">
  <MechanismProperties
    ...
  />
  <IdentityMap
    Match="[Tt][Dd]\\(\\(.+)"
    Pattern="cn=${1},ou=users,dc=td,dc=teradata,dc=com"/>
</Mechanism>
```

This identity map extracts no domain information from the username, but instead uses information in the Pattern attribute to construct the DN.

## Using Identity Searches

An identity search is similar in function to an identity map, except that an identity search actually searches the directory for the user DN rather than reconstructing it from the simple username.

If you configure one or more identity searches, the directory automatically performs a service bind when a user logs on. The service bind is automatic, so you do not need to set the LdapServiceBindRequired property, but you must configure other properties related to service binds. See [Working with Service Binds](#).

## Prerequisites

- Inform directory users about the allowable username formats, and configure identity searches only for allowable formats. For examples of common LDAP username logon forms, see [Use Case for Combined IdentityMap and IdentitySearch](#).
- You should configure TLS for protection for the service bind that automatically accompanies all identity searches.

## Identity Search Implementation Process

1. Make sure you understand the format and function of the configuration files. See [About the TDGSS Configuration Files](#).
2. Create the required identity search in a text editor, such as Notepad. See [Configuring an Identity Search](#).
3. Use the text editor to add the following items to the authentication mechanism(s) in the TdgssUserConfigFile.xml on Teradata Vantage nodes and to the TdgssUnityConfig.xml on the Unity server, if used. See [Making Changes to TdgssUserConfigFile.xml on Database Nodes](#).
  - Add the identity search created in step 2 above.
  - Add the LdapServiceFQDN and LdapServicePassword properties, and configure them as shown in [Directory Identification and Search Properties](#).

The LdapServicePasswordProtected property indicates whether the password is stored in encrypted form. You do not need to add this property to use the default setting (not protected). If you want to encrypt the password, use the tdspasswd tool to generate an encrypted password for the passphrase that is used to encrypt the private key file.

---

### Note:

You can store the password in plain text, but it is not recommended. If you use plain text, be sure to limit access to the TDGSS configuration files and the TDGSSCONFIG.GDO. See [Controlling Access to the Operating System](#).

---

Also see [LdapServicePasswordProtected](#).



- If the configuration is not useful, you can revert to the previous configuration. See [Returning to an Old Configuration](#).

## Configuring an Identity Search

You can configure an identity search for any valid username format.

For information on username specifications allowed for Teradata Vantage logons, see [Explanation of LDAP Logon Format Examples](#).

Create each identity search in the TdgssUserConfigFile.xml for each authentication mechanism employed by directory users for logging on to Vantage.

## Example Identity Search

In this example, users log on to Vantage using the NT-style logon `td/ab111222`.

```
</Mechanism>
<Mechanism Name="ldap">
  <MechanismProperties
    ...
  />
  <IdentitySearch
    Match="[Tt][Dd]\\(\\.(+)"
    Base="ou=user accounts,dc=td,dc=teradata,dc=com"
    Scope="subtree"
    Filter="(&(objectClass=user)(sAMAccountName=${1}))"/>
    BindName="${result}"
    DatabaseName="${1}"/>
  </Mechanism>
```

The IdentitySearch element contains attributes that define the parameters of a directory search, and cause TDGSS to conduct the search for each directory user logon:

Attribute Name	Attribute Value	Description
Match (required)	"[Tt][Dd]\\(\\.+)"	A regular Posix expression that matches the username (authcid).
Base (required)	"ou=user accounts,dc=td, dc=teradata,dc=com"	A pattern into which the search substitutes substrings from the Match attribute value and constructs the DN that it uses as the search base.
Scope (required)	"subtree"	A string that defines the search scope, from among these options: <ul style="list-style-type: none"> <li>• "base" requests a search of the object named in the Base attribute.</li> </ul>

Attribute Name	Attribute Value	Description
		<ul style="list-style-type: none"> <li>"one level" requests a search of the children of the Base object.</li> <li>"subtree" subtree requests a search of the entire subtree, starting with Base.</li> </ul>
Filter (required)	"(&(objectClass=user) (sAMAccountName=\${1}))"	A pattern into which the identity search substitutes substrings from the Match attribute value, and which the search uses for the search filter, as defined in IETF RFC 2254.
BindName (optional)	"\${result}"	<p>Defines how the system rewrites the username to bind to the directory.</p> <p>The default, BindName="\${result}" , maintains backward compatibility with older configurations. You can change to default based on directory requirements. For example, when using DIGEST-MD5 binding with directory services that require the DIGEST-MD5 user name to be "dn:" followed by the user's DN (common to many services), you can specify BindName="dn:\${result}" to prepend dn to the outcome of the Identity Search.</p> <p><b>Note:</b></p> <p>The DIGEST-MD5 authentication protocol used by LDAP is deprecated. Teradata strongly recommends you use simple binding with TLS protection, and stop using DIGEST-MD5.</p>
DatabaseName (optional)	"\${1}"	<p>Defines how the system rewrites the username so that the database can identify the user in a particular form.</p> <p>The value \${1} identifies the user in the database using only the uid portion of the logon, and drops the \${2}, \${3}, and \${4} portions of the username.</p>

#### Search Results:

Based upon a Windows domain TD, the existence of users ab111222 and xy333444 in the directory, and the search base and scope specified in the previous example, the identity search generates the following searches and results.

Username	Filter	\$(result)
td\ab111222	(&(objectClass=user) (sAMAccountName=ab111222))	CN=ab111222,OU=NorthAmerica, OU=User Accounts,DC=TD, DC=CORP,DC=COM
td\xy333444	(&(objectClass=user) (sAMAccountName=xy333444))	CN=xy333444,OU=NorthAmerica, OU=User Accounts,DC=TD, DC=CORP,DC=COM

Username	Filter	\$(result)
td\user1234	(&(objectClass=user) (sAMAccountName=user1234))	The search returns no results, which indicates that the user does not exist in the directory.

## Using Multiple IdentityMap and IdentitySearch Elements in Combination

Some sites define users in more than one directory subtree, while allowing users to employ a variety of username formats. To meet those diverse needs, you may need to configure a combination of IdentityMap and IdentitySearch elements, according to these guidelines:

- There is no limit to the number of IdentityMap and IdentitySearch elements you can add to the LDAP configuration.
- The elements can appear in the TdgssUserConfigFile.xml in any order.
- LDAP processes the IdentityMap and IdentitySearch elements in the order they appear in the mechanism configuration.
- LDAP authenticates the user with the first IdentityMap or IdentitySearch element that contains a Match expression that completely consumes the user name (authcid).
- If no identity map or identity search completely consumes the user name, LDAP attempts the bind with the simple username as it appears in the logon. If the directory cannot resolve the simple username to the user DN, the logon fails.
- If an identity search does not return exactly one object, the authentication fails.
- If a user defined in an identity map does not exist in the directory, the authentication fails.

---

### Note:

You can also use \${0} which represents the entire user name (authcid).

---

## Use Case for Combined IdentityMap and IdentitySearch

A common use of an identity map with an identity search is when you employ a directory with two trees. In this scenario, each tree defines only a subset of all users, but either tree can authenticate any user. Suppose that:

You have two trees in your directory:

- Tree A, named dc=div,dc=corp,dc=com and is in Windows domain DIV.
- Tree B, named dc=newyork,dc=corp,dc=com and is in Windows domain NEWYORK.

You want to allow any of several username formats in logons:

- UPN, for example, cc444555@div.com
- The user DN, for example, cn=cc444555,ou=users,dc=newyork,dc=corp,dc=com

You also want to allow users to logon without the domain name, so that the directory can authenticate the following usernames, which require the directory to use the DIV tree:

- cc444555
- div\cc444555
- cc444555@div.corp.com
- newyork\cc444555
- cc444555@newyork.corp.com

## Configuring Combined Identity Map and Identity Search

```
<Mechanism Name="ldap">
  <MechanismProperties
    ...
  />
  <IdentitySearch
    Match="([^\]=]+)\\[([^\]=]+\]"
    Base="dc=${1},dc=corp,dc=com"
    Scope="subtree"
    Filter="(&(objectClass=user)(sAMAccountName=${1}))"/>
  <IdentitySearch
    Match="[@=]+@[^\.=]+\.[^\.=]+\.[^\.=]+"
    Base="dc=${1},dc=${2},dc=${3}"
    Scope="subtree"
    Filter="(&(objectClass=user)(userPrincipalName=${0}))"/>
  <IdentitySearch
    Match="([^\]=@=)+"
    Base="dc=div,dc=corp,dc=com"
    Scope="subtree"
    Filter="(&(objectClass=user)(sAMAccountName=${1}))"/>
</Mechanism>
```

### Note:

Most sites have only one way of identifying users, so you normally need only a single identity search or identity map.

## Search Results for Combined Identity Map and Identity Search

User Name	Identity Search	Resulting Search Base	Resulting Search Filter
cc444555	3rd	dc=div, dc=corp,dc=com	(&(objectClass=user)(sAMAccountName=cc444555))

User Name	Identity Search	Resulting Search Base	Resulting Search Filter
div\cc444555	1st	dc=div, dc=corp,dc=com	(&(objectClass=user)(sAMAccountName=cc444555))
newyork\cc444555	1st	dc=newyork, dc=corp,dc=com	(&(objectClass=user)(sAMAccountName=cc444555))
cc444555@div.corp.com	2nd	dc=div, dc=corp,dc=com	(&(objectClass=user) (userPrincipalName=cc444555@div.corp.com))
cc444555@newyork\corp.com	2nd	dc=newyork, dc=corp,dc=com	(&(objectClass=user) (userPrincipalName=cc444555@newyork.corp.com))

## Search Characteristics

If LDAP uses the configuration shown in [Configuring Combined Identity Map and Identity Search](#) to process a username in DN form, for example:

```
cn=d1160010,ou=northamerica,ou=user accounts,dc=td,dc=teradata,dc=com
```

the search does not use a filter because every filter excludes the equal sign characters in the DN, and as a result, the search finds no matches and uses the username without change.

The [Configuring Combined Identity Map and Identity Search](#) example does not work in the corp environment because the two trees, dc=newyork,dc=corp,dc=com and dc=div,dc=corp,dc=com exist in separate directory environments. For this kind of configuration to work, the trees must be managed by the same directory service.

### Note:

The `${0}` in the second configured identity search refers to the entire username (authcid) string. Since the search pattern enforces the naming convention that is in both the DIV and NEWYORK domains for the userPrincipalName attribute, the authcid is adequate.

## Using Identity Map and Identity Search for Multiple Directory Services

You can put as many `<IdentityMap>` or `<IdentitySearch>` elements as needed into the `<Canonicalizations>` section of the `<LdapConfig>` section of the `TdgssUserConfigFile.xml`. Name each service using the `Id` attribute. Link the individual `<IdentityMap>` or `<IdentitySearch>` elements back to the service using the service name (`Id`) in the canonicalization's `Ref` attribute.

For information on using the `<LdapConfig>` element in the TDGSS configuration, see [Adding Identity Map and Identity Search Elements to the LdapConfig](#).

## Sample Configuration for Multiple Directory Services

This example configures an Identity Search for each directory service. Configuring Identity Maps for multiple services is similar.

```
<LdapConfig>
  <Tls .../>
  <Services>
    <Service Id="svc1" LdapClientMechanism="simple".../>
    <Service Id="svc2" LdapClientMechanism="simple".../>
  </Services>
  <Canonicalizations>
    <IdentitySearch
      Ref="svc1"
      Pattern="(.+ )@td.teradata.com"
      Base="dc=td,dc=teradata,dc=com"
      Filter="(uid=${1})"
      DatabaseName="svc1_${1}"
      BindName="dn:${result}"/>
    <IdentitySearch
      Ref="svc2"
      Pattern="(.+ )@pioneerstd.teradata.com"
      Base="dc=pioneerstd,dc=teradata,dc=com"
      Filter="(uid=${2})"
      DatabaseName="svc2_${1}"/>
    </Canonicalizations>
  </LdapConfig>
```

For a description of the attributes used in an <IdentitySearch> element, see [Configuring an Identity Search](#).

# Configuring LDAP for Site-Aware Authentication

A network may contain multiple directory servers that can authenticate directory users logging on to one or more Teradata Vantage systems. You can configure the `LdapServerName` property on each Vantage system, and on the Unity servers, if used, to use site awareness to automatically authenticate users in one or more selected directories local to the database or Unity server to which users log on:

- Within a domain. See [Configuring Site Aware User Authentication in a Windows Domain](#).
- Across all domains in a global catalog. See [Configuring Site Aware Authentication in a Global Catalog](#).

## Prerequisites

- The directory must be Active Directory
- The administrator that sets up site awareness must have:
  - Access to the `ldapsearch` utility and the `dig` or `nslookup` tool
  - Read privileges for the configuration context in the directory
  - Linux `tdtrusted` group login privileges on the database nodes and the Unity server, if used.

## Configuring Site Aware User Authentication in a Windows Domain

When a Teradata Vantage system connects to several directory servers, you can configure the `LdapServerName` property in the database, for any external authentication mechanism, to force authentication through one or more directory servers that are local to the database.

## Configuration Process

1. Determine the `configurationNamingContext` for the domain, which is required as the search base for locating individual directory sites. See [Locating the Configuration Naming Context](#).
2. Locate the site objects that define locations in the domain. See [Locating the Site Objects in a Domain](#).
3. Find the directories that serve a site that contains a Vantage system or Unity server. See [Finding Directories that Serve a Site](#).
4. Configure the `LdapServerName` property to specify an SRV RR that yields the directories local to the site that contains the Vantage system or Unity server. See [Configuring Site-Aware SRV Resource Records in TDGSS](#).

## Locating the Configuration Naming Context

You can access site information for a domain using the configuration naming context for the domain. The RootDSE object for each directory in the domain contains a `configurationNamingContext` attribute, with a CN value that is the naming context.

To find the `configurationNamingContext`, run the `ldapsearch` utility from the Vantage command prompt, using the DNS name of any directory server in the domain:

```
ldapsearch -x -b "" -s base -H ldap://dir_name:port configurationNamingContext
```

### Note:

The preceding `ldapsearch` command uses the `-H ldap` scheme, which usually requires only an anonymous bind, that is, it does not use credentials to authenticate the user executing the command.

#### **-x**

Specifies simple binding for the anonymous authentication of the user that executes the command.

#### **-b ""**

Specifies the base of the search, in this case the default base `""`.

#### ***dir\_name:port***

The DNS name of a directory server in the domain and the server port designation. If no port is specified, `ldapsearch` uses the default port for the scheme. See [LdapServerName](#).

#### **configurationNamingContext**

Identifies the object that the command locates.

The command produces output similar to:

```
# extended LDIF
#
# LDAPv3
# base <> with scope base
# filter: (objectclass=*)
# requesting: configurationNamingContext
#
dn: configurationNamingContext: CN=Configuration,DC=DOMAIN1,DC=COM
# search result
search: 2
result: 0 Success
```



```
# numResponses: 2
# numEntries: 1
```

where CN=Configuration,DC=DOMAIN1,DC=COM is the distinguished name that you must use to construct a search base in [Locating the Site Objects in a Domain](#).

## Locating the Site Objects in a Domain

Before you can find the directories that serve a site, you must locate the site objects in the domain and determine the CN of the site where the database is located.

## Specifying a Binding Method for an Ldapsearch

The directory must authenticate the user that executes the Ldapsearch command before it allows the user access to site information. The command must specify one of the following sets of authentication information, depending on the binding scheme the directory uses for user authentication.

Binding Method	Ldapsearch Options	Description
Simple	<code>-x -D <i>user_principal_name</i> -W -ZZ</code>	<ul style="list-style-type: none"> <li><code>-x</code>, <code>-D</code>, <code>-W</code> and <code>-ZZ</code> specify simple binding and associated protection.</li> <li><code>user_principal_name</code> identifies (in UPN form) the user running the command, for example: <code>user@domain</code></li> </ul>
DIGEST-MD5 [Deprecated]	<code>-Y DIGEST-MD5 -U <i>simple_user_name</i></code>	<ul style="list-style-type: none"> <li><code>-Y DIGEST-MD5</code> specifies the binding type</li> <li><code>-U</code> is the simple user name, derived from the directory <code>sAMAccountName</code> attribute</li> </ul> <p><b>Note:</b> The DIGEST-MD5 authentication protocol used by LDAP is deprecated. Teradata strongly recommends you use simple binding with TLS protection, and stop using DIGEST-MD5.</p>

## Searching for Site Objects

From the Vantage command prompt, use one of the following Ldapsearch commands to find the site objects in a domain:

- For simple binding:

```
ldapsearch -x -D user@domain -W -H ldap://dir_name -ZZ \
  -b "CN=Sites,CN=Configuration,DC=domain,DC=COM" -s one
  "(objectClass=site)" cn
```

- For DIGEST-MD5 binding [Deprecated]:

---

**Note:**

The DIGEST-MD5 authentication protocol used by LDAP is deprecated. Teradata strongly recommends you use simple binding with TLS protection, and stop using DIGEST-MD5.

---

```
ldapsearch -Y DIGEST-MD5 -U user_name -H ldap://dir_name \
  -b "CN=Sites,CN=Configuration,DC=domain,DC=COM" -s one
  "(objectClass=site)" cn
```

**-H ldap://**

Specifies the directory scheme.

**dir\_name**

Specifies the DNS name of a directory server in the domain, for example, the dir\_name in [Locating the Configuration Naming Context](#).

**-b "CN=Sites,CN=Configuration,DC=DOMAIN1,DC=COM"**

Specifies the scope of the search, in this case, all sites in the domain configuration naming context identified in [Locating the Configuration Naming Context](#).

**-s one "(objectClass=site)" cn**

Identifies the parent object one level above the site object class as the search base.

The ldapsearch command returns output similar to:

```
# extended LDIF
#
# LDAPv3
# base <CN=Sites,CN=Configuration,DC=DOMAIN1,DC=COM> with scope oneLevel
# filter: (objectClass=site)
# requesting: cn

# ChicagoDiv, Sites, Configuration, DOMAIN1.COM
dn: CN=ChicagoDiv,CN=Sites,CN=Configuration,DC=DOMAIN1,DC=COM
cn: ChicagoDiv

# NewYorkDiv, Sites, Configuration, DOMAIN1.COM
```

```
dn: CN=NewYorkDiv,CN=Sites,CN=Configuration,DC=DOMAIN1,DC=COM
cn: NewYorkDiv

# SanDiegoHQ, Sites, Configuration, DOMAIN1.COM
dn: CN=SanDiegoHQ,CN=Sites,CN=Configuration,DC=DOMAIN1,DC=COM
cn: SanDiegoHQ

# search result
search: 2
result: 0 Success
```

where SanDiegoHQ is the location of a Teradata Vantage system that requires site aware, local authentication of users from other locations.

## Finding Directories that Serve a Site

1. After you choose a site, you can search the DNS SRV RRs to find all the local directories that serve the site:

```
_ldap._tcp. site ._sites. domain
```

### ***site***

The name of a site identified in [Locating the Site Objects in a Domain](#).

### ***domain***

The name of the DNS domain that contains the site.

2. Convert the site name into DNS SRV RR format. For example, you can convert CN=SanDiegoHQ,CN=Sites,CN=Configuration,DC=DOMAIN1,DC=COM into a DNS SRV RR format, \_ldap.\_tcp.SanDiegoHQ.\_sites.DOMAIN1.com.
3. Use the Linux dig utility to search the DNS for directories that serve the site:

```
dig +tcp -t SRV _ldap._tcp.SanDiegoHQ._sites.DOMAIN1.com
```

The dig command produces output similar to:

```
<<>> DiG 9.2.4 <<>> +tcp -t SRV _ldap._tcp.SanDiegoHQ._sites.domain1.com;
;; global options: printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 49397
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 2, AUTHORITY: 5, ADDITIONAL: 8

;; QUESTION SECTION:
;_ldap._tcp.SanDiegoHQ._sites.domain1.com. IN      SRV
```

```
;; ANSWER SECTION:
_ldap._tcp.SanDiegoHQ._sites.domain1.com. 86400 IN SRV 0 100
389 sdhq7602.DOMAIN1.COM.
_ldap._tcp.SanDiegoHQ._sites.domain1.com. 86400 IN SRV 0 100
389 sdhq7603.DOMAIN1.COM.

;; AUTHORITY SECTION:
_sites.domain1.com. 86400 IN NS sdint7857.DOMAIN1.COM.
_sites.domain1.com. 86400 IN NS socal7210.DOMAIN1.COM.
_sites.domain1.com. 86400 IN NS socal7211.DOMAIN1.COM.
_sites.domain1.com. 86400 IN NS socal7212.DOMAIN1.COM.
_sites.domain1.com. 86400 IN NS sdhq7562.DOMAIN1.COM.

;; ADDITIONAL SECTION:
sdhq7544.DOMAIN1.COM. 1800 IN A 141.206.23.0
sdhq7602.DOMAIN1.COM. 1800 IN A 141.206.48.0
sdhq7603.DOMAIN1.COM. 1800 IN A 141.206.66.0
socal7210.DOMAIN1.COM. 1800 IN A 141.85.51.0
socal7211.DOMAIN1.COM. 1800 IN A 141.85.130.0
socal7212.DOMAIN1.COM. 1800 IN A 141.85.16.0
sdhq7562.DOMAIN1.COM. 1800 IN A 141.85.77.0
sdint7857.DOMAIN1.COM. 1800 IN A 141.29.162.0

;; Query time: 11 msec
;; SERVER: ip_address8#53(ip_address8)
;; WHEN: Wed Dec 16 14:19:06 2009
;; MSG SIZE rcvd: 397
```

where the ANSWER SECTION lists the two directories that serve the SanDiegoHQ site.

4. Do one of the following:

- If the ANSWER SECTION for the output lists one or more directories, go on to [Configuring Site-Aware SRV Resource Records in TDGSS](#), and use the DNS SRV RR formatted site name you used earlier in this procedure.
- If the ANSWER SECTION for the output of your dig command does not list one or more directories, redo the first two steps.

## Configuring Site-Aware SRV Resource Records in TDGSS

You can configure the authentication mechanism for site-aware selection of a local directory by editing the LdapServerName property to a DNS SRV RR formatted site name, for example:

```

<Mechanism Name="ldap">
  <MechanismProperties
    MechanismEnabled="yes"
    AuthorizationSupported="no"
    ...
    LdapClientMechanism="simple"
    LdapServerName="_ldap._tcp.SanDiegoHQ._sites.domain1.com"
    ...
  />
</Mechanism>

```

Configuration Option	Description
<Mechanism Name="ldap">	Site awareness requires directory authentication of the user, using the LDAP mechanism.
MechanismEnabled="yes"	The LDAP mechanism must be enabled.
AuthorizationSupported="no"	Site awareness functions whether or not the directory authorizes the user.
LdapClientMechanism="simple"	The example is for a system using simple binding, but site awareness also supports DIGEST-MD5 binding.
LdapServerName="_ldap._tcp.SanDiegoHQ._sites.domain1.com"	This setting requires a DNS SRV RR formatted site name, which identifies the local site directories available to authenticate the user.

**Note:**

In addition to performing the TDGSS configuration shown above, if the DNS service for the domain in which the database or Unity server resides is not the one where Active Directory registers its site-aware DNS SRV RRs (that is, a “foreign” service), then you must also manually configure the site-aware SRV RRs in the foreign DNS service.

For DNS SRV RR configuration instructions, see [LdapServerName](#).

When you configure the LdapServerName property for site awareness, the authenticating mechanism selects a directory at random from among the available local directories for the site.

## Configuring Site Aware Authentication in a Global Catalog

A Global Catalog (GC) contains a forest composed of several domains. The GC incorporates the DITs for all the domains into a single, enterprise-wide super directory.

When directory users connect to a Teradata Vantage system or Unity server across domains that do not talk directly to each other, you can configure the LdapServerName property to specify a GC server local to the connection, even if the location is outside the domain inhabited by the user.

In an Active Directory forest, the GC normally registers itself in the DNS, using the form:

```
_gc._tcp.root_domain
```

where *root\_domain* is the name of the super-domain that owns the GC.

## Configuration Process

1. Find the root domain name for use in finding GC servers. See [Finding the Root Domain Name](#).
2. Find the directory servers that function as GC servers. See [Finding All GC Servers in the Forest](#).
3. Find the sites that have GC servers. See [Finding the Available Sites in the Forest](#).
4. Verify that the database site has a GC server. See [Choosing a Site](#).
5. Configure the LdapServerName property. See [Configuring TDGSS](#).

## Finding the Root Domain Name

All Active Directory servers in a forest publish the rootDomainNamingContext attribute in their RootDSE object. This attribute contains the DN of the GC naming context. It also corresponds to the DNS domain name where the GC registers itself.

1. Use the ldapsearch utility to obtain the DNS domain name:

```
ldapsearch -x -b "" -s base -H ldap://dir_name rootDomainNamingContext
```

The meanings of -x, -b, "", -s base, and -H:// are similar to those shown in [Locating the Configuration Naming Context](#).

*dir\_name* is the DNS name of a directory in the root domain.

The ldapsearch command produces output similar to:

```
# extended LDIF
#
# LDAPv3
# base <> with scope base
# filter: (objectclass=*)
# requesting: rootDomainNamingContext
#
#
dn:
rootDomainNamingContext: DC=ROOTDOMAIN,DC=COM

# search result
search: 2
result: 0 Success
```

```
# numResponses: 2  
# numEntries: 1
```

where the rootDomainNamingContext is DC=ROOTDOMAIN,DC=COM.

2. You can derive the DNS domain name from the value of the rootDomainNamingContext by removing the DC=, resulting in the DNS domain name ROOTDOMAIN.COM.

---

**Note:**

Since domain names are not case sensitive, you can use the name rootdomain.com, where required, for the remaining tasks in this procedure.

---

## Finding All GC Servers in the Forest

To manage site awareness through the GC you must determine which directory servers function as GC servers, using either the dig or nslookup utility.

This task is required because each directory server in the forest may be set up as:

- A GC server only
  - Both a GC server and a domain server
  - A domain server only
1. Use the following command to generate a list of GC servers:

```
dig +tcp -t SRV _gc._tcp.rootdomain.com
```

**Note:**

For instructions on finding the value of *root\_domain*, see [Finding the Root Domain Name](#).

The dig command produces output similar to:

```
;; global options: printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 65377
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 10, AUTHORITY: 5, ADDITIONAL: 17

;; QUESTION SECTION:
_gc._tcp.rootdomain.com.      IN      SRV

;; ANSWER SECTION:
_gc._tcp.rootdomain.com. 600     IN      SRV      0 100
3268 sdhq7602.DOMAIN1.COM.
_gc._tcp.rootdomain.com. 600     IN      SRV      0 100
3268 sdhq7603.DOMAIN1.COM.
_gc._tcp.rootdomain.com. 600     IN      SRV      0 100
3268 alban140.DOMAIN1.COM.
_gc._tcp.rootdomain.com. 600     IN      SRV      0 100
3268 china7830.DOMAIN2.COM.
_gc._tcp.rootdomain.com. 600     IN      SRV      0 100
3268 india7831.DOMAIN3.COM.
_gc._tcp.rootdomain.com. 600     IN      SRV      0 100
3268 japan7832.DOMAIN4.COM.
_gc._tcp.rootdomain.com. 600     IN      SRV      0 100
3268 sdhq7544.DOMAIN1.COM.

;; AUTHORITY SECTION:
_tcp.rootdomain.com.      86400   IN      NS       sdint7857.DOMAIN1.COM.
_tcp.rootdomain.com.      86400   IN      NS       socal7210.DOMAIN1.COM.
_tcp.rootdomain.com.      86400   IN      NS       socal7211.DOMAIN1.COM.
_tcp.rootdomain.com.      86400   IN      NS       socal7212.DOMAIN1.COM.
_tcp.rootdomain.com.      86400   IN      NS       sdhq7562.DOMAIN1.COM.

;; ADDITIONAL SECTION:
alban140.DOMAIN1.COM. 1800    IN      A        ip_address10
```

The ANSWER SECTION lists the directory servers that have GC server capabilities, and it includes the alban140.DOMAIN1.COM server used in previous tasks. Note that 3268 is the default port for GC servers.



2. Choose a GC directory in the ANSWER SECTION to facilitate site aware authentication of users who cross domain boundaries when they log on to a Teradata Vantage system or Unity server.

## Finding the Available Sites in the Forest

Once you choose a GC server, you must use the `ldapsearch` command to search for sites that contain GC servers that can authenticate users.

1. Use the `ldapsearch` command to locate the `configurationNamingContext` for the forest:

```
ldapsearch -x -b "" -s base -H ldap://
GC_server_name:port configurationNamingContext
```

### ***GC\_server\_name***

The DNS name of the GC server. For instructions on finding the server, see [Finding All GC Servers in the Forest](#).

### ***port***

The port number for the `GC_server_name`.

The `ldapsearch` command produces output similar to the following:

```
# extended LDIF
#
# LDAPv3
# base <> with scope base
# filter: (objectclass=*)
# requesting: configurationNamingContext
#
#
dn:
configurationNamingContext: CN=Configuration,DC=ROOTDOMAIN,DC=COM

# search result
search: 2
result: 0 Success

# numResponses: 2
# numEntries: 1
```

where the value of the `configurationNamingContext` attribute is the `-b` search base you must use to find sites.

2. Use the value of the configurationNamingContext attribute to construct an ldapsearch command that lists the sites served by the GC server, that is, sites at which the GC server can locally authenticate users, based on the binding scheme used by the site.

- For simple binding:

```
ldapsearch -x -D user_principal_name -W -H ldap://GC_server_name:port -ZZ \
  -b "CN=Sites,config_naming_context" -s one "(objectClass=site)" cn
```

- For DIGEST-MD5 binding [Deprecated]:

---

**Note:**

The DIGEST-MD5 authentication protocol used by LDAP is deprecated. Teradata strongly recommends you use simple binding with TLS protection, and stop using DIGEST-MD5.

---

```
ldapsearch -Y DIGEST-MD5 -U user_name -H ldap://GC_server_name:port \
  -b "CN=Sites,config_naming_context" -s one "(objectClass=site)" cn
```

***user\_principal\_name***

The UPN for the user running the command.

***GC\_server\_name***

The GC server selected in [Finding All GC Servers in the Forest](#).

***port***

The GC server default port.

***config\_naming\_context***

The value of the configurationNamingContext attribute. See [Finding the Root Domain Name](#).

***user\_name***

The domain username for the user running the command.

---

**Note:**

For command options not described in the preceding table, see [Working with Ldapsearch](#).

---

The ldapsearch command returns output similar to:

```
# extended LDIF
#
# LDAPv3
```

```

# base <CN=Sites,CN=Configuration,DC=ROOTDOMAIN,DC=COM> with scope one
# filter: (objectClass=site)
# requesting: cn

# China, Sites, Configuration, ROOTDOMAIN.COM
dn: CN=China,CN=Sites,CN=Configuration,DC=ROOTDOMAIN,DC=COM
cn: China

# NewYorkDiv, Sites, Configuration, ROOTDOMAIN.COM
dn: CN=NewYorkDiv,CN=Sites,CN=Configuration,DC=ROOTDOMAIN,DC=COM
cn: NewYorkDiv

# SanDiegoHQ, Sites, Configuration, ROOTDOMAIN.COM
dn: CN=SanDiegoHQ,CN=Sites,CN=Configuration,DC=ROOTDOMAIN,DC=COM
cn: SanDiegoHQ

# India, Sites, Configuration, ROOTDOMAIN.COM
dn: CN=India,CN=Sites,CN=Configuration,DC=ROOTDOMAIN,DC=COM
cn: India

# Japan, Sites, Configuration, ROOTDOMAIN.COM
dn: CN=Japan,CN=Sites,CN=Configuration,DC=ROOTDOMAIN,DC=COM
cn: Japan

# ChicagoDiv, Sites, Configuration, ROOTDOMAIN.COM
dn: CN=ChicagoDiv,CN=Sites,CN=Configuration,DC=ROOTDOMAIN,DC=COM
cn: ChicagoDiv

# search result
search: 2
result: 0 Success

# numResponses: 7
# numEntries: 6

```

The example output shows three sites, China, India, and Japan, which are not listed in the search of the single domain shown in [Locating the Site Objects in a Domain](#), and therefore represent separate domains within the forest.

## Choosing a Site

Choose a site that contains the database or Unity server you are configuring, and verify that the site has one or more GC directories.

1. Choose a site that contains the Teradata Vantage system you are configuring.
2. Run the following commands to identify the GC servers for the site.

```
dig +tcp -t SRV _ldap._tcp.site_name._sites.gc._msdcs.rootdomain.com
```

### **site\_name**

The name of a site that appears on the list of sites generated in step 2 of [Finding the Available Sites in the Forest](#), and that is the location of the Teradata Vantage system you are configuring.

### **rootdomain**

The name of the super-domain for the GC found in [Finding the Root Domain Name](#).

The dig command returns output similar to:

```
; <<>> DiG 9.2.4 <<>> +tcp -t
SRV _ldap._tcp.SanDiegoHQ._sites.gc._msdcs.rootdomain.com
;; global options: printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 2228
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 3, AUTHORITY: 5, ADDITIONAL: 9

;; QUESTION SECTION:
_ldap._tcp.SanDiegoHQ._sites.gc._msdcs.rootdomain.com. IN SRV

;; ANSWER SECTION:
_ldap._tcp.SanDiegoHQ._sites.gc._msdcs.rootdomain.com. 600 IN SRV 0 100
3268 susday7542.ROOTDOMAIN.COM.
_ldap._tcp.SanDiegoHQ._sites.gc._msdcs.rootdomain.com. 600 IN SRV 0 100
3268 sdhq7544.DOMAIN1.COM.
_ldap._tcp.SanDiegoHQ._sites.gc._msdcs.rootdomain.com. 600 IN SRV 0 100
3268 sdhq7603.DOMAIN1.COM.

;; AUTHORITY SECTION:
_msdcs.rootdomain.com. 86400 IN NS socal7210.DOMAIN1.COM.
_msdcs.rootdomain.com. 86400 IN NS socal7211.DOMAIN1.COM.
_msdcs.rootdomain.com. 86400 IN NS socal7212.DOMAIN1.COM.
_msdcs.rootdomain.com. 86400 IN NS sdhq7562.DOMAIN1.COM.
_msdcs.rootdomain.com. 86400 IN NS sdint7857.DOMAIN1.COM.

;; ADDITIONAL SECTION:
sdhq7544.TD.TERADATA.COM. 1800 IN A 141.206.23.0
sdhq7602.TD.TERADATA.COM. 1800 IN A 141.206.48.0
sdhq7603.TD.TERADATA.COM. 1800 IN A 141.206.66.0
```

```
socal7210.TD.TERADATA.COM. 1800 IN      A      141.85.51.0
ssocal7211.TD.TERADATA.COM. 1800 IN      A      141.85.130.0
socal7212.TD.TERADATA.COM. 1800 IN      A      141.85.16.0
sdhq7562.TD.TERADATA.COM.  1800 IN      A      141.85.77.0
sdint7857.TD.TERADATA.COM. 1800 IN      A      141.29.162.0

;; Query time: 16 msec
;; SERVER: ip_address8#53(ip_address8)
;; WHEN: Wed Dec 16 17:17:50 2009
;; MSG SIZE rcvd: 46
```

where the ANSWER SECTION lists the GC servers for the site.

3. If the ANSWER SECTION contains one or more entries, copy the SRV RR service name for the GC from the QUESTION SECTION, for example, `_ldap._tcp.DaytonSDC._sites.gc._msdcs.teradata.com`, to use in [Configuring TDGSS](#).

## Configuring TDGSS

After verifying that the SRV RR service name for the GC can find the GC servers for a site, configure the `LdapServerName` property with the SRV RR service name for the site, for example:

```
<Mechanism Name="ldap">

  <MechanismProperties
    MechanismEnabled="yes"
    AuthorizationSupported="no"
    ...
    LdapClientMechanism="simple"
    LdapServerName="_ldap._tcp.SanDiegoHQ._sites.rootdomain.com"
    LdapServerPort="0"
    ...
  />

</Mechanism>
```

### Note:

You can configure other properties for the LDAP mechanism, if needed. For instructions, see [Changing the TDGSS Configuration](#).

Configuration Option	Description
<code>&lt;Mechanism Name="ldap"&gt;</code>	Site awareness requires directory authentication of the user, using the LDAP mechanism.

Configuration Option	Description
MechanismEnabled="yes"	The LDAP mechanism must be enabled.
AuthorizationSupported="no"	Site awareness functions whether or not the directory authorizes the user.
LdapClientMechanism="simple"	<p>The example is for a system using simple binding. Site awareness also supports DIGEST-MD5 binding.</p> <p><b>Note:</b> The DIGEST-MD5 authentication protocol used by LDAP is deprecated. Teradata strongly recommends you use simple binding with TLS protection, and stop using DIGEST-MD5.</p>
LdapServerName="_ldap._tcp.SanDiegoHQ._sites.rootdomain.com"	This setting requires a DNS SRV RR formatted site name, which identifies the local GC directories available to authenticate the user.

When you configure the LdapServerName property for GC site awareness, LDAP selects a directory at random from among the available GC directories for the site.

# Configuring LDAP to Use Multiple Directory Services

A network may contain multiple directory services. A directory service is made up of one or more directory servers that replicate the same directory structure.

A company with multiple directory services may or may not need to configure LDAP to address the services separately:

- If the directory services are all in the same forest and are all visible through a Global Catalog (GC), LDAP can authenticate users to access Teradata Vantage in the GC, and no special set up is required. However, for organizations with widely scattered locations, you may find it beneficial to configure LDAP authentication for site awareness. See [Configuring Site Aware Authentication in a Global Catalog](#).
- Sometimes directory services are entirely separate and it is not possible to connect them through a GC, for example, when a company acquires or merges with another company. If database users must be authenticated by multiple directory services, you must configure a separate entry in the TdgssUserConfigFile.xml for each service.

## Prerequisites

- Directory types can vary among services, but they must all be LDAPv3-compliant.
- The administrator that sets up site awareness must have:
  - Access to the tdgssauth or tdsbind utilities
  - Administrator privileges in the directory
  - Administrator and tdtrusted privileges in the database
  - Administrator privileges on the Unity server, if used

## Implementation Overview

1. Make a backup copy of the TdgssUserConfigFile.xml file.
2. Revise TdgssUserConfigFile.xml to include the elements and properties required to support authentication using multiple directory services. See [Adding Multiple Directory Services to the TDGSS Configuration](#).
3. Verify the configuration is correct:
  - a. Run tdgssstestcfg to test the configuration. It launches a test environment in a new shell that contains the updates to the configuration file.

```
/opt/teradata/tdgss/bin/tdgssstestcfg
```

- b. Run the tdgssauth utility to test the new configuration before you commit the changes to the TDGSSCONFIG GDO.

See [Working with tdgssauth](#).

- c. Exit the test shell:

```
exit
```

- d. Continue editing and testing until the configuration is correct.
4. Run the `run_tdgssconfig` utility to update the TDGSSCONFIG GDO.
5. If `run_tdgssconfig` indicates that a TPA reset is required, run `tpareset`.

```
tpareset "use updated TDGSSCONFIG GDO"
```

6. If users log on through Unity, duplicate the configuration on the Unity server and all connected database systems. See [Using <LdapConfig> with Unity](#).
7. Configure Teradata directory objects and make directory user mappings in each authenticating directory, based on the directory features used.

## Adding Multiple Directory Services to the TDGSS Configuration

When database users are authenticated by a single directory service, all necessary TDGSS configuration can be done in the LDAP mechanism section of the `TdgssUserConfigFile.xml`, or the `TdgssUnityConfig.xml` for Unity, if used.

When database users are authenticated in multiple directory services, you must create a new `<LdapConfig>` section in the `TdgssUserConfigFile.xml`, and populate it with the needed subsections and properties, on each Teradata Vantage system served by the directories, and on the Unity server, if used.

For general information, see [Changing the TDGSS Configuration](#).

## Preparing to Edit the `TdgssUserConfigFile.xml`

1. On the Teradata Vantage node with the lowest ID number, navigate to the directory that provides access to `TdgssUserConfigFile.xml`.

```
cd /opt/teradata/tdat/tdgss/site
```

2. Make a backup copy of the `TdgssUserConfigFile.xml` and save it according to your site standard backup procedures, in case you need to return to the current configuration.
3. Open a text editor, such as `vi`, and bring up a working copy of the TDGSS user configuration file.

```
vi TdgssUserConfigFile.xml
```

4. Edit the `TdgssUserConfigFile.xml` as specified in the topics that follow, according to the Editing Guidelines for each property shown in the topics beginning with [Mechanism Properties](#).



## Preparing to Edit the TdgssUnityConfigFile.xml

1. If Unity is used, for each Unity server navigate to the directory that provides access to TdgssUnityConfig.xml.

```
cd /etc/opt/teradata/config/unity
```

2. Make a backup copy of the TdgssUnityConfig.xml and save it according to your site standard backup procedures, in case you need to return to the current configuration.
3. Open a text editor, such as vi, and bring up a working copy of the TDGSS Unity configuration file.

```
vi TdgssUnityConfigFile.xml
```

4. Edit the TdgssUnityConfigFile.xml as specified in the topics that follow, according to the Editing Guidelines for each property shown in the topics beginning with [Mechanism Properties](#).

## Disabling the Existing LDAP Mechanism

1. Copy the current LDAP mechanism configuration and save it for use in the new <LdapConfig> section, or if you need to return to the current configuration.
2. Copy the UseLdapConfig property from the LDAP mechanism in the TdgssLibraryConfigFile.xml into the LDAP mechanism in the TdgssUserConfigFile.xml, and set the property to yes. This directs the system to use the <LdapConfig> section, which can contain multiple directory service configurations. For example:

```
UseLdapConfig="yes"
```

## Creating the <LdapConfig> Section in the TdgssUserConfigFile.xml

The <LdapConfig> section must contain at least:

- 1 <Service>
  - 1 canonicalization, that is, 1 <Identity Map> or <Identity Search> element
1. Create the <LdapConfig> section by adding the following elements to the TdgssUserConfigFile.xml. The <LdapConfig> section must be located directly following the <Mechanisms> section and on the same level as the <Mechanisms> section.

For example:

```
<LdapConfig>
  <Services>
    <Service
      ...
```

```

    </Service>
  <Service
    ...
  <Service
    ...
  </Service>
</Services>
<Canonicalizations>
  ...
</Canonicalizations>
</LdapConfig>

```

2. Add the optional <Tls> protection section as the first element of the configuration. This section is the global default for all configured services. For example:

```

<LdapConfig>
  !-- The default TLS configuration goes here. -->
  <Tls
    LdapClientTlsCACertDir="/etc/ssl/certs"
    LdapClientTlsReqCert="allow"
    LdapClientTlsCACert="/etc/ssl/certs.pem"
    LdapClientTlsCert="/etc/ssl/certs/client.pem"
    LdapClientTlsKey="/etc/ssl/certs/key.pem"
    LdapClientTlsRandFile="/dev/random"
    LdapClientTlsCipherSuite="!LOW"/>

```

**Note:**

If the TLS requirements vary among directory services, you can configure TLS separately for each <Service>, as shown in the next step. Also see [Using TLS with a Directory Server](#). The settings for a service override the global settings.

3. Configure a <Service> element for each directory service. For example:

```

<Service
  Id="svc.div1root"
  LdapServerName="ldap://div1root/ ldap://tdgss/ ldap://wave/"
  LdapBaseFQDN="dc=div1rootdom,dc=div1dev,dc=corp"
  LdapServiceFQDN="cn=div1root,ou=services,dc=div1rootdom,
dc=div1dev,dc=corp"
  LdapSystemFQDN="cn=end2end,cn=tdat,dc=div1rootdom,
dc=div1dev,dc=corp"
  LdapServicePassword="password"
  LdapClientUseTls="no"
  LdapClientMechanism="simple">

```

```

<!-- Overrides to the default TLS configuration go here. -->
  LdapClientTlsReqCert="demand"
  LdapClientTlsCert="/home/mycert"/>
</Service>
<Service
...
</Service>
</Services>

```

Property	Description
Id="svc.div1root"	Uniquely names a directory service
LdapServerName= ... LdapClientMechanism=	The list of LDAP properties that are required for the service, according to site needs.  <b>Note:</b> You only need to include properties with non-default values.
LdapServicePassword="password"	Set this property to the encrypted password created in step 4.
LdapClientTlsReqCert="demand" LdapClientTlsCert="/home/mycert"/>	Optional TLS sub-section that contains values to override the defaults in the main TLS section, for this service only.

- Use the `tdspasswd` tool to generate an encrypted password for the private key file.

```

# tdspasswd -s svc.div1root
Enter New password:
Confirm New password:
ASfb+l7norNgJHZZBufEmRS=

```

where `svc.div1root` is the value specified for the `Id` property in the `<Service>` configuration shown in step 3.

When prompted by the tool, enter a password. The tool generates an encrypted version of the password, for example: `ASfb+l7norNgJHZZBufEmRS=`

- Enter the encrypted version of the password as the value of the `LdapServicePassword` property in the configuration shown in step 3.

## Adding Identity Map and Identity Search Elements to the LdapConfig

You must add at least 1 Identity Map or Identity Search element to the <Canonicalizations> subsection of the <LdapConfig> to assist LDAP in finding users. The system considers the canonicalizations in the order they appear in the configuration file.

1. Create the needed <Identity Map> and <Identity Search> elements. See the topics beginning with [Working with Directory User Identification Options](#).
2. Add the <Identity Map> and <Identity Search> elements to <Canonicalizations> section, for example:

```
<Canonicalizations>
  <IdentitySearch
    Ref="svc.div1root"
    Pattern="(.)@td.teradata.com"
    Base="dc=td,dc=teradata,dc=com"
    Filter="(uid=${1})"
    DatabaseName="div1_${1}"
    BindName="dn:${result}"/>
  <IdentitySearch
    ...
  />
</Canonicalizations>
```

## Completing the <LdapConfig> Configuration Change

After you complete the editing of the TDGSS configuration file to include the <LdapConfig> section, you should test the configuration before committing the configuration on the Teradata Vantage system or Unity server.

1. Verify the configuration is correct:
  - a. Run `tdgsstestcfg` to test the configuration. It launches a test environment in a new shell that contains the updates to the configuration file.

```
/opt/teradata/tdgss/bin/tdgsstestcfg
```

- b. Use the `tdgssauth` utility to test the new <LdapConfig> to verify that users from each service can be authenticated.
  - Specify a directory user and the necessary options using `tdgssauth -u dir_user`. See [Working with tdgssauth](#).
  - Check the output.
  - If the authentication fails, exit the test shell and make the necessary configuration changes and rerun `tdgsstestcfg` and `tdgssauth` until the authentication succeeds.

- c. Exit the test shell:

```
exit
```

2. On the Vantage system (or Unity server), run the run\_tdgssconfig utility to update the TDGSSCONFIG GDO with the new version of the <LdapConfig>.

```
/opt/teradata/tdgss/bin/run_tdgssconfig
```

3. If run\_tdgssconfig indicates that a TPA reset is required, run tpareset.

```
tpareset -f "use updated TDGSSCONFIG GDO"
```

## Using <LdapConfig> with Unity

If you want the same LDAP authentication behavior for database users who log on through Unity and those who log on directly to a database system connected to Unity, you must duplicate <LdapConfig> changes on each Unity server and on all connected databases.

For information about Unity, see *Teradata® Unity™ Installation, Configuration, and Upgrade Guide for Customers*, B035-2523 and *Teradata® Unity™ User Guide*, B035-2520.

# Network Security Policy

You can configure the system to enforce several network security policy options.

## Prerequisites

Security policies must be configured in an LDAPv3-compliant directory. See [Evaluating the System for Directory Management of Users](#).

---

### Note:

You can use `LdapServerName` to configure automatic failover to a second directory containing duplicate security policies (recommended). See [LdapServerName](#).

---

## Implementation Overview

1. Review network security policy options. See [About Network Security Policy](#).
2. Determine whether to use Teradata directory schema extensions or native directory schema to define security policy objects in the directory, and if necessary install Teradata schema extensions. See [Directory Schema Considerations](#).
3. Configure the top level security policy objects in the directory. See [Configuring Top-Level Security Policy Objects](#).
4. Configure the needed security policies:
  - For mechanism policies, see [Configuring a Security Mechanism Policy](#).
  - Review the function of confidentiality and integrity QOP policies. See [About Confidentiality and Integrity QOP Policy](#).
  - Configure `ipNetworks` and `Network Groups` for use in assigning QOP and Options policies. See [Configuring ipNetworks and Network Groups](#).
  - For integrity, see [Configuring an Integrity QOP Policy](#).
  - For confidentiality, see [Configuring a Confidentiality QOP Policy](#).
  - Review the function of options policies and configure any needed options. See [Configuring Options Policies](#).
5. Review the effects of using `RequireConfidentiality` with QOP. See [Requiring Confidentiality](#).
6. Enable policies by configuring security policy attributes in the TDGSS user configuration file. See [Configuring Security Policies in the TdgssUserConfigFile.xml](#).
7. Monitor logs to detect QOP security policy violations and identify violators. See [Monitoring QOP Security Policy](#).

## About Network Security Policy

Teradata Vantage provides the ability to configure the following security policy options:

- Restrict users to logging on with specific authentication mechanisms.
- Enforce the use of, and the strength of encryption algorithms for:
  - Integrity, the checksum used to prevent loss or change of data during transmission
  - Confidentiality (message encryption with a checksum)
- Assign integrity and confidentiality policy by:
  - User (Vantage user or directory principal)
  - Profile
  - IP address
- Configure and assign options policies, by user, profile or IP address:
  - For sessions that are encrypted between clients and Unity, require use of clear text between the Unity server and a co-located database to minimize processing overhead.
  - Specify the users who must log on through Unity rather than directly to the database.

---

### Note:

You can also configure a policy to allow or deny logons by IP address. For details about IP logon restriction, see [Restricting Logons by IP Address](#).

---

## Security Policy and Related Client Settings

You can specify some security policy parameters on a Teradata Vantage™ client.

- All Vantage applications allow you to select the authentication mechanism.
- All Vantage client applications and the ODBC driver allow you to request confidentiality (message encryption). Confidentiality includes integrity checking.
- .NET Data Provider for Teradata allows you to request integrity only, when confidentiality is not used.
- Client applications and drivers do not allow you to specify QOP strength.

## System Processing of Security Policies

When a user logs on to Teradata Vantage, TDGSS checks the <LdapConfig> section of the TdgssUserConfigFile.xml for <Policy> elements. A <Policy> element found in the <LdapConfig> directs TDGSS to a directory object that is the parent of a security policy configuration. TDGSS enforces any applicable policies that it finds in the directory.

**Note:**

If a QOP-related policy or option applies to a session that logs on from a pre-14.10 client, the system does not permit the logon, except when:

- The applicable policy derives from setting the RequireConfidentiality flag to yes. See [Requiring Confidentiality](#).
- The --secpynotsupported logon flag is set. See [Configuring the Gateway to Allow Logons from Older Interfaces or Proxies](#).

The system determines the security policy for a session based upon client settings, such as a client request for confidentiality or integrity, combined with other applicable security policy.

- In the absence of an applicable QOP or option policy, a request for confidentiality or integrity on the client uses the DEFAULT QOP.
- If multiple QOP policies apply to the session, the strongest QOP takes precedence.

## Directory Schema Considerations

You can use either native directory schema or Teradata schema extensions to create Teradata security policy definitions in the directory structure. The schema type (native or Teradata) can vary from one tdatPolicy entry to another, but must be the same within a policy entry.

If you already use directory authorization for Vantage users, the policy schema type does not need to match the existing authorization schema type.

## Using Teradata Schema Extensions to Configure Security Policy

To use Teradata schema extensions to configure security policies, you must install special Teradata policy schema extensions in the directory.

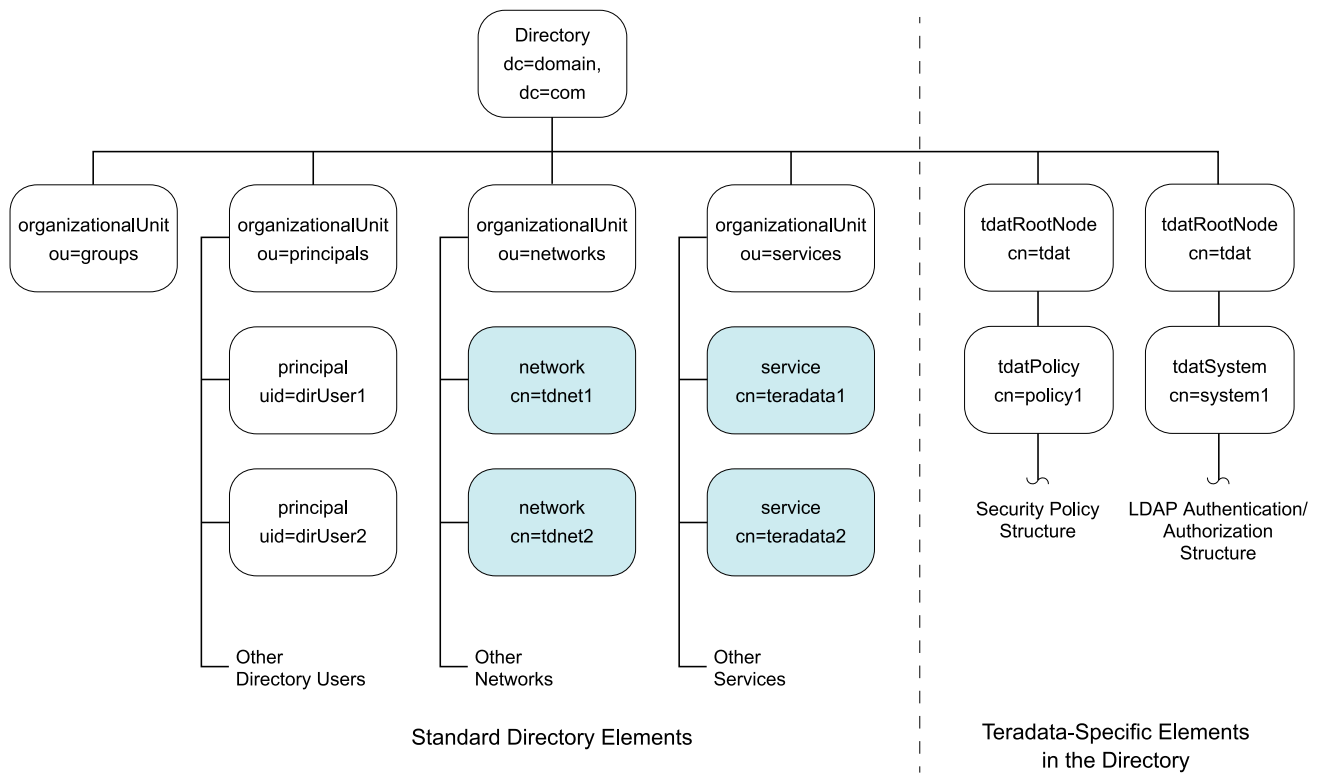
If you use Teradata policy schema to create the policy structure in the directory, you must also install the base Teradata schema extensions (tdat.<dir\_type>.schema) normally used to set up directory authorization, and add the user and profile objects needed to define policy membership, even if you are not using LDAP authorization.

For information on installing Teradata schema extensions for your directory type, see [Installing Teradata Schema Extensions in a Certified Directory](#).

## Standard Directory Entries and Security Policy

When defining security policies in a directory, some of the required directory objects are created within the normal directory structure, as shown in the following diagram.





The following typical LDIF directory structural elements, shown in the preceding diagram are referenced in the security policy examples. Your directory structure may vary.

```
dn: dc=domain1,dc=com
objectClass: organization
objectClass: dcObject
dc: domain
ou: domain.com

dn: ou=networks,dc=domain1,dc=com
objectClass: organizationalUnit
ou: networks

dn: ou=services,dc=domain1,dc=com
objectClass: organizationalUnit
ou: services

dn: ou=principals,dc=domain1,dc=com
objectClass: organizationalUnit
ou: principals

dn: ou=groups,dc=domain1,dc=com
```

```
objectClass: organizationalUnit
ou: groups
```

## Creating a Service Object for Each Teradata Vantage System

Create a service object for each Teradata Vantage system for which the directory contains a security policy using native directory schema. Give the service permission to read the directory, so it can find applicable security policies. Do not allow the service to write to the directory.

```
dn: cn=teradata1,ou=services,dc=domain1,dc=com
objectClass: device
objectClass: simpleSecurityObject
cn: teradata1
userPassword: {SSHA}ReSK+0po4ZTwpi+OSTnKuBZF5+Vab1t7
```

---

### Note:

This example shows an encrypted password, but the password can also appear in plaintext depending on your site security requirements, although plaintext is not recommended.

---

## Creating ipNetwork Objects for Use in Assigning Policies by IP Address

An ipNetwork object defines a range of IP addresses, each of which can be used for assigning:

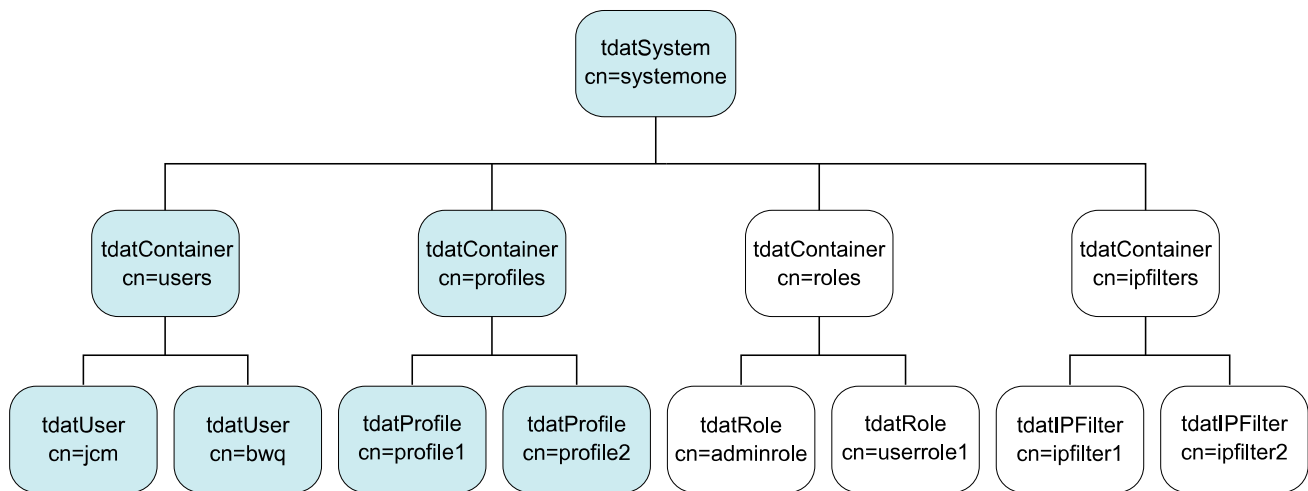
- Confidentiality policies
- Integrity policies
- Options policies

You can create ipNetwork objects in the networks container or other location. See [Configuring ipNetworks and Network Groups](#).

## Using LDAP Directory Objects in Policies

You can assign a security policy to Teradata user and profile objects in the directory.

Some Teradata user and profile objects may already exist in the directory as a result of configuring LDAP authorization, as shown in the following diagram.



Teradata Vantage users and profiles must exist in the directory before you can assign security policies to them.

In a multiple system environment, where users log on through Unity, a single `tdatSystem` object, related authorization structure, and set of security policies applies to all directory users.

In cases where directory users can log on through Unity or directly to the database, Teradata recommends a single authorization structure for all logons.

For information on creating profile and user objects, see [Provisioning Directory Users with Teradata Schema Extensions](#) or [Using Native Directory Schema to Provision Directory Users](#).

## Rules for Specifying Users as Policy Members

You can specify the DN of a `tdatUser` object, or in some cases the DN of a directory principal object, as a member of a policy to apply the policy to the user.

The DN specification requirements depend on how the user is authenticated and authorized, regardless of policy type.

Authentication Mechanism	Member Definition
TD2	The DN must be an existing Teradata user object in the directory with a cn that matches a Teradata Vantage user name.
KRB5 (AuthorizationSupported=no)	The DN must be an existing Teradata user object in the directory with a cn that matches Kerberos domain user name.
LDAP (AuthorizationSupported=no)	The DN must be an existing Teradata user object in the directory with a cn matches the LDAP log on name for the user.
KRB5 or LDAP with (AuthorizationSupported=yes)	Must be either: <ul style="list-style-type: none"> <li>The DN of a Teradata user object</li> <li>The DN of a directory principal</li> </ul> The choice of user object is subject to the following rules:

Authentication Mechanism	Member Definition
	<p>If the directory principal is mapped to a Teradata user object, use the DN of the Teradata user object for the member attribute.</p> <p>If the directory principal is not mapped to a Teradata user object, use the DN of the directory principal for the member attribute.</p>
PROXY	<p>The PROXY mechanism is only used by the Unity server for logging on to connected Vantage systems.</p> <p>If users logging on through Unity are externally authenticated, PROXY must be configured. If PROXY is configured, Unity also uses the PROXY mechanism for TD2 sessions.</p> <p>If PROXY is configured, create a PROXY mechanism policy and assign policy membership to the Unity user for each server, to ensure the security of Unity connections to the database,</p> <ol style="list-style-type: none"> <li>1. Define the Unity user as part of initial setup of each Unity server. See the Teradata Unity documentation.</li> <li>2. Re-specify the Unity user and password on each Unity server when configuring the certificate and private key for use with externally authenticated Vantage users. For information about Unity, see <i>Teradata® Unity™ Installation, Configuration, and Upgrade Guide for Customers</i>, B035-2523 and <i>Teradata® Unity™ User Guide</i>, B035-2520.</li> <li>3. Define each Unity user as a Teradata user object in the directory, as shown in the diagram in <a href="#">Using LDAP Directory Objects in Policies</a>.</li> <li>4. Assign PROXY policy membership to the Unity user for each Unity server in the directory. For instructions on the syntax used to assign membership to users, see the topics for each policy type beginning with <a href="#">Configuring a Security Mechanism Policy</a>.</li> </ol> <p><b>Note:</b></p> <p>Do not assign a PROXY policy to any other user.</p>
JWT	JWT allows a user that has been authenticated to the Identity Provider to do a single sign-on to establish a session with Vantage.

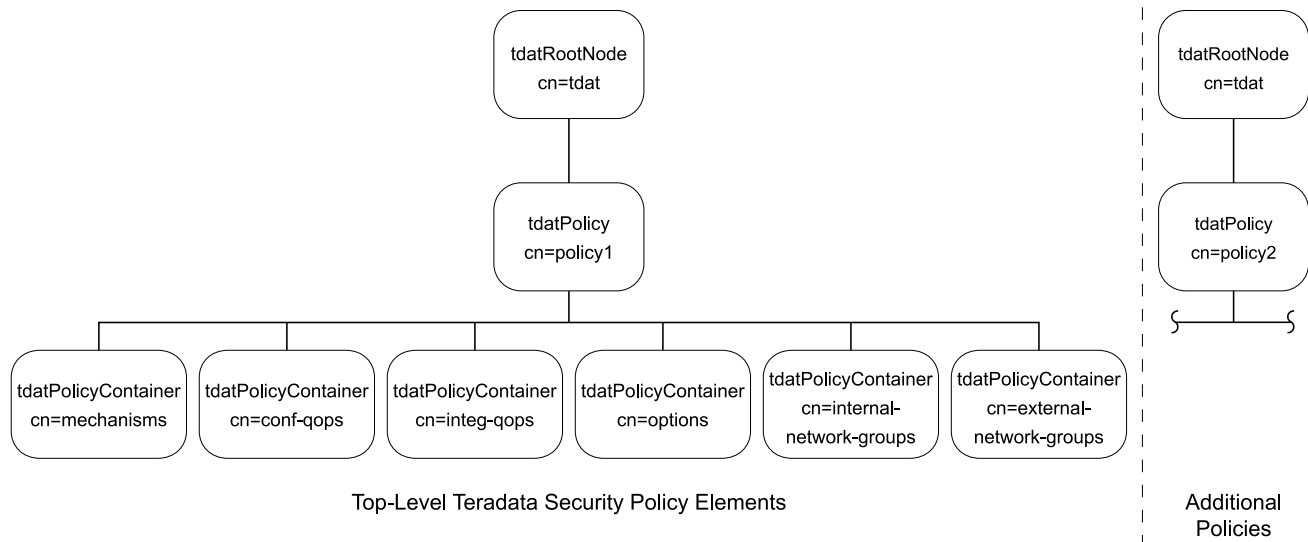
## Specifying Profiles as Policy Members

When assigning policy membership to a profile, consider if the user for the session is a:

- Teradata user object in the directory, the session uses the profile assigned to the corresponding Teradata Vantage user.
- Directory principal mapped to a Teradata user object, the session inherits the profile assigned to the corresponding Vantage user.
- Directory principal mapped to a Teradata profile, the session uses the corresponding Vantage profile even if the principal is also mapped to a Vantage user.

## Configuring Top-Level Security Policy Objects

The following diagram shows the top level security policy structure and the container object for each policy type. Detailed diagrams and instructions for creating specific policies are shown in the topics beginning with [Configuring a Security Mechanism Policy](#).



If a directory serves multiple Teradata Vantage systems with differing policy requirements, you can configure multiple policy structures, for example, a local policy and a global policy. The `TdgssUserConfigFile.xml` on each system must point to the related `tdatPolicy` object. See [Configuring Security Policies in the TdgssUserConfigFile.xml](#).

If multiple databases served by a single directory are accessed through Unity, all databases should be subject to the same policy structure.

## Creating the tdatRootNode Object

The `tdatRootNode` object is the top level Teradata object in a directory. If an LDAP authentication and authorization structure for Teradata Vantage users already exists in the directory, the structure contains a `tdatRootNode`.

A security policy structure also requires a `tdatRootNode` object. You can configure a `RootNode` for each policy structure separately from the `tdatRootNode` for the LDAP authorization structure to allow for separate portability of each structure, but it is not required.

## Using Teradata Schema Extensions to Create a tdatRootNode

```

dn: cn=tdatP,dc=domain1,dc=com
objectClass: tdatRootNode
cn: tdatP
  
```

## Using Native Directory Schema to Create a RootNode Object

```
dn: ou=tdatP,dc=domain1,dc=com
objectClass: organizationalUnit
ou: tdatP
```

## Creating the tdatPolicy Object

The top level security policy object is the tdatPolicyContainer object. All other policy objects are children of the top level policy object, including the individual tdatPolicyContainer object for each policy type.

## Using Teradata Schema Extensions to Create the Top-Level Policy Object

```
dn: cn=policy1,cn=tdatP,dc=domain1,dc=com
objectClass: tdatPolicy
cn: policy1
```

## Using Native Directory Schema to Create the Top-Level Policy Object

```
dn: cn=policy1,ou=tdatP,dc=domain1,dc=com
objectClass: organizationalUnit
cn: policy1
```

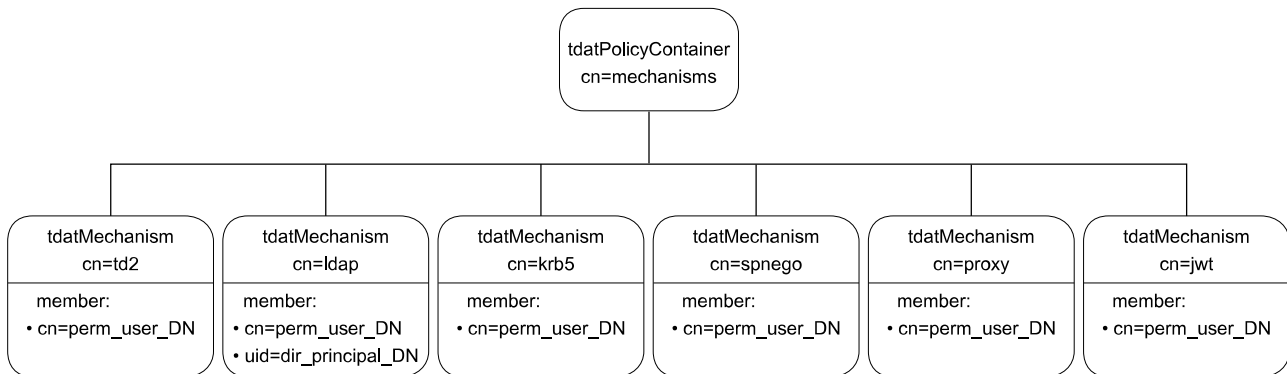
## Configuring a Security Mechanism Policy

### About Security Mechanism Policies

You can create security mechanism policies to restrict the mechanisms available to users when they log on to the database.

Users that are members of at least one policy can only use mechanisms in which they have membership. Users that are not members of any security mechanism policy are not restricted in their use of security mechanisms.

The TDNEGO mechanism itself is not restricted by security mechanism policy, but the mechanisms it selects may be restricted. Users do not have to be permitted to use TDNEGO, but they do have to be permitted to use mechanisms that TDNEGO might negotiate for them, so users need to be members of the mechanisms they want TDNEGO to pick for them. For example, if a user's mechanism policy permits KRB5 and LDAP, then TDNEGO will restrict the user to those mechanisms.



To create a mechanism policy:

1. Create the mechanisms container. See [Creating the Mechanisms Container](#).
2. Create mechanism objects in the mechanism container. See [Creating Mechanism Objects in the Mechanisms Container](#).
3. Specify the users that are members of the mechanism. See [Adding Member Users to a Mechanism Policy](#).

## Creating the Mechanisms Container

Create the mechanisms container as a child of the tdatPolicyContainer object to contain individual security mechanism objects.

### Using Teradata Schema Extensions to Create a Mechanism Container

```
dn: cn=mechanisms,cn=policy1,cn=tdatrootP,dc=domain1,dc=com
objectClass: tdatPolicyContainer
cn: mechanisms
```

### Using Native Directory Schema to Create a Mechanism Container

```
dn: ou=mechanisms,ou=policy1,ou=tdatrootP,dc=domain1,dc=com
objectClass: organizationalUnit
ou: mechanisms
```

## Creating Mechanism Objects in the Mechanisms Container

Create mechanism objects in the mechanisms container as needed for mechanism restrictions.

**Note:**

The SPNEGO mechanism does not appear in the example directory entries that follow because it is not required. SPNEGO is a negotiated mechanism that can only negotiate KRB5 (Kerberos authentication), which does appear in the examples. SPNEGO policy is completely defined by KRB5 policy.

## Using Teradata Schema Extensions to Create Mechanism Objects

```
dn: cn=td2,cn=mechanisms,cn=policy1,cn=tdatrootP,dc=domain1,dc=com
objectClass: tdatMechanism
cn: td2

dn: cn=krb5,cn=mechanisms,cn=policy1,cn=tdatrootP,dc=domain1,dc=com
objectClass: tdatMechanism
cn: krb5

dn: cn=ldap,cn=mechanisms,cn=policy1,cn=tdatrootP,dc=domain1,dc=com
objectClass: tdatMechanism
cn: ldap

dn: cn=proxy,cn=mechanisms,cn=policy1,cn=tdatrootP,dc=domain1,dc=com
objectClass: tdatMechanism
cn: proxy
```

## Using Native Directory Schema to Create Mechanism Objects

```
dn: cn=td2,ou=mechanisms,ou=policy1,ou=tdatrootP,dc=domain,dc=com
objectClass: groupOfNames
cn: td2
member: ou=mechanisms,ou=policy1,ou=tdatrootP,dc=domain,dc=com

dn: cn=krb5,ou=mechanisms,ou=systems,ou=tdatrootP,dc=domain,dc=com
objectClass: groupOfNames
cn: krb5
member: ou=mechanisms,ou=policy1,ou=tdatrootP,dc=domain,dc=com

dn: cn=ldap,ou=mechanisms,ou=policy1,ou=tdatrootP,dc=domain,dc=com
objectClass: groupOfNames
cn: ldap
member: ou=mechanisms,ou=policy1,ou=tdatrootP,dc=domain1,dc=com
```



```
dn: cn=proxy,ou=mechanisms,ou=policy1,ou=tdatrootP,dc=domain1,dc=com
objectClass: groupOfNames
cn: proxy
member: ou=mechanisms,ou=policy1,ou=tdatrootP,dc=domain1,dc=com
```

**Note:**

The member attribute is required for any object with a GroupOfNames class. The preceding examples use member: ou=mechanisms to allow the mechanism object to exist in the directory prior to being populated with individual member users.

## Adding Member Users to a Mechanism Policy

You can use the member attribute to assign users to a mechanism object.

If a user is not a member of any mechanism policy, the user has no policy-based mechanism usage restrictions, and can use any mechanism that is not disabled in the TDGSS configuration.

If a user is a member of one or more mechanisms, the user can log on only with a mechanism of which he or she is a member.

For a list of rules governing the specification of users as policy members, see [Rules for Specifying Users as Policy Members](#).

**Note:**

Users accessing the database through pooled sessions are subject to the security policies assigned to the application logon user, rather than the policies assigned to them as individuals.

## Using Teradata Schema Extensions to Add Users to a Mechanism

You can add tdatUser objects in the directory as members of a mechanism.

```
dn: cn=ldap,cn=mechanisms,cn=policy1,cn=tdatrootP,dc=domain1,dc=com
changetype: modify
add: member
member: cn=jcm,cn=users,cn=system1,cn=tdatrootA,dc=domain1,dc=com
```

**Note:**

When addition of a directory principal member is required, enter:

```
member: uid=dirUser1,ou=principals,dc=domain1,dc=com
```

## Using Native Directory Schema to Add Users to a Mechanism

You can add users that already exist as native directory principal objects to be members of a mechanism. For example:

```
dn: cn=ldap,ou=mechanisms,ou=policy1,ou=tdatrootP,dc=domain1,dc=com
changetype: modify
add: member
member: cn=jcm,ou=users,ou=system1,ou=tdatrootA,dc=domain1,dc=com
```

---

### Note:

When addition of a directory principal member is required, use the form:

```
member: uid=dirUser1,ou=principals,dc=domain1,dc=com
```

---

## About Confidentiality and Integrity QOP Policy

The quality of protection (QOP) defines the strength of the encryption algorithms the system uses when transmitting messages between Teradata Vantage and its clients.

There are 2 types of policy subject to Quality of Protection (QOP):

- An integrity QOP determines the strength of the algorithm the system uses for calculating the checksum that guarantees message integrity.
- A confidentiality QOP determines the strength of the algorithm for encrypting a message exchange between a client and Vantage. In the absence of a confidentiality QOP policy, client requests for confidentiality use the DEFAULT QOP. See [Encryption](#).

You can assign confidentiality and integrity policies by:

- Database user name or directory user name
- Database profile
- Client IP address

---

### Note:

Users who access the database through a middle-tier application that uses pooled sessions are subject to the security policies assigned to the application logon user, rather than the policies assigned to them as individuals.

---

You can also enforce use of the DEFAULT confidentiality QOP by host group ID. See [Requiring Confidentiality](#).

---

**Note:**

Java clients do not support encryption stronger than AES-128 without installation of a special security policy package. See [QOP Configuration Change Guidelines](#).

---

## System Processing of Confidentiality and Integrity QOP Policies

The system enforces confidentiality or integrity policy as a result of a:

- Client request for confidentiality or integrity
- QOP policy assigned to the user, IP address, profile, or host group for the session

When only a client request for confidentiality is in effect, the system uses the DEFAULT QOP.

If a session is subject to multiple QOP security policies, the system determines the session security policy and QOP according to the following hierarchy, where the higher number and letter always takes precedence:

1. No applicable policy

---

**Note:**

This is the only policy condition that allows a logon from a pre-14.10 client.

---

2. Clear text

---

**Note:**

This policy condition occurs only when the has-policy Option applies to a session, but no other client request or policy for confidentiality or integrity applies.

---

3. Integrity

- a. Default QOP
- b. Low QOP
- c. Medium QOP
- d. High QOP

4. Confidentiality

- a. Default QOP
- b. Low QOP
- c. Medium QOP
- d. High QOP

## Mechanism Effects

QOP enforcement varies depending on the authentication mechanism used for the session, as shown in the following table.

Mechanism	Enforcement Considerations
All mechanisms (without PROXY connection)	If the client does not specify confidentiality or integrity for a session, but a confidentiality or integrity QOP policy applies to the session, the system uses the applicable confidentiality or integrity. Involvement of specific security mechanisms can affect how the policy is enforced.
All mechanisms (with PROXY connection)	When the a session passes through a Unity server where the PROXY connection is configured, QOP applies as follows: <ul style="list-style-type: none"> <li>• If a QOP policy does not apply to the Unity user, the system uses the same QOP for transmissions between Unity and the destination database as for the message transmissions between the client and Unity.</li> <li>• If a QOP policy applies to the Unity user, the system uses the Unity user QOP on message transmissions between the Unity server and the destination database.</li> </ul>
TD2, LDAP, and JWT	If the client specifies confidentiality or integrity, the system defaults to the DEFAULT QOP. If an applicable QOP policy requires a stronger QOP than the default, the system uses the stronger QOP.
Kerberos	If the client specifies, or applicable policy requires, confidentiality or integrity, the system uses it. However, the QOP is determined by Kerberos, regardless of the default QOP or the QOP specified in the applicable policy.
SPNEGO	

## Configuring ipNetworks and Network Groups

You can configure ipNetwork objects and Network Group objects for use in assigning security policy by client IP address.

To configure the ipNetwork and Network Group objects necessary to use in assigning policy:

1. Make sure you understand the function of internal and external network groups. See [About Using ipNetworks and Network Groups to Assign Policy](#).
2. Determine whether your directory supports the use of the ipNetwork object. See [Directory Schema Requirements for Using ipNetwork Objects](#).
3. Create the required ipNetwork objects. See [Creating ipNetwork Objects](#).
4. Create the Network Group containers. See [Creating Network Group Containers](#).
5. Create network group objects to use in assigning QOP and Options policies. See:
  - [Creating Internal Network Groups](#).
  - [Creating External Network Groups](#).

6. Add ipNetwork objects as members of internal and external Network Group objects to define the IP address ranges controlled by the group objects. See [Adding ipNetworks to a Network Group](#).
7. Optionally remove ipNetwork objects from Network Groups when needed. See [Removing ipNetworks from a Network Group](#).

## About Using ipNetworks and Network Groups to Assign Policy

An ipNetwork object defines an IP address range that can be used to define security policy assignments. However, ipNetwork objects do not link directly to a security policy. Instead, you must create internal and external network group objects, assign network group membership for ipNetwork objects that define affected IP address ranges, and then assign security policy membership to the network groups to link the IP address ranges to policies.

The following policy types can be assigned by IP address:

- Quality of Protection (QOP) policies for integrity and confidentiality.
- Options policies (has-policy and no-direct-logon)

## Comparing the Function of Internal and External Network Groups

Network Group Type	Description
Internal Network Group	Contains one or more ipNetwork objects that define a range of IP addresses which are subject to any security policies in which the network group is a member.
External Network Group	Contains one or more ipNetwork objects that define a range of IP addresses which are exempted from any security policies in which the network group is a member. IP addresses defined in ipNetwork objects that are not members of an External Network Group are subject to any policy in which the External Network Group is a member.

## Rules for Using Network Groups to Define Policy Effects

Each ipNetwork object can appear in as many network groups as needed.

Each ipNetwork object functions independently, so overlap of IP address ranges among several ipNetwork objects is allowed.

You can create as many ipNetwork and network group objects as is required to represent the IP address ranges you want to use for assigning security policies.

### Note:

To avoid detailed searches of the directory that would be required to verify the effects of IP based policy assignments, the database holds policy-related IP information in the network cache. The network cache is not updated dynamically.

If you make changes to ipNetwork or Network Group objects in the directory you must perform a TPA reset to force the database to reload the network cache and make the revised information available for policy enforcement.

## Directory Schema Requirements for Using ipNetwork Objects

Most directory types certified for use with TDGSS contain schema that conforms to IETF RFC 2307. This RFC defines a standard data storage entry for naming services, including the ipNetwork entry and its required attributes.

The ADAM and AD LDS directories do not contain schema to support an ipNetwork entry. If you plan to assign policy by IP address and your directory is ADAM or AD LDS, you must install one of the following to enable use of ipNetwork entries:

- An IETF RFC 2307-compatible schema for the ipNetwork object from an outside source.
- The ipNetwork schema extension provided by Teradata, ipnetwork.adam.schema, which is based on the ipNetwork object schema found in Active Directory. The Teradata-provided schema appears in the tdgss/etc directory.

## Installing the ipNetwork Schema Extension on ADAM and AD LDS

If your directory is ADAM or AD LDS and you plan to use IP address ranges to enforce security policy, install the Teradata ipnetwork.adam.schema extension. See [Installing Teradata Schema Extensions in a Certified Directory](#).

## Creating ipNetwork Objects

Create ipNetwork objects that specify an IP address range that is consistent with how you want to apply a security policy.

```
dn: cn=homenet,ou=networks,dc=domain1,dc=com
objectClass: ipNetwork
cn: homenet
ipNetworkNumber: 141.206.0.0
ipNetmaskNumber: 255.255.0.0

dn: cn=remotenet1,ou=networks,dc=domain1,dc=com
objectClass: ipNetwork
cn: remotenet1
ipNetworkNumber: 153.64.0.0
ipNetmaskNumber: 255.255.0.0

dn: cn=remotenet2,ou=networks,dc=domain1,dc=com
objectClass: ipNetwork
```

```
cn: remotenet2
ipNetworkNumber: 141.24.0.0
ipNetmaskNumber: 141.40.0.0
```

## About Network Groups

Network groups are container objects that contain one or more member ipNetwork objects, each of which represents a range of IP addresses.

Although a network group object can be a member of multiple policies, you may find that creating separate network group objects for each policy provides needed flexibility in case IP address coverage requirements vary over time among policies.

## Creating Network Group Containers

If you plan to manage security policy by IP address you must create 2 network group containers using the required common names.

The container name determines the function of network group objects in the container.

Container Type	Required Common Name	Function
Internal network groups container	cn=internal-network-groups	Internal network groups contain ipNetwork objects that specify the IP addresses included in a policy of which the group is a member.
External network groups container	cn=external-network-groups	External network groups contain ipNetwork objects that specify IP addresses excluded from a policy in which the group is a member.

## Using Teradata Schema to Create a Network Group Container

Use the following syntax and specify the container name to create a network groups container, for example, an internal network groups container. The format of the external network version is similar.

```
dn: cn=internal-network-groups,cn=policy1,cn=tdatrootP,dc=domain1,
dc=com
objectClass: tdatPolicyContainer
cn: internal-network-groups
```

## Using Native Directory Schema to Create a Network Group Container

Use the following syntax and specify the container name to create a network groups container, for example, an external network groups container. The internal network version is similar.

```
dn: cn=external-network-groups,ou=policy1,ou=tdatrootP,dc=domain1,
dc=com
objectClass: tdatPolicyContainer
cn: external-network-groups
```

## Creating Internal Network Groups

You can create internal network groups only in the internal network group container. There is no limit on the number of internal network groups.

Internal groups contain ipNetwork objects that define the IP addresses explicitly effected by the associated policy.

## Using Teradata Schema Extensions to Create an Internal Network Group

```
dn: cn=behind-the-firewall,cn=internal-network-groups,cn=policy1,
cn=tdatrootP,dc=domain1,dc=com
objectClass: tdatInternalNetworkGroup
cn: behind-the-firewall
```

## Using Native Directory Schema to Create an Internal Network Group

```
dn: cn=behind-the-firewall,ou=internal-network-groups,ou=policy1,
ou=tdatroot,dc=domain1,dc=com
objectClass: groupOfNames
cn: behind-the-firewall
member: ou=internal-network-groups,ou=policy1,ou=tdatrootP,dc=domain1,
dc=com
```

### Note:

The preceding example includes the definition of a member attribute, which points to the internal network groups container, only because member is a required attribute of groupOfNames and it must have at least one value.

The DN of the internal network group container is not used but it is recommended because use of the DN of the parent cannot result in a “group loop.”

## Creating External Network Groups

You can create external network groups in the external network group container.



## Using Teradata Schema Extensions to Create an External Network Group

```
dn: cn=outside-the-firewall,cn=external-network-groups,cn=policy1,
   cn=tdatrootP,dc=domain1,dc=com
objectClass: tdatExternalNetworkGroup
cn: outside-the-firewall
```

## Using Native Directory Schema to Create an External Network Group

```
dn: cn=outside-the-firewall,ou=external-network-groups,ou=policy1,
   ou=tdatrootP,dc=domain1,dc=com
objectClass: groupOfNames
cn: outside-the-firewall
member: ou=external-network-groups,ou=policy1,ou=tdatrootP,dc=domain1,
       dc=com
```

### Note:

The preceding example includes the definition of a member attribute, which points to the external network groups container, only because member is a required attribute of groupOfNames and it must have at least one value.

The network group container DN is not used in any searches, but it is recommended because use of the DN of the parent cannot result in a “group loop.”

## Adding ipNetworks to a Network Group

A network group can contain one or more ipNetwork objects that each define a range of IP addresses.

Adding ipNetwork objects to:

- An internal network group applies any policy of which the group is a member to the IP addresses specified in the contained ipNetwork objects.
- An external network group exempts the IP addresses specified in the contained ipNetwork objects from any policy of which the group is a member.

### Note:

Changes to the network configuration in the directory service, including changes to the internal and external network groups, require a TPA reset to force the database to reload the network cache.

## Using Teradata Schema Extensions to Add ipNetwork Objects to an External Network Group

```
dn: cn=behind-the-firewall,cn=internal-network-groups,cn=policy1,
   cn=tdatrootP,dc=domain1,dc=com
changetype: modify
add: member
member: cn=remotenet1,ou=networks,dc=domain1,dc=com
member: cn=remotenet2,ou=networks,dc=domain1,dc=com
-
```

## Using Native Directory Schema to Add ipNetwork Objects to an External Network Group

```
dn: cn=behind-the-firewall,ou=internal-network-groups,ou=policy1,
   ou=tdatrootP,dc=domain1,dc=com
changetype: modify
add: member
member: cn=remotenet1,ou=networks,dc=domain1,dc=com
member: cn=remotenet2,ou=networks,dc=domain1,dc=com
-
```

## Removing ipNetworks from a Network Group

If an ipNetwork should no longer be a member of a network group, you can remove the ipNetwork object.

### Note:

Changes to the network configuration in the directory service, including changes to the internal and external network groups and their member ipNetwork objects, do not take effect until you perform a TPA reset to force the database to reload the network cache.

## Using Teradata Schema Extensions to Remove ipNetwork Objects from an External Network Group

```
dn: cn=behind-the-firewall,cn=internal-network-groups,cn=policy1,
   cn=tdatrootP,dc=domain1,dc=com
changetype: modify
remove: member
```

```
member: cn=remotenet1,cn=networks,dc=domain1,dc=com
-
```

## Using Native Directory Schema to Remove ipNetwork Objects from an External Network Group

```
dn: cn=behind-the-firewall,ou=internal-network-groups,ou=policy1,
   ou=tdatrootP,dc=domain1,dc=com
changetype: modify
remove: member
member: cn=remotenet1,ou=networks,dc=domain1,dc=com
-
```

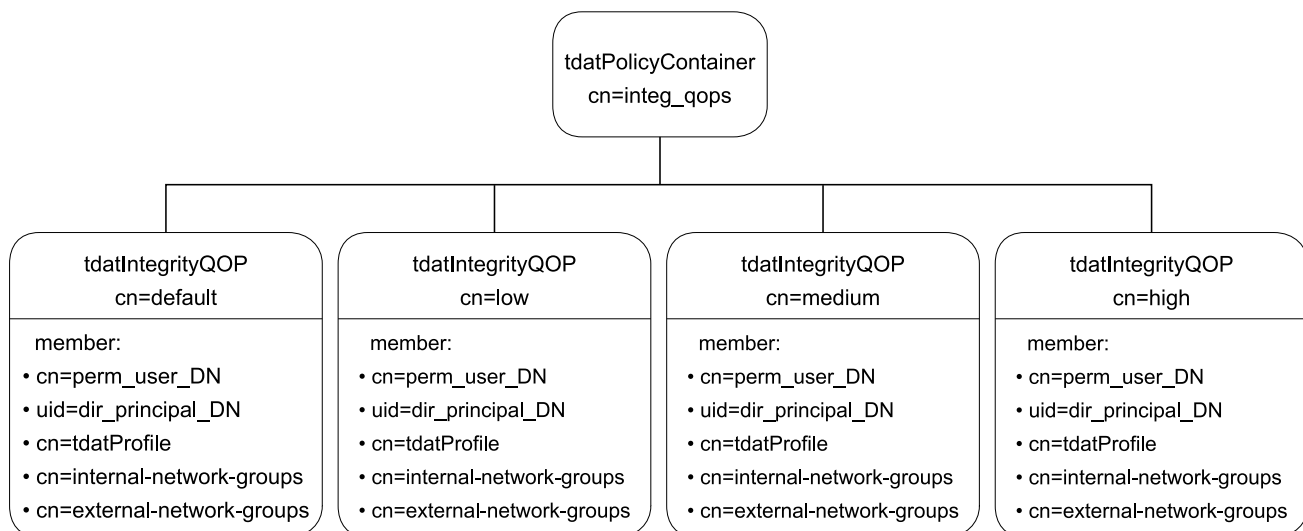
## Configuring an Integrity QOP Policy

Integrity is an automated checksum routine that the system uses to guarantee that data is not altered during transmission between a Teradata Vantage system and its clients.

You can create integrity policies that require use of a specific hash algorithm for the checksum routine, and apply the policies by user, profile, or network group.

### Note:

If a session subject to an Integrity or Confidentiality QOP uses the Kerberos authentication mechanism (which does not support QOP policy), the system enforces the use of integrity, but it ignores the QOP algorithm specified in the policy and uses the algorithm provided by Kerberos.



To configure an integrity QOP policy:

1. Examine the TdgssUserConfigFile.xml and make sure that the QOP entries are enabled and set according to your requirements. See [Working with Quality of Protection Options](#).
2. Create the integrity QOP container. See [Creating the integ-qops Container](#).
3. Create the needed integrity QOP objects. See [Creating Integrity QOP Objects in the integ-qops Container](#).
4. Add members to each integrity QOP to apply QOP policy. [Assigning Members to an Integrity QOP Policy](#).
5. Optionally remove members from an integrity QOP. See [Removing Members from an Integrity QOP](#).

## Prerequisites

QOP policies are based on the QOP configurations that exist in the TdgssUserConfigFile.xml on each Teradata Vantage system. Make sure you understand QOP concepts and functions and perform any needed QOP configuration tasks in the TdgssUserConfigFile.xml before you attempt to configure a policy that requires an integrity QOP. For information, see [Working with Quality of Protection Options](#).

## Creating the integ-qops Container

The QOP container is required to contain individual integrity QOP policy objects.

### Using Teradata Schema Extensions

```
dn: cn=integ-qops,cn=policy1,cn=tdatrootP,dc=domain1,dc=com
objectClass: tdatPolicyContainer
cn: integ-qops
```

### Using Native Directory Schema

```
dn: cn=integ-qops,ou=policy1,ou=tdatrootP,dc=domain1,dc=com
objectClass: groupOfNames
cn: integ-qops
```

## Creating Integrity QOP Objects in the integ-qops Container

Create a QOP object for each of the QOP strengths defined in the TdgssUserConfigFile.xml (default, low, medium, and high), as a child of the integ-qops container.

## Using Teradata Schema Extensions

```
dn: cn=default,cn=integ-qops,cn=policy1,cn=tdatrootP,dc=domain1,dc=com
objectClass: tdatIntegrityQoP
cn: default
```

```
dn: cn=low,cn=integ-qops,cn=policy1,cn=tdatrootP,dc=domain1,dc=com
objectClass: tdatIntegrityQoP
cn: low
```

```
dn: cn=medium,cn=integ-qops,cn=policy1,cn=tdatrootP,dc=domain1,dc=com
objectClass: tdatIntegrityQoP
cn: medium
```

```
dn: cn=high,cn=integ-qops,cn=policy1,cn=tdatrootP,dc=domain1,dc=com
objectClass: tdatIntegrityQoP
cn: high
```

## Using Native Directory Schema

```
dn: cn=default,ou=integ-qops,ou=system,ou=tdatrootP,dc=domain1,dc=com
objectClass: groupOfNames
cn: default
member: ou=integ-qops,ou=policy1,ou=tdatrootP,dc=domain1,dc=com
```

```
dn: cn=low,ou=integ-qops,ou=policy1,ou=tdatrootP,dc=domain1,dc=com
objectClass: groupOfNames
cn: low
member: ou=integ-qops,ou=policy1,ou=tdatrootP,dc=domain1,dc=com
```

```
dn: cn=medium,ou=integ-qops,ou=policy1,ou=tdatrootP,dc=domain1,dc=com
objectClass: groupOfNames
cn: medium
member: ou=integ-qops,ou=policy1,ou=tdatrootP,dc=domain1,dc=com
```

```
dn: cn=high,ou=integ-qops,ou=policy1,ou=tdatrootP,dc=domain1,dc=com
objectClass: groupOfNames
cn: high
member: ou=integ-qops,ou=policy1,ou=tdatrootP,dc=domain1,dc=com
```

## Assigning Members to an Integrity QOP Policy

You can assign an integrity QOP membership by Vantage user, directory user, Vantage profile, and internal or external network group.

If a user is a member of more than one integrity policy, including those included as part of a confidentiality policy, the strongest policy takes precedence.

The following examples show how to assign membership in a QOP using the DN of an existing tdatUser or directory principal object.

---

### Note:

For a list of rules governing the specification of users as policy members, see [Rules for Specifying Users as Policy Members](#).

---

## Applying Integrity QOP Policy to a Teradata Vantage User

Specify the DN of a tdatUser object that exists in the directory.

Using Teradata schema extensions:

```
dn: cn=high,cn=integ-qops,cn=policy1,cn=tdatrootP,dc=domain1,dc=com
changetype: modify
add: member
member: cn=xyz,cn=users,cn=system1,cn=tdatrootA,dc=domain1,dc=com
-
```

Using native directory schema:

```
dn: cn=high,ou=integ-qops,ou=policy1,ou=tdatrootP,dc=domain1,dc=com
changetype: modify
add: member
member: cn=xyz,ou=users,ou=system1,ou=tdatrootA,dc=domain1,dc=com
-
```

## Applying Integrity QOP Policy to a Directory User

Specify the DN of a directory principal for LDAP users that are not mapped to a tdatUser object.

---

### Note:

You can specify policy membership for directory principals only if AuthorizationSupported=no for both KRB5 and LDAP.

---

Using Teradata schema extensions:

```
dn: cn=default,cn=integ-qops,cn=policy1,cn=tdatrootP,dc=domain1,dc=com
changetype: modify
add: member
member: uid=dirUser1,ou=principals,dc=domain1,dc=com
-
```

Using native directory schema:

```
dn: cn=default,ou=integ-qops,ou=policy1,ou=tdatrootP,dc=domain1,dc=com
changetype: modify
add: member
member: uid=dirUser1,ou=principals,dc=domain1,dc=com
-
```

## Applying Integrity QOP Policy to a Teradata Vantage Profile

When you configure integrity QOP policy membership by Vantage profile, the corresponding tdatProfile object must already exist in the directory.

---

### Note:

When users log on to the Vantage through an application that uses pooled sessions, the users are subject to the profile for the application user, not profiles in which the users have membership or to which they are mapped.

---

Using Teradata schema extensions:

```
dn: cn=low,cn=integ-qops,cn=policy1,cn=tdatrootP,dc=domain1,dc=com
changetype: modify
add: member
member: cn=profile1,cn=profiles,cn=system1,cn=tdatrootA,dc=domain1,dc=com
-
```

Using native directory schema:

```
dn: cn=low,cn=integ-qops,cn=policy1,cn=tdatrootP,dc=domain1,dc=com
changetype: modify
add: member
member: cn=profile2,ou=profiles,ou=system1,ou=tdatroot,dc=domain1,dc=com
-
```

## Applying Integrity QOP Policy to a Network Group

The examples below show configurations for internal network groups. Configuration of external groups is similar.

For an explanation of internal and external network group function, see [Creating Network Group Containers](#).

Using Teradata schema extensions:

```
dn: cn=medium,cn=integ-qops,cn=policy1,cn=tdatrootP,dc=domain1,dc=com
changetype: modify
add: member
member: cn=behind-the-firewall,cn=internal-network-groups,cn=policy1,
       cn=tdatrootP,dc=domain1,dc=com
-
```

Using native directory schema:

```
dn: cn=medium,ou=integ-qops,ou=policy1,ou=tdatrootP,dc=domain1,dc=com
changetype: modify
add: member
member: cn=behind-the-firewall,ou=internal-network-groups,ou=policy1,
       ou=tdatrootP,dc=domain1,dc=com
-
```

## Removing Members from an Integrity QOP

You can remove members from an integrity QOP. Just substitute the remove: attribute for the add: attribute shown in [Assigning Members to an Integrity QOP Policy](#), and remove the unneeded members. For example:

```
dn: cn=high,cn=integ-qops,cn=policy1,cn=tdatrootP,dc=domain1,dc=com
changetype: modify
remove: member
member: cn=jcm,cn=users,cn=systemone,cn=tdatrootA,dc=domain1,dc=com
-
```

## Configuring a Confidentiality QOP Policy

You can configure confidentiality policies to enforce confidentiality, at a specified algorithm strength, for all sessions regardless of whether they request encryption.

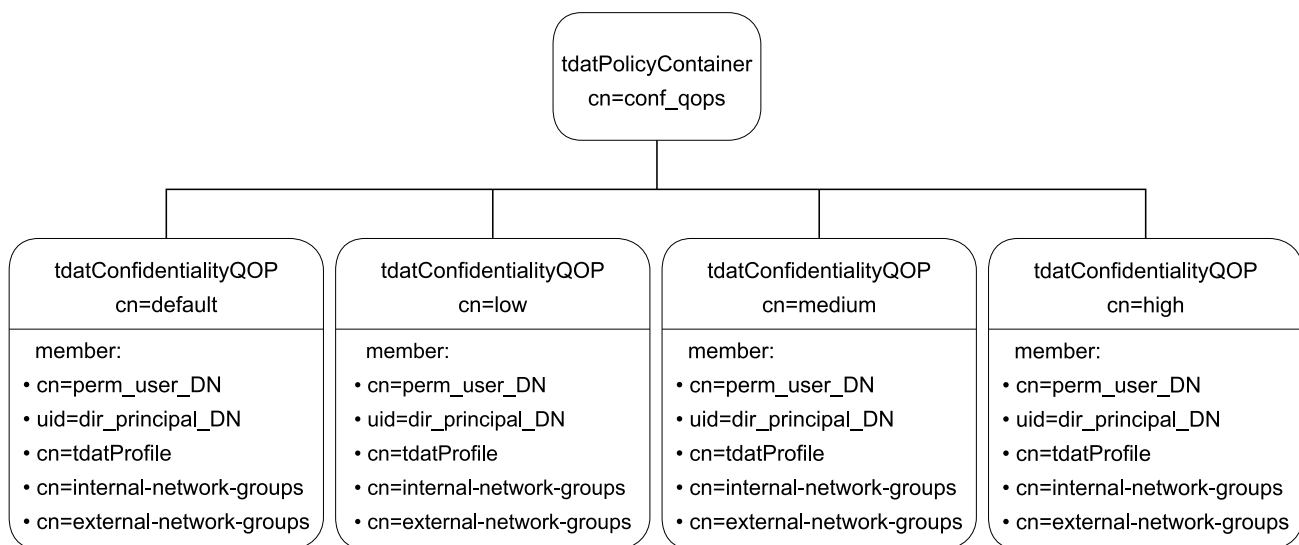


Confidentiality policy only applies to the TD2, KRB5, LDAP, and PROXY mechanisms. See the comparison table in [System Processing of Confidentiality and Integrity QOP Policies](#).

**Note:**

If a session subject to a Confidentiality QOP uses the Kerberos authentication mechanism (which does not support QOP policy), the system enforces the use of confidentiality, but ignores the QOP algorithm specified in the policy and uses the algorithm provided by Kerberos.

Confidentiality policies are based on the configuration of the LOW, MEDIUM, and HIGH QOP entries in the TdgssUserConfigFile.xml. You must enable these QOP entries in the configuration file before configuring a confidentiality policy. For information, see [Working with Quality of Protection Options](#).



To configure a confidentiality QOP policy:

1. Examine the TdgssUserConfigFile.xml and make sure that the QOP entries are enabled and set according to your requirements. See [Working with Quality of Protection Options](#).
2. Create the confidentiality QOP container. See [Creating the conf-qops Container](#).
3. Create the needed confidentiality QOP objects. See [Creating Confidentiality QOP Objects in the Confidentiality QOP Container](#).
4. Add members to each confidentiality QOP to define QOP effects. [Adding Members to a Confidentiality QOP to Require QOP Usage](#).

**Note:**

You can also apply the default confidentiality QOP by host group. See [Requiring Confidentiality](#).

5. Optionally remove members from a confidentiality QOP to remove QOP effects. See [Removing Members from a Confidentiality QOP](#).

## Prerequisites

QOP policies are based on the QOP configurations that exist in the TdgssUserConfigFile.xml on each Teradata Vantage system. Make sure you understand QOP concepts and functions and perform any needed QOP configuration tasks in the TdgssUserConfigFile.xml before you attempt to configure a policy that uses QOP.

For information, see [Working with Quality of Protection Options](#).

## Creating the conf-qops Container

Create the confidentiality QOPs container to contain individual QOP policies. Create the confidentiality QOP container as a child of the tdatPolicy entry. The container name must be cn=conf-qops, because that is where TDGSS searches for confidentiality policy.

### Using Teradata Schema Extensions

```
dn: cn=conf-qops,cn=policy1,cn=tdatrootP,dc=domain1,dc=com
objectClass: top
objectClass: tdatPolicyContainer
cn: conf-qops
```

### Using Native Directory Schema

```
dn: cn=conf-qops,ou=policy1,ou=tdatrootP,dc=domain1,dc=com
objectClass: top
objectClass: groupOfNames
cn: conf-qops
```

## Creating Confidentiality QOP Objects in the Confidentiality QOP Container

Create a QOP object for each of the QOP strengths defined in the TdgssUserConfigFile.xml (default, low, medium, and high), as a child of the conf-qops container.

### Using Teradata Schema Extensions

```
dn: cn=default,cn=conf-qops,cn=policy1,cn=tdatrootP,dc=domain1,dc=com
objectClass: tdatConfidentialityQoP
cn: default
```

```
dn: cn=low,cn=conf-qops,cn=policy1,cn=tdatrootP,dc=domain1,dc=com
objectClass: tdatConfidentialityQoP
cn: low

dn: cn=medium,cn=conf-qops,cn=policy1,cn=tdatrootP,dc=domain1,dc=com
objectClass: tdatConfidentialityQoP
cn: medium

dn: cn=high,cn=conf-qops,cn=policy1,cn=tdatrootP,dc=domain1,dc=com
objectClass: tdatConfidentialityQoP
cn: high
```

## Using Native Directory Schema

```
dn: cn=default,ou=conf-qops,ou=policy1,ou=tdatrootP,dc=domain1,dc=com
objectClass: groupOfNames
cn: default
member: ou=conf-qops,ou=policy1,ou=tdatrootP,dc=domain1,dc=com

dn: cn=low,ou=conf-qops,ou=policy1,ou=tdatrootP,dc=domain1,dc=com
objectClass: groupOfNames
cn: low
member: ou=conf-qops,ou=policy1,ou=tdatrootP,dc=domain1,dc=com

dn: cn=medium,ou=conf-qops,ou=policy1,ou=tdatrootP,dc=domain1,dc=com
objectClass: groupOfNames
cn: medium
member: ou=conf-qops,ou=policy1,ou=tdatrootP,dc=domain1,dc=com

dn: cn=high,ou=conf-qops,ou=policy1,ou=tdatrootP,dc=domain1,dc=com
objectClass: groupOfNames
cn: high
member: ou=conf-qops,ou=policy1,ou=tdatrootP,dc=domain1,dc=com
```

## Adding Members to a Confidentiality QOP to Require QOP Usage

You can assign a confidentiality QOP by Vantage user, directory user, Vantage profile, and internal or external network group.

If a user is a member of more than one confidentiality policy, the strongest confidentiality policy takes precedence.

**Note:**

For a list of rules governing the specification of users as policy members, see [Rules for Specifying Users as Policy Members](#).

## Applying Confidentiality QOP Policy to Teradata Vantage Users

The following examples show how to assign membership in a QOP to an existing Teradata user object in the directory. A directory principal mapped to the Teradata user object inherits the QOP policy.

When using Teradata schema extensions:

```
dn: cn=high,cn=conf-qops,cn=policy1,cn=tdatrootP,dc=domain1,dc=com
changetype: modify
add: member
member: cn=xyz,cn=users,cn=system1,cn=tdatrootA,dc=domain1,dc=com
-
```

When using native directory schema:

```
dn: cn=high,ou=conf-qops,ou=policy1,ou=tdatrootP,dc=domain1,dc=com
changetype: modify
add: member
member: cn=xyz,ou=users,ou=system1,ou=tdatrootA,dc=domain1,dc=com
-
```

## Applying Confidentiality QOP Policy to Directory Principals

You can specify a directory principal DN for an LDAP user not mapped to a Teradata user object.

**Note:**

You can specify policy membership for directory principals only if AuthorizationSupported=no for both KRB5 and LDAP.

When using Teradata schema extensions:

```
dn: cn=default,cn=conf-qops,cn=policy1,cn=tdatrootP,dc=domain1,dc=com
changetype: modify
add: member
member: uid=jdoe,ou=principals,dc=domain1,dc=com
-
```

When using native directory schema:

```
dn: cn=default,ou=conf-qops,ou=policy1,ou=tdatrootP,dc=domain1,dc=com
changetype: modify
add: member
member: uid=jdoe,ou=principals,dc=domain1,dc=com
-
```

## Applying Confidentiality QOP Policy to Teradata Vantage Profiles

The following examples show how to assign membership in a QOP to an existing profile object in the directory.

---

### Note:

When users log on to the Vantage through an application that uses pooled sessions, the users are subject to the profile for the application user, not profiles in which the users have membership or to which they are mapped.

---

When using Teradata schema extensions:

```
dn: cn=low,cn=conf-qops,cn=policy1,cn=tdatrootP,dc=domain1,dc=com
changetype: modify
add: member
member: cn=profile1,cn=profiles,cn=system1,cn=tdatA,dc=domain1,dc=com
```

When using native directory schema:

```
dn: cn=low,cn=conf-qops,cn=policy1,cn=tdatrootP,dc=domain1,dc=com
changetype: modify
add: member
member: cn=profile1,ou=profiles,ou=system1,ou=tdatrootA,dc=domain1,dc=com
```

## Applying QOP Restrictions to Network Groups

The examples below show configurations for internal network groups. Configuration of external groups is similar.

When using Teradata schema extensions:

```
dn: cn=medium,cn=conf-qops,cn=policy1,cn=tdatrootP,dc=domain1,dc=com
changetype: modify
add: member
```

```
member: cn=behind-the-firewall,cn=internal-network-groups,cn=policy1,
cn=tdatrootP,dc=domain1,dc=com
```

When using native directory schema:

```
dn: cn=medium,ou=conf-qops,ou=policy1,ou=tdatrootP,dc=domain1,dc=com
changetype: modify
add: member
member: cn=behind-the-firewall,ou=internal-network-groups,ou=policy1,
ou=tdatrootP,dc=domain1,dc=com
```

## Removing Members from a Confidentiality QOP

To remove members from a confidentiality QOP, substitute the `remove: attribute` for the `add: attribute` shown in the procedure for adding members, [Assigning Members to an Integrity QOP Policy](#), then list the unneeded members with member attributes.

For example:

```
dn: cn=high,cn=conf-qops,cn=policy1,cn=tdatrootP,dc=domain1,dc=com
changetype: modify
remove: member
member: cn=xyz,cn=users,cn=system1,cn=tdatrootA,dc=domain1,dc=com
```

## Configuring Options Policies

Additional security policy choices are available as Options policies, some of which are applicable only in a Unity environment.

To configure an Options policy:

1. Make sure you understand Options policy variations. See:
  - [About the Has-Policy Option](#)
  - [About the No-Direct-Connect Option](#)
2. Create the Options container. See [Creating the Options Container](#).
3. Create the needed Options policy objects. See:
  - [Creating the has-policy Option](#)
  - [Creating the no-direct-connect Option](#)
4. Add members to each Options policy. See [Adding Members to an Option Policy](#).
5. You can also remove members from an Options policy, if necessary. See [Removing Members from an Option Policy](#).

## About the Has-Policy Option

In a Unity environment, application of the has-option policy causes the system to transmit message traffic between Unity and a connected database in clear text.

The has-policy option is useful if the Unity server is co-located with the connected Teradata Vantage systems. Encryption is maintained between Teradata Vantage clients and the Unity server, while being eliminated for an otherwise secure connection between the Unity server and Vantage, saving processing costs associated with the unneeded encryption-decryption cycle.

- If you enable the has-policy option, and neither the Unity server IP address or the Unity user that connects to Vantage has a QOP explicitly defined, the system requires the transmittal in clear text.
- If the Unity user or IP address has an assigned QOP policy, the system ignores the has-policy option.
- If you do not enable the has-option policy, and the Unity user or IP address does not have an assigned QOP, the system uses the same QOP that applies to transmissions between the client and Unity.

You can apply the has-option policy to the DN of a:

- Vantage user name (tdatUser object) or a directory user name (directory principal)
- Vantage profile name (tdatProfile object)
- Network group (tdatNetworkGroup object)

## About the No-Direct-Connect Option

Application of the no-direct-connect option causes the gateway to reject logons from a networked client, and instead requires a mainframe connection or a connection through Unity.

You can restrict direct logons by specifying the DN of a:

- Vantage user name (tdatUser object) or directory user name (directory principal)
- Vantage profile name (tdatProfile object)
- Network group (tdatNetworkGroup object)

## Creating the Options Container

To configure an options policy, first create the options container as a child of a policy object.

Using Teradata schema extensions:

```
dn: cn=options,cn=policy1,cn=tdatrootP,dc=domain1,dc=com
objectClass: tdatPolicyContainer
cn: options
```

Using native directory schema:

```
dn: ou=options,ou=policy1,ou=tdatrootP,dc=domain1,dc=com
objectClass: organizationalUnit
ou: options
```

## Creating the has-policy Option

Using Teradata schema extensions:

```
dn: cn=has-policy,cn=options,cn=policy1,cn=tdatrootP,dc=domain1,dc=com
objectClass: tdatOption
cn: has-policy
```

Using native directory schema:

```
dn: cn=has-policy,ou=options,ou=policy1,ou=tdatrootP,dc=domain1,dc=com
objectClass: groupOfNames
cn: has-policy
member: ou=options,ou=policy1,ou=tdatrootP,dc=domain1,dc=com
```

## Creating the no-direct-connect Option

Using Teradata schema extensions:

```
dn: cn=no-direct-connect,cn=options,cn=policy1,cn=tdatrootP,dc=domain1,
  dc=com
objectClass: tdatOption
cn: no-direct-connect
```

Using native directory schema:

```
dn: cn=no-direct-connect,ou=options,ou=policy1,ou=tdatrootP,dc=domain1,
  dc=com
objectClass: groupOfNames
cn: no-direct-connect
member: ou=options,ou=policy1,ou=tdatrootP,dc=domain1,dc=com
```

## Adding Members to an Option Policy

You can use the member attribute of an Option policy object to add user, profile, or network group members that are subject to the effects of the option.



**Note:**

For a list of rules governing the specification of users as policy members, see [Rules for Specifying Users as Policy Members](#).

## Applying an Option Policy to a Teradata Vantage User

The following examples show how to apply an option policy (has-policy or no-direct-connect) to an existing tdatUser object in the directory.

Specify the DN of a tdatUser object for:

- A user that logs on with the TD2 mechanism
- An LDAP user that is mapped to the tdatUser object.

Example using Teradata schema extensions:

```
dn: cn=has-policy,cn=options,cn=policy1,
   cn=tdatrootP,dc=domain1,dc=com
changetype: modify
add: member
member: cn=xyz,cn=users,cn=system1,cn=tdatrootA,dc=domain1,dc=com
```

Example using native directory schema:

```
dn: cn=no-direct-connect,ou=options,ou=policy1,
   ou=tdatrootP,dc=domain1,dc=com
changetype: modify
add: member
member: cn=xyz,ou=users,ou=system1,ou=tdatrootA,dc=domain1,dc=com
```

## Applying an Option Policy to a Directory Principal

The following examples show how to assign membership in a specified option (has-policy or no-direct-connect) to a directory principal user.

**Note:**

You can specify policy membership for directory principals only if AuthorizationSupported=no for both KRB5 and LDAP.

For example, when using Teradata schema extensions:

```
dn: cn=no-direct-connect,cn=options,cn=policy1,
   cn=tdatrootP,dc=domain1,dc=com
changetype: modify
add: member
member: uid=dirUser1,cn=principals,dc=domain1,dc=com
-
```

For example, when using native directory schema, for example:

```
dn: cn=has-policy,ou=options,ou=policy1,
   ou=tdatrootP,dc=domain1,dc=com
changetype: modify
add: member
member: uid=dirUser1,ou=principals,dc=domain1,dc=com
-
```

## Applying an Option Policy to a Teradata Vantage Profile

When you configure option membership by Vantage profile, the corresponding tdatProfile object must already exist in the directory.

---

### Note:

Users accessing the database through a middle-tier application are subject to the policies assigned to the application user, instead of policies assigned to them as individuals.

---

For example, when using Teradata schema extensions, specify a profile member as follows:

```
dn: cn=has-policy,cn=options,cn=policy1,
   cn=tdatrootP,dc=domain1,dc=com
changetype: modify
add: member
member: cn=profile1,cn=profiles,cn=system1,cn=tdatrootA,dc=domain1,dc=com
-
```

For example, when using native directory schema, specify a profile member as follows:

```
dn: cn=no-direct-connect,ou=options,ou=policy1,
   ou=tdatrootP,dc=domain1,dc=com
changetype: modify
add: member
member: cn=profile1,ou=profiles,ou=system1,ou=tdatrootA,dc=domain1,dc=com
-
```

## Applying an Option Policy to a Network Group

When you configure option membership by network group, the type of network group determines the effects of membership.

- Internal network group IP addresses are subject to an Option in which the group is a member
- External network group IP addresses are exempt from an Option in which the group is a member.

For example, when using Teradata schema extensions:

```
dn: cn=no-direct-connect,cn=options,cn=policy1,
   cn=tdatrootP,dc=domain1,dc=com
changetype: modify
add: member
member: cn=behind-the-firewall,cn=internal-network-groups,cn=policy1,
       cn=tdatrootP,dc=domain1,dc=com
```

For example, when using native schema extensions:

```
dn: cn=has-policy,ou=options,ou=policy1,
   ou=tdatrootP,dc=domain1,dc=com
changetype: modify
add: member
member: cn=behind-the-firewall,ou=internal-network-groups,ou=policy1,
       ou=tdatrootP,dc=domain1,dc=com
```

## Removing Members from an Option Policy

You can remove members from an Options policy similarly to the method for adding members, for both Teradata and native schema.

Substitute the remove: attribute for the add: attribute shown in [Adding Members to an Option Policy](#), and remove the unneeded members.

For example:

```
dn: cn=has-policy,cn=options,cn=policy1,
   cn=tdatrootP,dc=domain1,dc=com
changetype: modify
remove: member
member: cn=behind-the-firewall,cn=internal-network-groups,cn=policy1,
       cn=tdatrootP,dc=domain1,dc=com
```

## Security Policies in the TDGSS Configuration

After configuring policy definitions in the directory, you must enable the policies by configuring the `<LdapConfig>` section in the `TdgssUserConfigFile.xml` on the Teradata Vantage system, and in the `TdgssUnityConfig.xml` on Unity (if used).

The `<LdapConfig>` section defines the directory services and any associated global or local security policy structures, so that TDGSS can find and enforce the applicable policy when a user logs on to Teradata Vantage.

## About Global and Local Security Policies

You can configure a global security policy and one or more local security policies (specific to each directory service) in the `TdgssUserConfigFile.xml`, using `<policy>` elements and associated properties.

A single global policy usually meets most needs.

Each `<policy>` element includes a value specifying the location of the corresponding Teradata Vantage security policy structure in a directory.

## Basic Policy Structure

The following example shows a local `<policy>` element located on the `hrsvc` service, and a global `<policy>` located below the `<Canonicalizations>` section.

```
<LdapConfig>

  <Tls ... />
  <Services>
    <Service
      Id="hrsvc" ... />
    <Service
      Id="domain2" ... />
    <Service
      Id="domain3" ... />
      <Policy
        .../>
      </Policy>
    </Service>
  </Services>
  <Canonicalizations>
    ...
  </Canonicalizations>
  <Policy
    .../>
```

```
</Policy>
</LdapConfig>
```

## Use Case for Configuring Global and Local Security Policies

A single global security policy structure usually meets company security needs. However, some companies may need to configure both global and local policies. For example:

Assume that a company has three semi-independent divisions, Human Resources, Development, and Marketing.

All three divisions store information on the same Teradata Vantage system.

Each division has a dedicated directory service to independently manage its users.

Human Resources deals with sensitive information about employees and needs to enforce a strict, local QOP (encryption) policy on their users.

Development and Marketing share security policy requirements so they can share the same global policy structure, whereas Human Resources requires a separate local policy structure.

## System Processing of Global and Local Policies

TDGSS follows certain rules when determining security policy for a session. When a user logs on to Vantage:

1. If the authentication mechanism is TD2, TDGSS first looks for a global policy and enforces any policies that apply to the user. Local policies do not apply to TD2 users.
2. If the authentication mechanism is not TD2, TDGSS searches each <service> element for a <policy> element (local policy).
  - a. If TDGSS finds a local policy that applies to the user, it enforces the policy.
  - b. If TDGSS does not find a local policy structure in the authenticating service, it searches for a global policy and enforces any global policy that applies to the user.
3. If the TdgssUserConfigFile.xml contains neither a local nor a global policy, the policy enforcement feature is disabled regardless of what may be configured in the directory.

## Configuring Security Policies in the TdgssUserConfigFile.xml

### Configuring the Directory Services

You can refer to the example configuration in [Use Case for Configuring Global and Local Security Policies](#) as an aid in understanding configuration steps.

**Note:**

To ensure uninterrupted operation, configure duplicate security policies in a backup directory and configure the `LdapServerName` property to automatically switch to the alternate directory in the event of failure. See [LdapServerName](#).

1. Make a backup copy of the `TdgssUserConfigFile.xml` file.
2. Add the `<LdapConfig>` section to `TdgssUserConfigFile.xml` on Teradata Vantage nodes, and to the `TdgssUnityConfig.xml` on the Unity server, if used. See [Adding Multiple Directory Services to the TDGSS Configuration](#). Use this procedure for configuring security policies even if you have only one directory service to configure.

**Note:**

If you have already configured multiple directory services in an `<LdapConfig>` section for LDAP authentication (as shown in [Configuring LDAP to Use Multiple Directory Services](#)), the existing configuration contains many of the elements necessary for policy configuration. You only need to add the required policy-related elements to the configuration.

- a. Open the `TdgssUserConfigFile.xml` for editing.
- b. Disable the existing LDAP mechanism, saving property settings for use in the `<LdapConfig>` section.
- c. Create the `<LdapConfig>` section.
- d. Add the optional `<Tls>` section, if required at your site. See [Using TLS with a Directory Server](#).
3. Configure an entry for each directory service using the standard LDAP properties needed for security policies. See [Standard LDAP Properties Used for All Policy Configurations](#).
4. Optionally configure a service element for a global security policy. See [Configuring Policy-Related Properties for a Global Security Policy](#).
5. Add the necessary policy-specific properties to each local service. See [Configuring Policy-Related Properties for a Local Security Policy](#).
6. Verify the configuration is correct:
  - a. Run `tdgsstestcfg` to test the configuration. It launches a test environment in a new shell that contains the updates to the configuration file.
 

```
/opt/teradata/tdgss/bin/tdgsstestcfg
```
  - b. Test the policy configuration using the `tdspolicy` tool. See [Investigating Security Policy Assignments](#).
  - c. Exit the test shell:
 

```
exit
```
7. After you complete the required edits to the `TdgssUserConfigFile.xml`, run the `run_tdgssconfig` utility to update the TDGSSCONFIG GDO.

```
/opt/teradata/tdgss/bin/run_tdgssconfig
```

8. If run\_tdgssconfig indicates that a TPA reset is required, run tpareset.

```
tpareset -f "use updated TDGSSCONFIG GDO"
```

## Directory Service Setup in the TdgssUserConfigFile.xml

The following example shows the configuration of the three directory services in the <LdapConfig> section of the TdgssUserConfigFile.xml

### Note:

The basic configuration structure required for defining security policies may have already been set up as part of configuring LDAP authentication using multiple directory services. See [Configuring LDAP to Use Multiple Directory Services](#).

```
<LdapConfig>

  <Tls .../>

  <Services>

    <Service
      Id="hrsvc"
      LdapServerName="_ldap._tcp.hr.domain.com"
      LdapBaseFQDN="dc=hr,dc=domain,dc=com"
      LdapServiceFQDN="uid=dbcsvc,ou=services,dc=hr,dc=domain, dc=com"
      LdapServicePassword="secret"
    .../>

    <Service
      Id="devsvc"
      LdapServerName="_ldap._tcp.dev.domain.com"
      LdapBaseFQDN="dc=dev,dc=domain,dc=com"
      LdapServiceFQDN="uid=dbcsvc,ou=services,dc=dev,dc=domain, dc=com"
      LdapServicePassword="secret"
    .../>

    <Service
      Id="mktsvc"
      LdapServerName="_ldap._tcp.mkt.domain.com"
      LdapBaseFQDN="dc=mkt,dc=domain,dc=com"
      LdapServiceFQDN="uid=dbcsvc,ou=services,dc=mkt,dc=domain, dc=com"
```

```

        LdapServicePassword="secret"
        .../>

</Services>

<Canonicalizations>

    <IdentitySearch
        Ref="hrsvc"
        Match="hr_(.*)"
        Base="dc=hr,dc=domain,dc=com"
        Filter="(uid=${1})"
        Scope="subtree"
        DatabaseName="${1}"/>

    <IdentitySearch
        Ref="devsvc"
        Match="dev_(.*)"
        Base="dc=dev,dc=domain,dc=com"
        Filter="(uid=${1})"
        Scope="subtree"
        DatabaseName="${1}"/>

    <IdentitySearch
        Ref="mktsvc"
        Match="mkt_(.*)"
        Base="dc=mkt,dc=domain,dc=com"
        Filter="(uid=${1})"
        Scope="subtree"
        DatabaseName="${1}"/>

</Canonicalizations>

</LdapConfig>

```

## Standard LDAP Properties Used for All Policy Configurations

You can configure the following properties for any Service element (directory) that contains a security policy structure. When adding Policy elements to services previously configured for LDAP authentication, you may find that some of these properties are already configured. For information on configuring Service elements for LDAP, see [Configuring LDAP to Use Multiple Directory Services](#).



Property Name	Description
LdapServerName	Required, Identifies the directory that contains the policy being configured. Must be a valid URI or DNS SRV RR specification. For details, see <a href="#">LdapServerName</a> .
LdapServiceFQDN	Required unless the service is anonymously readable. Identifies the bindable object in the directory that represents the service identity, that is, the Teradata Vantage system or Unity server that contains the TDGSS configuration file that is being configured. See <a href="#">LdapServiceFQDN</a> .  <b>Note:</b> Directories that serve multiple Vantage systems should contain a separate bindable object for each system and for the Unity server, if used.
LdapServicePassword	If your site security policy requires a password for the service FQDN, configure a password as the value of this property. For details, see <a href="#">LdapServicePassword</a> .
LdapServicePasswordProtected	Indicates whether the LDAP service password (if used) is encrypted. <ul style="list-style-type: none"> <li>• Yes means that TDGSS stores the LdapServicePassword in encrypted form.</li> <li>• No (the default) means that TDGSS stores the LdapServicePassword in plain text.</li> </ul> For details, see <a href="#">LdapServicePasswordProtected</a> .
LdapSystemFQDN	Identifies the FQDN of the tdatSystem directory object, to assist in constructing the DNs of Vantage users and profiles.
LdapBaseFQDN	Specifies the FQDN of the directory object that contains directory users and groups, which provides the search base for locating user and group objects. Not required if the LdapNetworkBaseFQDN is configured. See <a href="#">Configuring Policy-Related Properties for a Global Security Policy</a> and <a href="#">Configuring Policy-Related Properties for a Local Security Policy</a> .

For additional information on configuring LDAP properties, see [TDGSS Configuration Files, Valid Settings, and Editing Guidelines](#).

## Configuring Policy-Related Properties for a Global Security Policy

To configure a global security policy, you must add a Policy element after the Canonicalizations section in the LdapConfig section of the TdgssUserConfigFile.xml, and add the necessary attributes and values.

A global policy can contain the following attributes.

Attribute Name	Required	Description
Ref	Yes	The service (directory) that contains the global policy.
LdapPolicyFQDN	Yes	The FQDN of the policy container in the directory that contains the global policy structure.
LdapNetworkBaseFQDN	No	Locates the container for ipNetwork entries. If LdapNetworkBaseFQDN is not provided, the system uses the value in the LdapBaseFQDN attribute for the containing service. If no value is present for either LdapBaseFQDN or LdapNetworkBaseFQDN, the system does not use the client IP address for determining the applicable QOP policy or options.

For example:

```
<LdapConfig>

  <Tls ... />
  <Services>
    ...
  <Services>
  <Canonicalizations>
    ...
  </Canonicalizations>
  <Policy
    Ref="globalpolicysvc"
    LdapPolicyFQDN="?"
    LdapNetworkBaseFQDN="dc=domain,dc=com"/>
  </Policy>
</LdapConfig>
```

## Configuring Policy-Related Properties for a Local Security Policy

A local security policy is contained in a service and applies only to that service. For example:

```
<Services>
  <Service
    Id="domain1" ... />
  <Service
    Id="domain2" ... />
  <Service
    Id="domain3" ... />
    <Policy
      ..."/>
```

```

    </Policy>
  </Service>

```

If the TdGSSUserConfigFile.xml already contains <Service> elements configured for LDAP authentication or authorization (see [Configuring LDAP to Use Multiple Directory Services](#)), you can add any necessary <Policy> elements to the existing Service elements.

Attribute Name	Required	Description
LdapPolicyFQDN	Yes	Locates the policy container
LdapNetworkBaseFQDN	No	Locates the container for ipNetwork entries. If LdapNetworkBaseFQDN is not provided, the system uses the value in the LdapBaseFQDN attribute for the containing service. If no value is present for either LdapBaseFQDN or LdapNetworkBaseFQDN, the system does not use the client IP address for determining the applicable QOP policy or options.

## Sample Configuration Containing Both Local and Global Policies

The following example shows configured policy elements in the TdgssUserConfigFile.xml.

```

<LdapConfig>

  <Tls ... />
  <Services>
    <Service
      Id="globalpolicysvc"
      LdapServerName="_ldap.tcp.domain.com"
      LdapServiceFQDN="cn=div1,ou=services,dc=domain1,dc=com"
      LdapSystemFQDN="cn=system1,cn=tdat,dc=domain1,dc=com"
      LdapServicePassword="password"... />
    <Service
      Id="domain1" ... />
    <Service
      Id="domain2" ... />
    <Service
      Id="domain3" ... />
      LdapServerName="_ldap.tcp.domain.com"
      LdapServiceFQDN="cn=div1,ou=services,dc=domain,dc=com"
      LdapSystemFQDN="cn=systemone,cn=tdat,dc=domain,dc=com"
      LdapServicePassword="password" ... />
    <Policy
      LdapPolicyFQDN="cn=policy1,ou=tdatrootP,dc=domain1,dc=com"
      LdapNetworkBaseFQDN="dc=networks,dc=domain1,dc=com"/>
  
```

```

        </Policy>
    </Service>
</Services>
<Canonicalizations>
...
</Canonicalizations>
<Policy
    Ref="globalpolicysvc"
    LdapPolicyFQDN="cn=policyGLO,ou=tdatrootP,dc=domain1,dc=com"
    LdapNetworkBaseFQDN="dc=networks,dc=domain1,dc=com"/>
</LdapConfig>

```

**Note:**

The example above shows an entry  
of: LdapServiceFQDN="cn=div1,ou=services,dc=domain1,dc=com"

which is valid only for Active Directory, ADAM and AD LDS. For other directory types, the configuration must specify:

LdapServiceFQDN="uid=div1,ou=services,dc=domain1,dc=com"

## Configuring the Gateway to Allow Logons from Older Interfaces or Proxies

The gateway can be configured to allow logons from older client interfaces or proxies that do not support Teradata Vantage network security policy, even when security policy is configured. A Gateway Control (gtwcontrol) utility option can be set to allow older clients to log on even if security policy that they are unable to automatically follow has been set for them. This allows a mix of newer clients that can accept security policy from the database and older clients that cannot accept it.

When the gateway has been configured to allow logons from older client interfaces, the client interfaces must be manually configured to be within policy or they will be forced off for violating policy.

When the gateway has been configured to allow logons from older proxies, these proxies cannot guarantee that the clients logging on through them can automatically follow policy, nor can they transmit policy to clients that could otherwise automatically follow it. For this reason, all clients that log on through such proxies must be manually configured to be within policy or they will be forced off for violating policy.

For more information, see *Teradata Vantage™ - Database Utilities*, B035-1102.

To identify client interfaces or proxies that do not support Teradata Vantage network security policy, see [Auditing Logons by Clients that Cannot Automatically Follow Security Policy](#).

### About the Default Setting for --secpynotsupported logon

By default the gateway does not allow logons by clients or proxies that are unable to support security policy when security policy applies. To restore the default setting, use:

```
gtwcontrol --sepcynotsupported logon=no
```

To enable the gateway to allow these logons, use Gateway Control:

```
gtwcontrol --sepcynotsupported suboptions
```

## Examples: Enabling Clients and Proxies that are Unable to Automatically Support Security Policy to Log On

### Example: Enabling Logon for All

Setting the `--sepcynotsupported logon` flag to `all` configures the gateway to allow logons using clients or proxies that are unable to automatically support security policy, even when policy applies.

```
gtwcontrol --sepcynotsupported logon=all
```

A client that cannot automatically follow policy that has not been manually configured to be within policy can send a single out-of-policy message per session before the security violation is caught and the session is logged off.

Proxies that cannot automatically follow security policy cannot guarantee that the clients that connect through them follow policy, nor can they transmit policy to clients that could otherwise follow it. For this reason, all clients that log on through such proxies must be manually configured to be within policy, even if they are otherwise capable of following policy automatically. In practice, the gateway can identify security violations by client sessions logged on through such a proxy and log them off, but not until after a single out-of-policy message has already been sent.

### Example: Enabling Logon for Clients

Setting the `--sepcynotsupported logon` flag to `client` configures the gateway to allow logons using clients that are unable to automatically support security policy, even when policy applies.

```
gtwcontrol --sepcynotsupported logon=client
```

A client that cannot automatically follow policy that has not been manually configured to be within policy can send a single out-of-policy message per session before the security violation is caught and the session is logged off.

### Example: Enabling Logon for Proxy

Setting the `--sepcynotsupported logon` flag to `proxy` configures the gateway to allow logons through proxies that are unable to automatically support security policy, even when policy applies.

```
gtwcontrol --sepcynotsupported logon=proxy
```

Proxies that cannot automatically follow security policy cannot guarantee that the clients that connect through them follow policy, nor can they transmit policy to clients that could otherwise follow it. For this reason, all clients that log on through such proxies must be manually configured to be within policy, even if they are otherwise capable of following policy automatically. In practice, the gateway can identify security violations by client sessions logged on through such a proxy and log them off, but not until after a single out-of-policy message has already been sent.

## Related Information

For more information, see *Teradata Vantage™ - Database Utilities*, B035-1102.

To enable logging for security policy, see [Auditing Logons by Clients that Cannot Automatically Follow Security Policy](#).

## Requiring Confidentiality

You can use the gtwcontrol RequireConfidentiality flag to require the use of encryption globally, for all messages between the client and database.

---

### Note:

Turning on RequireConfidentiality on the database system enforces message encryption for new sessions that are logged on after the flag is turned on, but traffic between client systems and the database system for existing (already logged on) sessions is not encrypted. In order to enforce traffic encryption on all the sessions that connect to the database system, all existing sessions must be logged off prior to turning on RequireConfidentiality. It is recommended to turn on the RequireConfidentiality option only when the system is quiescent.

---

To enable this functionality, use gtwcontrol:

```
gtwcontrol -x YES
```

If the system is set up with host groups, which are separate Vantage gateways for groups of IP addresses, you can set the confidentiality requirement separately for each host group, for example:

```
gtwcontrol -x YES -g [host_ID]
```

Also see [Restricting Logons by Host Group](#), and Gateway Control (gtwcontrol) in *Teradata Vantage™ - Database Utilities*, B035-1102.

If no other confidentiality policy applies, a session that is subject to the RequireConfidentiality flag uses the DEFAULT QOP, as configured in the TdgssUserConfigFile.xml.

**Teradata Tools and Utilities (TTU) clients:** If the RequireConfidentiality flag is set, the gateway server sends the security policy information in the logon response back to the client, informing the client interface (such as ODBC, JDBC, CLI, or .NET Data Provider for Teradata) that all requests must be encrypted for this

session. TTU client interfaces are able to read and comply with the security policy information in the logon response. This means the client silently follows the policy and encrypts the messages, whether or not the application enables or disables the data encryption option. So messages are automatically encrypted even though the enable data encryption option was not set. For example, if the user did not set the ODBC DSN encrypt option and RequireConfidentiality is set, messages are encrypted.

If other security policies that require the use of a stronger QOP also apply to the session, the system defers to the stronger QOP.

## Setting Up Host Groups

To require confidentiality by host group, you must set up host groups:

1. A Teradata Customer Service representative configures PDE to define multiple host groups using the Vconfig utility. Each host group appears as a separate HGID in the vconfig.txt file.
2. A Teradata Customer Service representative configures the database to define multiple hosts using the Configuration utility ADD HOST command. Each host must include the same vprocs as the corresponding host group in Vconfig. You can verify the current host configuration with the LIST command.
3. The network administrator assigns multiple aliases (tdpids) to the Teradata Vantage system, and maps each tdpid, which corresponds to a configured host group, to a set of COP names and IP addresses.
4. The network administrator assigns a Vantage client or group of clients to a single tdpid that corresponds to a host group.

## Investigating Security Policy Assignments

After configuring security policy elements in the directory and the TdgssUserConfigFile.xml, you can use the tdsbind and tdspolicy tools to verify that the security policies are assigned to users as you intended. For additional information about using tdsbind, see [Working with tdsbind](#).

## Using tdgssauth to Determine tdspolicy Search Parameters

Before using tdspolicy to investigate the security policy applicable to a user defined in the directory, you can run tdgssauth to determine parameters that may be required to run tdspolicy.

```
$ tdgssauth -m ldap -u diperm01 -i 141.206.3.15
TDGSS_BIN_FILE not set.
TDGSSCONFIG GDO used in tdgss.
Please enter a password:
                Status: authenticated, not authorized
                Database user: perm01 [permanent user]
                Profile: profile01
                External roles: extrole01perm01,
extrole02perm01, extrole03perm01
```

```

Authenticated user: ldap://
esroot.example.com:389/CN=diperm01,OU=people,OU=testing,DC=example,DC=com
Audit trail identifier: diperm01
Authenticating service: esroot1
Actual mechanism employed: ldap [OID 1.3.6.1.4.1.191.1.1012.1.20]
Mechanism specific data: diperm01

Security context capabilities: replay detection
                             out of sequence detection
                             confidentiality
                             integrity
                             protection ready
                             exportable security context

Minimum quality of protection: high with confidentiality and integrity
Options: none

$

```

## Using tdspolicy to Find Policy Assignments for a User

You can run the tdspolicy tool from the command prompt on a Teradata Vantage node to investigate the security policy assignments that are currently in effect for a specific combination of user, profile, and logon IP address.

tdspolicy example:

```
tdspolicy -u user -i ip_address [-s service] [-p profile]
```

### ***user***

Specify a Vantage user name in these cases:

- The user is authenticated by Teradata (TD2 mechanism)
- The user is authenticated by Kerberos (KRB5 mechanism) or LDAP and AuthorizationSupported=no
- The user is authenticated by Kerberos (KRB5 mechanism) or LDAP, AuthorizationSupported=yes, and the user is mapped to a tdatUser entry.

If a directory user is mapped to multiple tdatUser objects, and more than one object has security policy assignments, the most restrictive policy applies. For details, see the configuration instruction for each policy type.

Specify the DN of a directory principal for a directory user if the user is authenticated using KRB5 or LDAP, AuthorizationSupported=yes, and the user is not mapped to a tdatUser entry.



***ip\_address***

The IP address from which the user logs on.

***service***

[Required to return information on a local security policy.] Specify the DN of the service that contains the local policy.

If the -u user authenticates in a specific service, -s must specify the DN of that service.

If this option is not present to request local policy information for a specific service, tdspolicy returns information for the global policy, if a global policy exists.

For information on global policy, see [Configuring Policy-Related Properties for a Global Security Policy](#).

***profile***

[Optional] Identifies an existing profile that is assigned to the user.

For permanent Vantage users, *profile* is the profile specified in the user definition. For directory principals, it is a profile to which the principal is mapped in the directory.

The tdspolicy command returns information indicating whether any policy applies to the specified profile.

If a directory principal is mapped to a Vantage user and a profile in the directory, the mapped profile takes precedence over the profile assigned to the mapped permanent user.

For externally authenticated or authorized users, you can use tdgssauth to obtain the tdspolicy command line arguments:

```
$ tdgssauth -m ldap -u diperm01 -i 141.206.3.15
TDGSS_BIN_FILE not set.
TDGSSCONFIG GDO used in tdgss.
Please enter a password:
                Status: authenticated, not authorized
                Database user: perm01 [permanent user]
                Profile: profile01
                External roles: extrole01perm01,
extrole02perm01, extrole03perm01
                Authenticated user: ldap://
esroot.example.com:389/CN=diperm01,OU=people,OU=testing,DC=example,DC=com
                Audit trail identifier: diperm01
                Authenticating service: esroot1
                Actual mechanism employed: ldap [OID 1.3.6.1.4.1.191.1.1012.1.20]
                Mechanism specific data: diperm01
```

```

Security context capabilities: replay detection
                             out of sequence detection
                             confidentiality
                             integrity
                             protection ready
                             exportable security context

Minimum quality of protection: high with confidentiality and integrity
Options: none

$

```

## tdspolicy for a Directory Principal Mapped to a Teradata Vantage User

If a directory principal is mapped to a Teradata user object, specify the -u as the name of the database user.

```

$ tdspolicy -u perm01 -p profile01 -s local -i 141.206.3.15
Querying policy using the following parameters:

    Teradata user: perm01
    Teradata profile: profile01
    IP address: 141.206.3.15

    Mechanisms: td2, ldap
Confidentiality QoPs: high
    Integrity QoPs: low
    Options: no-direct-connect

```

where:

- The directory principal (-u) can use only the TD2 or LDAP mechanism to log on.

---

### Note:

Profile01, which is mapped to the directory principal, applies only for LDAP logons. Profile-based policy does not apply to TD2 sessions.

---

- The system automatically uses the high confidentiality QOP (which supersedes the low integrity QOP) for all user message transmissions.
- The directory principal cannot connect directly to the database from the network, but must log on through a Unity tdpid or a mainframe connection.

## tdspolicy for a Directory Principal not Mapped to a Teradata User

For directory principals not mapped to a Teradata user, specify -u as the directory principal user name, along with the IP address and any mapped profile.

```
$ tdspolicy -u uid=drct01,ou=principals,dc=domain1,dc=com -p profile01
-s local_service_DN -i 141.206.3.219
Querying policy using the following parameters:
```

```
    External user: uid=drct01,ou=principals,dc=domain1,dc=com
    Teradata profile: profile01
    IP address: 141.206.3.219
```

```
    Mechanisms: krb5, ldap
    Confidentiality QoPs: low, high
```

where the directory principal specified by -u:

- Can use only the KRB5, SPNEGO, or LDAP mechanism to log on.
- Must use confidentiality with high QOP for LDAP logons. Confidentiality is also enforced for KRB5 and SPNEGO logons, but the QOP strength is determined by Kerberos.

## tdspolicy for a TD2 User

Users who log on with the TD2 mechanism are not subject to local policy because they are not authenticated or authorized in the directory. When you specify a Vantage username for -u, TDGSS looks in the TdgssUserConfigFile.xml to see if a global policy applies to the user.

---

### Note:

Profile-based policies do not apply to users authenticated by TD2.

```
$ tdspolicy -u td2user -i 141.206.3.173
Querying policy using the following parameters:
```

```
    Teradata user: td2user
    IP address: 141.206.3.173
```

```
    Mechanisms: td2
    Confidentiality QoPs: default
```

---

where the Vantage user specified by -u:

- Can use only the TD2 mechanism to log on.
- Confidentiality is required, but because a TD2 user is not authenticated or authorized in the directory, QOP strength defaults to the DEFAULT QOP.

## tdspolicy for a TD2 User with No Assigned Policy

If a user has no policy assignments, tdspolicy returns something like the following:

```
$ tdspolicy -u othertd2user -i 141.206.3.139
Querying policy using the following parameters:
```

```
Teradata user: othertd2user
IP address: 141.206.3.139
```

## Monitoring QOP Security Policy

When the system enforces QOP security policy for a session, it logs policy-related information for the session.

## Logging QOP Security Policy in the DBC.LogOnOffV

Teradata Vantage logs session information in the DBC.LogOnOffV, including information related to security policy.

DBC. LogOnOffV Field	Description
Security_Policy	Names the QOP security policy enforced by: <ul style="list-style-type: none"> <li>• The gateway, on a direct connection between the client and the gateway</li> <li>• Unity, on a connection between the client and Unity</li> </ul> The DBC.LogOnOffV view does not track violations of mechanism policy.
Unity_ Security_Policy	Names the QOP security policy enforced by the Gateway on the connection between Unity and the Gateway.
Unity_AuthUser	The DN of the tdatUser object that represents the Unity user, by which TDGSS authenticates the Unity server connection to the database.

## Logging QOP Security Policy on Unity

Unity logs session information in the Unity security log, including information on QOP security policy. For information, see *Teradata® Unity™ User Guide*, B035-2520.

## Logging QOP Policy Violations

Violations of policy during logon are logged as an 8057 error in the system log at /var/log/messages. Coordinate the session ID in the system log with the session ID in the DBC.LogOnOffV or Unity security log to determine which policy was violated.

## Auditing Policy Violations

You can check the system log at /var/log/messages to find 8057 error messages, and then use the session ID to search the DBC.LogOnOffV view to find which users are experiencing the policy violations.

# Using Logon Controls

## Logon Control Implementation Options

- Grant logon privileges. See [About Logon Privileges](#).
- Revoke logon privileges. See [Granting and Revoking Logon Privileges](#).
- Enable specialized controls for:
  - Application user logons. See [Controlling Logons through a Middle-Tier Application](#).
  - Logons through a host ID. See [Restricting Logons by Host Group](#).
  - Logons by client IP Address. See [Restricting Logons by IP Address](#).

## About Logon Privileges

By default, the database automatically grants logon permission to all users defined in the database, from all client system connections (hostids).

You only need to grant the logon privilege to a user if you:

- First revoke the logon privilege
- Want to specify an extension of logon privileges to GRANT LOGON ... WITH NULL PASSWORD, to allow the user to be externally authenticated.
- Want to specify user logons through a host ID or list of host IDs.

## Controlling the Granting and Revoking of Logons

The ability to grant and revoke logons for database users is controlled by the EXECUTE privilege on the DBC.LogonRule macro. User DBC has this privilege by default. You must grant the privilege to any other user who needs to use the GRANT LOGON or REVOKE LOGON statement. For an example:

```
GRANT EXECUTE ON DBC.LogonRule
```

See [Creating the Security Administrator User](#) for other administrator privileges.

When an administrator with the GRANT EXECUTE ON DBC.LogonRule privilege submits a GRANT LOGON or REVOKE LOGON statement, the DBC.LogonRule macro adds or deletes a row in the DBC.logonRules table for the affected user. See the information about DBC.LogonRulesV in *Teradata Vantage™ - Data Dictionary*, B035-1092 and the information about GRANT LOGON and REVOKE LOGON in *Teradata Vantage™ - SQL Data Control Language*, B035-1149.

## Working with Precedence of Clauses

The system enforces the rules in a GRANT LOGON or REVOKE LOGON statement according to the host group and user ID clauses in the statement. More specific clauses contained in a current GRANT or REVOKE statement take precedence over any more general clauses from a previous GRANT or REVOKE statement, for example:

```
GRANT LOGON ON ALL TO Adam
```

is superseded by limitation on host ID HGID2:

```
REVOKE LOGON ON HGID2
```

if user Adam is assigned to host group HGID2.

## Granting and Revoking Logon Privileges

You can use the GRANT LOGON or REVOKE LOGON statement to enable or disable:

- All logons to the database for a particular user
- Logons to the database by a user through one or more host IDs. For information on setting up host groups, see [Restricting Logons by Host Group](#).

### Syntax

```
{ GRANT | REVOKE } LOGON ON hostid FROM username
```

### Syntax Elements

#### *hostid*

One of the following:

- A single *hostid*
- A comma-separated list of *hostids*
- ALL

#### *username*

One of the following:

- Vantage username
- Comma-separated list of user names

A REVOKE LOGON statement affects only future logon attempts. It does not affect currently logged on users. You can restore logon privileges with the GRANT LOGON statement.

## Enforcing Temporary Logon Restrictions When Restoring Data

An attempt to execute a RESTORE DBC may fail and cause a database restart if users other than DBC are logged on to the system. This and other administrative activities may require that you temporarily disable all logons to the database until the activity is complete. Do not use REVOKE LOGONS for this purpose. Instead use the DISABLE LOGONS command to:

- Prevent subsequent non-DBC logons
- Disallow the execution of RESTORE DBC until all non-DBC users are off the system

---

### Note:

Users logged on at the time a DISABLE LOGONS command is issued continue their sessions to completion.

---

To safely run a RESTORE DBC operation and avoid an unplanned database restart:

1. Use Database Window (xdbw) to submit the DISABLE LOGONS, and prevent subsequent non-DBC user logons. All current session continue to completion.
2. When all non-DBC users are off the system, run the RESTORE operation.
3. When the RESTORE is complete, you can use the Database Window ENABLE ALL LOGONS command to re-enable all user logon privileges defined in the database.

For more information, see the information about the Database Window in *Teradata Vantage™ - Database Utilities*, B035-1102.

## Revoking Logons in a Unity Environment

When Unity manages multiple Teradata Vantage systems, you can submit REVOKE LOGON ALL through Unity to halt system operations, for example, for maintenance or software upgrade. If you only need to lock users out of a single database system, use Unity routing rules instead of REVOKE LOGON.

## Controlling Logons through a Middle-Tier Application

Many systems include middle-tier, non-Teradata, applications that stand between end-users and Teradata Vantage. A middle-tier application typically logs on to the database as a single permanent database user, and establishes a connection pool. The application then authenticates individual end-users, some of whom may request access to the database through the application connection pool.

By default, Vantage authorizes database privileges for application end-users, and monitors their database activity, based on the logon user identity of the application server.

You can optionally set up middle-tier applications to run trusted sessions, which enables the system to individually identify, authorize, and monitor application end-users.



## Setting Up Trusted Sessions and Proxy Users

1. Grant the CTCONTROL privilege to one or more administrators, which allows them to grant trusted user status to middle-tier applications, and to define proxy users and their database privileges.

See [About the CTCONTROL Privilege](#).

2. Create a database user for the trusted user application, that is, the identity with which the application logs on to Teradata Vantage.
3. Use a GRANT CONNECT THROUGH statement to define:
  - An existing database user identity for the trusted user
  - One or more proxy users who can log on to the database through the trusted user
  - One or more database role names, which define privileges available to the proxy users
  - A profile, which defines session attributes for application proxy users, including their temp and spool space and query band parameters.

---

### Note:

You can submit a separate GRANT CONNECT THROUGH statement with the WITH TRUST\_ONLY clause to prevent end-users from submitting SET QUERY\_BAND statements that set or update a proxy user.

See [Working with Middle-Tier Application Users](#).

---

4. Set up the trusted user application to use query banding, which collects user information and sends it to the database. Since the application authenticates the end users, logons to the database from the application must include a SET QUERY\_BAND statement to send proxy user information to the database.

Developers or application programmers must embed code in the middle-tier application program to derive the required information from the user logon, insert it into a SET QUERY\_BAND statement, and then forward the statement to the database.

---

### Note:

The application must tag a user request as trusted or not trusted, to enforce the WITH TRUST\_ONLY clause of the GRANT CONNECT THROUGH statement.

For detailed information on using SET QUERY\_BAND in a middle-tier application to facilitate trusted sessions, see *Teradata Vantage™ - SQL Data Definition Language Syntax and Examples*, B035-1144.

---

## Trusted Sessions and Space Usage

When a trusted session is established with a permanent proxy user:

- Perm space usage for the logged on user is charged to the permanent proxy user.
- The profile session attributes are set to those of the permanent proxy user.

When a trusted session is established with an application proxy user:

- The logged on user has no perm space.
- If a profile is specified on the GRANT CONNECT THROUGH statement for an application proxy user, profile session attributes are set to those of the specified profile. Attributes not specified in the profile are set to the values of the trusted user. If no profile is specified on the GRANT CONNECT THROUGH statement for an application proxy user, session attributes remain those of the trusted user.

For a permanent proxy user, if the user is logged on directly and runs queries and is also logged on to a middle tier with queries being run as a proxy, the SPOOL and TEMP usage accumulation includes both. The individual user's query fails when the collective usage exceeds the individual user's limits.

## About Session Processing for Trusted Sessions

After you set up trusted sessions, logons by end-users proceed as follows:

1. An application defined as a trusted user logs on to Teradata Vantage as a permanent database user, and creates a connection pool.
2. An application end user (defined as a proxy user) logs on and is authenticated by the application.
3. The proxy user requests a service that requires access to Vantage.
4. The application gets a connection from the pool and issues a SET QUERY\_BAND, which identifies the PROXYUSER and sets the role and query band duration, establishing a trusted session.
5. The database authorizes proxy user privileges in Vantage based on the role(s) assigned to the user.
6. Vantage records the proxy user identity in all access and query log entries.
7. A trusted session persists for the life of the query band.

## Security Considerations for Trusted Sessions

- The middle-tier application authenticates end users before it connects them to Teradata Vantage through a trusted session. Then Vantage controls access to database objects based on the proxy user role.
- Use the WITH TRUST ONLY clause in the GRANT CONNECT THROUGH to require that SET QUERY\_BAND statements be part of trusted requests.
- The system enforces logon controls, such as logons restrictions by IP address, only for the middle-tier application logon user (trusted user), because it does not authenticate proxy users.
- When a trusted session is established with a permanent proxy user, the permanent proxy user is the owner of and is granted default privileges on new objects.
- When a trusted session is established with an application proxy user, no automatic privileges are granted on new objects.
- The system enforces security policies based on the trusted user, not the end (proxy) user. For information on security policy, see [Network Security Policy](#).

- The system does not allow the SET ROLE statement in a trusted session. The operant role for a proxy user connection is determined by the roles you specify in the CONNECT THROUGH statement that defines the proxy user, along with any role limitations contained in the SET QUERY\_BAND statement submitted by the application.
- Construct the SET QUERY\_BAND statement to uniquely identify each end user so that the system can accurately log user sessions.

## Restricting Logons by Host Group

You can control access to Teradata Vantage for a large group of users by disabling logons for a host group associated with a set of connections to the database. The Gateway subsequently denies database access to clients that connect through the disabled host groups. The restriction does not affect clients that use other connections.

---

### Note:

The database defaults to a single host group, HGID 1.

---

1. A Teradata Customer Service representative configures PDE to define multiple host groups using the Vconfig utility. Each host group appears as a separate HGID in the vconfig.txt file.
2. A Teradata Customer Service representative configures the database to define multiple hosts using the Configuration utility ADD HOST command. Each host must include the same vprocs as the corresponding host group in Vconfig. You can verify the current host configuration with the LIST command.
3. The network administrator assigns multiple aliases (tdpids) to the Teradata Vantage system, and maps each tdpid to a set of COP names and IP addresses, which corresponds to a configured host group.
4. The network administrator assigns a Vantage client or group of clients to a single tdpid that corresponds to a host group.
5. You can disable a host group and tdpid without affecting clients assigned to other tdpids.
  - You can use the REVOKE LOGON statement to revoke all logons to a host group:

```
REVOKE LOGON ON hostid AS DEFAULT
```

where *hostid* corresponds to a host group (HostNo value).

You can limit the restriction to a user or comma-separated list of users with the clause FROM *userid*:

- You can use the Gateway Global utility to disable logons:
    - Use the SELECT HOST command to identify the host group to be disabled.
    - Use the DISABLE LOGONS command to disable logons through the selected host.
- 

### Note:

Make sure you understand the relationship between the host group (HostNo), tdpid, and networked clients so that you disable the correct host group.

---

## Using Host Group Restrictions

- By default, a client can log on to the database through all tdpids to which the client is connected.
- For Teradata Vantage host connections:
  - All COP connections assigned to the same tdpid should be in the same host group.
  - A LAN card can be part of more than one host group, but each host group must have a separate IP address. By default, each LAN card connects to only a single IP address.
  - Each LAN card allows multiple parallel sessions.
- Teradata recommends that you do not define individual IP address connections or COP connections in the Teradata ODBC driver or the client host file. These connections override DNS assignments and could compromise the setup for REVOKE LOGON ON *hostid*.

## Related Information

For information on...	See...
Setting up client host connections	<ul style="list-style-type: none"> <li>• <i>ODBC Driver for Teradata® User Guide</i>, B035-2526</li> <li>• <i>Teradata® Call-Level Interface Version 2 Reference for Workstation-Attached Systems</i>, B035-2418.</li> <li>• <i>Teradata® Director Program Reference</i>, B035-2416.</li> </ul>
Using the Gateway Global utility to enable or disable logons by hostid	For the Gateway Global utility see <i>Teradata Vantage™ - Database Utilities</i> , B035-1102
REVOKE LOGON syntax and options	<i>Teradata Vantage™ - SQL Data Control Language</i> , B035-1149.

## Restricting Logons by IP Address

Teradata Vantage provides the following methods for restricting database access by IP address:

- Create IP restrictions in an XML document or a directory and then transfer them to the IP restriction GDO. See the topics that follow this one.
- Create a security policy that defines IP restrictions. For details about configuration and use of policy IP restrictions, see [Network Security Policy](#).

IP restrictions apply to direct database logons, and logons through Unity. For logons through Unity, in addition to the logon user name and IP address, the Teradata Vantage gateway also receives the Unity user ID and IP address.

**Note:**

IP restrictions are not applicable to users who logon through middle-tier applications because the Teradata Vantage gateway does not see the originating IP address. The exception to this rule is Unity, which passes the client IP address to the gateway.

**Note:**

You can also set security policy for IP addresses in a network group. See [Network Security Policy](#).

**Link-local IP Addresses**

IPv6 and IPv4 link-local IP addresses are blocked from connecting to the database. During Teradata Vantage installation, an ipfilter is added to the ipfilter GDO restricting access to the link-local IP address range (fe80:: for IPv6 and 169.254.0 for IPv4).

The following ipfilter is added to ipfilter.xml to permit all IP addresses to connect to the database, except for blocked addresses in the listed ranges:

```
<ipfilter name="linklocal" type="permissive">
  <deny ip="fe80::/10"/>
  <deny ip="169.254.0.0/255.255.0.0"/>
  <appliedto tagref="allusers" />
</ipfilter>
```

Once the link-local restrictions are configured, backing down to an earlier release of Vantage will not remove the restrictions. If Link-local IP addresses are needed, they must be manually allowed.

If the upgrade or installation detects the customer is currently using ipfilters, the link-local restriction will not be imposed and a warning message will advise the customer to add the link-local restrictions manually.

To modify the link-local IP address configuration, see [Editing or Disabling IP Restrictions](#).

For information on how to configure IP restrictions, see [Creating XML-Based IP Restrictions](#).

**About GDO-Based IP Access Restriction**

You can define IP restrictions in:

- Teradata Vantage, by creating an XML IP document
- A supported directory, by configuring Teradata schema objects in the directory

**Note:**

You must use Teradata schema extensions to configure IP filter directory objects. Directories configured without Teradata extensions, as shown in [Using Native Directory Schema to Provision Directory Users](#), cannot use directory-based IP restrictions.

After defining the IP restrictions, you must transfer them to the IP restriction GDO.

The system applies IP restrictions to users based on:

- Filters that define allowed or denied IP addresses or address ranges.
- The users assigned to each filter.

The Teradata Vantage gateway screens each database logon and allows or denies the logon according to the IP restrictions in the GDO. If no IP restrictions exist, the database allows logons from any IP address to an authenticated user.

## Using IP Access Restrictions

- If any IP filter rejects a user, the user logon fails, even if all other filters allow the user.
- There is no limit to the number of IP restrictions concurrently in effect, but the database limits the size of the GDO that contains the limits to 128 KB, for both XML and directory implementations. If you plan IP restrictions carefully, the 128KB limit should be sufficient for most systems.
  - The GDO can contain dozens of filters and over 10,000 user names of 10 characters.
  - Companies with very large user bases can save GDO space by employing the directory-based implementation of IP restrictions and mapping multiple directory users to a smaller number of Teradata Vantage users that have the same access restrictions.
- Only a single set of restrictions, either XML or directory based, can exist at a time.
- To change the IP restrictions, revise the existing XML document or directory set up and then re-import the file into the GDO using the appropriate utility. The new restrictions overwrites the old GDO. See [Editing or Disabling IP Restrictions](#).
- You must perform a database restart to activate the initial IP restrictions. Subsequent changes to the restrictions do not require a restart. For more information, see the `tpareset` utility in *Teradata Vantage™ - Database Utilities*, B035-1102.
- Unity does not require a restart to see new or changed IP restrictions.
- Use of some applications, for example, network address translation (NAT) devices or other middle ware, prevents the Teradata Vantage gateway from seeing or restricting the user IP address. However, Unity passes IP addresses to the gateway for enforcement.
- If you add or alter an IP restriction that denies access to the IP address through which the user is already logged on, the pre-existing user session remains connected. The gateway denies the user access from that IP at the next logon, including a reconnect of the pre-existing session caused by a system restart.
- You can create IP restrictions for either IPv4 or IPv6 formatted IP addresses.

## Ensuring Security

The list of names and valid IP addresses in the XML file, or in the directory (if used), could represent a threat to your system. Only provide access to these files to trusted administrators.

## Creating XML-Based IP Restrictions

You can implement IP-based access restrictions for database users in an XML document.

Only one XML document can exist in Vantage at any time. If you create and import a second document, it nullifies the effects of the first document.

---

### Note:

Instead of using the XML document method, you can create IP restriction objects in a supported directory, but the directory-based restrictions only apply to database user objects. See [Creating IP Restrictions in a Directory](#).

---

## Prerequisites

All users referenced in an IP XML restriction document must exist in the database, and in a Unity environment; the same users must exist on all connected database systems.

## Implementation Process for XML-Based Restrictions

1. Review the IP restriction elements. See [Designing IP XML Restrictions](#).
2. Design IP restrictions, that is, filters and masks, according to your needs.
3. Create the restriction document. See [Creating an IP XML Restriction Document](#).
4. Save the completed restriction document. See [Saving a Completed XML IP Restriction Document](#).
5. Test the XML document to make sure the restrictions operate as needed. See [Testing XML-Based IP Restrictions](#).
6. Enable the restrictions by importing them into the IP GDO or bin file (Unity). See [Enabling XML-Based IP Restrictions with the ipxml2bin Utility](#).

---

### Note:

If users log on to through Unity, the IP restrictions must be configured exactly the same on all connected Vantage systems. No additional configuration is required on Unity.

---

## Designing IP XML Restrictions

You can use any text editor, for example vi or Notepad, to create an IP XML restriction document, but you must use a valid structure to construct the restrictions.

- Copy the examples provided and customize them for your site needs.
- Templates for creating IP filter XML documents are located in /opt/teradata/tdgss/etc/ipfilter.xml.

## About IP XML Documents

You can define IP restriction parameters in an XML document by specifying the necessary IP restriction elements and element values.

### Example: IP Restriction Elements

```
<tdat name="tdat">
  <system name="gizmo">
    <users>
      <user name="drct01" tag="xyzyz"/>
      <user name="perm01" tag="noside"/>
      <user name="extuser" tag="shazam"/>
    </users>
    <ipfilters>
      <ipfilter name="filter1" type="restrictive">
        <allow ip="141.206.0.0/255.255.0.0"/>
        <deny ip="141.206.35.0/255.255.255.0"/>
        <appliesto tagref="xyzyz"/>
        <appliesto tagref="shazam"/>
      </ipfilter>
      <ipfilter name="filter2" type="permissive">
        <deny ip="141.206.35.0/255.255.255.0"/>
        <allow ip="141.206.35.175/255.255.255.255"/>
        <appliesto tagref="noside"/>
        <appliesto tagref="xyzyz"/>
      </ipfilter>
    </ipfilters>
  </system>
</tdat>
```

## About IP Restriction Elements

The IP restriction hierarchy is similar whether you define the restrictions as elements in an IP XML document or as the objects and attributes in a directory information tree (DIT).

For information on how the comparable attributes appear in a directory-based implementation of IP restrictions, see [Creating IP Restrictions in a Directory](#).



## **tdat**

The tdat root element defines a name attribute, and specifies that the XML IP restriction document applies to Teradata Vantage systems and users. There are no restrictions on the name value for this attribute.

## **system**

The system element defines a name attribute that identifies the Vantage system to which you want to restrict access. A system element must always appear as a child of the tdat element.

The system element defines a name attribute. In the interest of simplicity you can use the cn of the Vantage system for this name attribute, but this approach is not required.

If the restriction document contains multiple system elements, the gateway uses only the first system element in the document.

## **users**

The users element must appear at the beginning and end of the list of Vantage users that are affected by the filters defined in the document. You can add user names to the list, each with a separate users element, however, the users element does have to contain individual users. If you do not specify any user elements as children of the users element, the filter applies to all users. See [Applying a Filter to All Users](#).

The users element must appear as a child of a system element. The XML document can contain as many users elements as needed, but only one appearance is typical.

## **user**

A user element describes a Vantage user that exists in the database, and that is subject to the IP restrictions. Each user element bears a name attribute and a tag attribute.

- The value of the name attribute must be a Vantage username, including end users, trusted user applications, and the Unity user.
- The value of the tag attribute must be a non-colonized XML name, unique to the entire XML document.

You can specify the value of a tag attribute in an IP filter appliesto tagref attribute, to apply the filter to the user that corresponds to the tag.

## **ipfilters**

An ipfilters element marks the beginning and end of a list of individual IP filters. Each filter in the list is represented by an ipfilter element.

The `ipfilters` element must be the child of a `system` element. You can use as many `ipfilters` elements as needed in the XML IP restriction document, although only one list of filters is typical.

## ipfilter

The `ipfilter` element corresponds to an IP filter. Each `ipfilter` has two attributes, `name` and `type`.

- The `ipfilter` name must be unique, to differentiate one filter from another.
- The `type` attribute can contain a value of either `permissive` or `restrictive`, depending on the intended function of the filter. See [About Permissive Filters](#) and [About Restrictive Filters](#).

The `ipfilter` element must be a child of an `ipfilters` element. You can use as many `ipfilter` elements as necessary to define the active IP filters in an XML IP restriction document.

## allow

The `allow` element defines an allowed IP address or range of addresses, and contains an `ip` attribute, composed of an `ip` value and a mask value, separated by a slash (/).

The Teradata Vantage gateway ANDs each incoming IP address with the `allow` mask. Then it ANDs the `ip` value in the `allow` element with the mask. If the results of the two ANDs match, the gateway allows the incoming IP address to access the database.

The `allow` element can appear only as a child of an `ipfilter` element. You can use as many `allow` elements as needed in an XML IP restriction document. When multiple `allow` elements appear in a document, only one match is necessary to allow the session to proceed.

## deny

The `deny` element defines a denied IP address or range of IP addresses, and contains an `ip` attribute, composed of an `ip` value and a mask value, separated by a slash (/).

The Teradata Vantage gateway ANDs each incoming IP address with the `deny` mask. Then it ANDs the `ip` value in the `deny` element with the mask. If the results of the two ANDs match, the gateway denies the incoming IP address to access the database.

The `deny` element can appear only as a child of an `ipfilter` element. You can use as many `deny` elements as needed in the XML IP restriction document. When multiple `deny` elements appear in a document, only one match is needed to deny the incoming IP address.

## applies to

The `applies to` element contains the attribute `tagref`, which links an IP filter to a user element that represents a Vantage user. The `tagref` value must exactly match the value of the `tag` attribute for the user element, or the filter does not affect the user.

The applies to element can appear only as a child of an ipfilter element.

## Working with Restrictions on XML Attribute Name Values

When you specify certain element attributes in an IP XML document, the attribute values must conform to XML rules.

### Note:

There is no length restriction for an attribute value.

Attribute	Restrictions
<user name>	An XML string name that conforms to the following: <ul style="list-style-type: none"> <li>Any character is allowed</li> <li>You must escape characters that have special functions in XML, for example, you must express:               <ul style="list-style-type: none"> <li>" (double-quotation marks) as &amp;quot</li> <li>&amp; (ampersand) as &amp;amp</li> <li>&lt; (less-than) as &amp;lt</li> </ul> </li> </ul> The value of a user name attribute must correspond to an existing Teradata Vantage username.
<ip filter name>	
<ul style="list-style-type: none"> <li>&lt;tag&gt;</li> <li>&lt;tagref&gt;</li> </ul>	A non-colonized XML name, which can contain only: <ul style="list-style-type: none"> <li>Letters A through Z and/or a through z</li> <li>Digits zero through 9</li> <li>_ (low line)</li> <li>- (hyphen)</li> <li>. (period)</li> <li>Ideographs</li> </ul> You cannot start a name with a digit, hyphen, or period.

## About IP Filters

IP filters are the active components in an XML IP restriction document. They identify IP addresses subject to restriction. Review the effects of the filter elements and attributes, and how they work together, before attempting to create an IP filter.

### Note:

The IP addresses shown are IPv4 format, but the system also accepts IPv6 formatted filters and masks.

## Example: IP Filter

```
<ipfilter name="filter1" type="restrictive">
  <allow ip="141.206.0.0/255.255.0.0"/>
  <deny ip="141.206.35.0/255.255.255.0"/>
  <appliesto tagref="xyzyz"/>
</ipfilter>
```

where:

Filter Component	Description
<code>&lt;ipfilter name="filter1"</code>	Each filter definition begins (and ends) with an ipfilter element, which must be the child of an ipfilters element. The ipfilter element at the beginning of the filter definition must contain the name attribute and a corresponding value. In this case, the name is filter1.
<code>type="restrictive"&gt;</code>	Identifies the type of filter as either permissive or restrictive, and determines how the system processes the filter allow and deny elements. See <a href="#">About Permissive Filters</a> and <a href="#">About Restrictive Filters</a> .
<code>&lt;allow ip="141.206.0.0/255.255.0.0"/&gt;</code>	A restrictive filter must use an allow element to define all the IP addresses that the filter allows to log on. The <i>allow</i> element is composed of the: <ul style="list-style-type: none"> <li>• Allowed IP address range, in this example, 141.206.0.0</li> <li>• Mask, 255.255.255.0, which defines how much of the allow IP address the system considers when determining which IPs it allows to logon.</li> </ul> For information about how the allow and deny elements affect both permissive and restrictive filters, see <a href="#">Working with the Effects of Filter Type on allow and deny Elements</a> . For more information about masking, see the topics beginning with <a href="#">About IP Addresses and Mask Structure</a> .
<code>&lt;deny ip="141.206.35.0/255.255.255.0"/&gt;</code>	Restrictive filters can optionally use a deny element to define exceptions to the range of addresses specified in the allow element. The deny element is composed of the: <ul style="list-style-type: none"> <li>• Deny IP address or address range, 141.206.35.0/</li> <li>• Mask, 255.255.255.0, which defines how much of the IP address the Teradata Vantage gateway uses to determine if a logon is allowed.</li> </ul> For information about allow and deny element function within a filter, see <a href="#">Working with the Effects of Filter Type on allow and deny Elements</a> . For information on masking, see the topics beginning with <a href="#">About IP Addresses and Mask Structure</a> .

Filter Component	Description
<code>&lt;appliesto tagref="xyzy" /&gt;</code>	The tagref value in each applies to element links the filter to the user element with a matching tag attribute value. In this case, the value is xyzy. This link applies the rules for the IP filter to the user. You can use an appliesto element for each user to which the IP filter applies. For more information on applying IP filter effects to all users, see <a href="#">Creating an IP XML Restriction Document</a> and <a href="#">Applying a Filter to All Users</a> .
<code>&lt;/ipfilter&gt;</code>	This element defines the end of the IP filter definition.

## Working with the Effects of Filter Type on allow and deny Elements

IP filters often contain both an allow and deny elements, although use of both elements is not required. The first element in the filter specifies the range of IP addresses to which the IP filter applies. The second element defines exceptions within that range to which the filter does not apply. The filter type determines which element is the primary and which is the exception.

Consider the differences in function of the allow and deny elements between the restrictive filter in [Example: IP Filter](#), and the following permissive filter:

```
<allow ip="141.206.0.0/255.255.0.0"/>
<deny ip="141.206.35.0/255.255.255.0"/>
```

The following table compares element function within the two filter environments.

Filter Type	Element	Function
Restrictive	allow	Allows access to the specified IP address or address range. The true range depends on both the IP address and the mask.
	deny	Defines an exception to the address range specified in the allow element. This exception denies access to a specified IP address or address range that is a subset of the allowed address range.
Permissive	deny	Denies access to the specified IP address or range of addresses.
	allow	Defines an exception to the address range specified in the deny element. This exception allows access to a specified IP address or address range that is a subset of the denied address or address range.

## About IP Addresses and Mask Structure

The allow and deny elements define the IP addresses or address ranges to which an IP filter applies. The address or address range must appear together with a mask as a single allow or deny element masked IP value.

---

### Note:

The masking procedures and examples shown here are based on IPv4 addresses, but are also applicable to IPv6 addresses.

---

### Example: Allow IP

```
<allow ip="141.206.35.0/255.255.255.0"/>
```

Within the string, the four decimal-separated segments preceding the first slash (/) represent the IP address or range. Each segment contains a number from 0 to 255. The mask is the four decimal-separated segments that follow the first slash (/).

### Masking Effects on an Incoming IP Address

When the Teradata Vantage gateway ANDs the mask with the IP, the result acts as an 8-bit filter that tests the IP source addresses of incoming logons.

The mask tells the filter which part of an IP address or range is important, and to what extent it must test an incoming IP address against IP restrictions in the allow and deny elements. If an element does not define a mask, the masking defers to the value 255.255.255.255, meaning that the incoming IP must match the filter IP exactly or the filter has no effect.

In the example [Example: Allow IP](#), the mask uses the value 255 in the first three decimal-separated segments (24 bits) to instruct the filter to consider the entire value of each of the corresponding segments of the IP. The segments are binary, and the 8 bits represent (from right to left) the first eight values in the binary sequence, 1, 2, 4, 8, 16, 32, 64, and 128, for a total value of 255.

To consider only part of a binary IP string, you can use a mask similar to:

```
255.255.192.0"/>
```

The gateway applies the masking values to the binary string from right to left. A value of 192 means that the mask considers the 2 left positions of the third binary segment, 128 and 64, which total 192.

**Note:**

Partial segment masking can have complex effects on filter function. Before you use this type of masking, see [Masking Partial Binary IP Segments](#).

You can also use an alternate form of masking that expresses the mask as the number of binary bits (from left to right in the binary string) that the restriction must consider. Using the bit method, the 255.255.255.0"/> becomes 24"/>, or 3 decimal-separated, 8-bit segments.

## Applying a Mask to a Filter

When an IP filter encounters an incoming IP address during a logon, it uses the following process to determine whether or not the IP address is allowed access to Teradata Vantage.

**Note:**

The example process is based on a typical allow element in a restrictive filter. If the filter also contains a deny element, it continues evaluation of the incoming IP until it also applies the deny parameters, which represent exceptions to the allow.

1. Convert the specified IP in the primary element, for example, the allow element IP 141.206.35.0 in a restrictive filter, to a binary string:

```
10001101.11001110.00100011.00000000
```

2. Convert the primary element mask, for example 255.255.255.0, to a binary string.
3. AND the binary string representing the allow element IP with the mask, to obtain the allow result (shown in **bold**):

```
10001101.11001110.00100011.00000000
11111111.11111111.11111111.00000000
-----
10001101.11001110.00100011.00000000
```

4. Examine the incoming IP address and convert it to binary format. For example, convert the incoming IP address 141.206.35.62 to the following binary string:

```
10001101.11001110.00100011.00111110
```

5. AND the binary incoming IP address with the allow element mask to obtain the incoming IP result (shown in **bold**):

```
10001101.11001110.00100011.00111110
11111111.11111111.11111111.00000000
-----
10001101.11001110.00100011.00000000
```

6. Compare the binary incoming IP result with the allow element IP result (for this example, they are equal).

A filter has an effect on an incoming logon only if both of the following are true:

- The incoming IP result matches the allow result.
- The username in the logon appears in the appliedto element of the filter.

---

**Note:**

The filter continues to test the incoming IP address against the secondary parameters, in this case, the deny portion of the filter. If the secondary testing denies the logon, it fails, even if the primary testing allows the logon.

---

## Masking Partial Binary IP Segments

The effects of simple masking are clear and easy to understand, for example, the IP/mask 141.206.0.0/16 in an allow element allows IP address 141.206.35.62 to log on because:

- Zeros in the third and fourth decimal segments of the IP indicate that those segments are not significant to determining the IPs that are allowed.
- The mask /16 indicates that only the first 16 bits of the allowed IP range must match exactly for the allow filter to function.

If you need to use more complex masking, Teradata recommends that you do a detailed masking analysis to fully understand the effects before you implement the restriction.

## Example: Complex Mapping

A company must use the IP/mask 141.206.0.0/13 to restrict all employees of certain departments from accessing the database. This mask, with a value not divisible by 8, includes many additional IP addresses beyond the 255 x 255 addresses represented by the zeros in segments three and four, because it also partially masks segment two.

The following masking analysis helps explain the effect of a partial segment mask on the content of the top level subnet address:

- AND the binary values of the subnet address with those of the mask:

```
10001101.11001110.00000000.00000000 (141.206.0.0)
11111111.11111000.00000000.00000000 (255.248.0.0 or /13)
-----
10001101.11001000.00000000.00000000 (141.200.0.0)
```

- The result shows the first 13 digits in bold text to indicate that they must be present in any address allowed by the allow element. Note that the first 13 digits of the result match the first 13 digits of the original range. The remaining 19 digits appear in normal text to indicate that they can be either a zero or a 1 and still be part of the subnet.



- Expressing all 19 digits as 1, while retaining the first 13 digits as shown in bold, results in the largest possible address in this subnet, or 10001101.11001111.11111111.11111111 (141.207.255.255).
- The total range of addresses in subnet 141.206.0.0/13 includes all addresses from 141.200.0.0 through 141.207.255.255.

To apply partial segment masking to IP filters, see [Example: Secondary Element Processing—Single Address Exception](#) and [Example: Secondary Element Processing—Carve Out Exception](#).

## Understanding Interactions Between Primary and Secondary Masked IPs

An IP filter can contain both an allow and a deny element, although it is not required. The interaction between the two element types (and their masks) depends on the type of filter they inhabit. The filter type determines which element the system processes first (the primary), as shown in [Working with the Effects of Filter Type on allow and deny Elements](#). The primary element determines the basic rules by which the filter operates. The secondary element defines the exceptions to those rules.

The examples that follow show the interaction between allow and deny elements.

---

### Note:

These examples are for a restrictive filter. Masking principles for a permissive filter are similar, but the filter tests allow and deny elements in the opposite order.

---

## Example: Primary Element Processing

The Teradata Vantage gateway processes the primary filter element first and defines the rule the filter uses to evaluate incoming IP addresses. The primary element specifies a range of IP addresses.

- In a restrictive filter, the allow element is the primary. Suppose the allow element allows the following range of IP addresses:

```
<allow ip="141.206.35.0/
```

Note that the allow element contains a zero for the last segment rather than specifying each allowed address within the subnet.

If you specify this value for the element, it indicates that the filter allows any IP address in the 141.206.35 subnet, possibly a department within a large company.

- A user attempts to access the database from the incoming IP address:

```
141.206.35.175
```

- The allow element includes the following mask, which it uses to test an incoming IP:

```
255.255.255.0"/>
```

The allow element mask has a zero in the fourth segment, so it tests only the first three segments of any incoming IP address. Since the first three segments of the mask have values of 255, the corresponding segments of the allow element and incoming IP address must match exactly to allow the logon. The first three segments match, the logon succeeds.

---

**Note:**

The allow element achieves the same restriction capability if you express the mask as 24"/>.

---

Filtering is not complete at this point if the filter also contains a deny element, which the gateway must also consider.

## Example: Secondary Element Processing—Single Address Exception

After considering the primary element, the Teradata Vantage gateway considers the secondary element, which represents an exception to the filter rule stated in the primary. In the example below, the secondary element specifies an individual address, contained within the range defined by the primary element, to exempt the address from the allow.

- In the following example, a secondary deny element denies a single IP address from within the range of the primary allow element. This address could be a training computer that should not have direct access to the database.

```
<deny ip="141.206.35.175/
```

- You can use the following mask to ensure that the filter tests all 32 bits of the IP address to enforce the deny restriction.

```
255.255.255.255"/>
```

The deny processing for the incoming IP address denies access even though the allow element allows it. The mask format indicates that all 32 bits of the address are significant. The format is necessary because the denied IP address is unique only in the fourth decimal segment.

---

**Note:**

The allow element achieves the same restriction capability if you express the mask as 32"/>

---

## Example: Secondary Element Processing—Address Range Exception

The secondary element can also specify an exception for a range of IP addresses that are contained within the larger range defined by the primary element.

- Instead of a single IP address exception, you can deny access to IP addresses for several computers in the company, for example, work stations 141.206.35.192 through 141.206.35.255, with the following deny element:

```
<deny ip="141.206.35.255/
```

The deny element is equivalent to the following binary number:

```
10001101.11001110.00100011.11111111
```

The 255 in the final segment of the deny IP is optional. You can use any number between 192 and 255 to give the same results, based on the mask construction shown in the following bullet.

- The following mask forces the filter to deny access to all workstations with IP addresses from 141.206.35.192 through 141.206.35.255:

```
255.255.255.192"/>
```

This mask format indicates that only the last two bits of the fourth segment are significant. If you AND the binary values for the deny IP and the mask, the result shows why you can specify such a wide range of addresses in the forth segment of the deny IP.

Deny IP	10001101.11001110.00100011.11111111
Mask	11111111.11111111.11111111.11000000
Result	<b>10001101.11001110.00100011.11000000</b>

The mask is equivalent to /26">, and indicates that the first 26 bits (the bold characters in the result) of the incoming IP address must match the masked deny IP to access to deny the incoming IP address. All IP addresses from 141.206.35.192 through 141.206.35.255 match the bold characters. IP addresses from 141.206.35.1 through 141.206.35.191 have a value of zero for either bit 25 or 26 (or both), do not match all 26 significant binary values, and therefore are not denied.

The restriction process applies the range of the secondary element, whether it is an allow or a deny, to the binary string from left to right, that is, high to low address. The further to the left you extend the zeros in the mask, the more restrictive the secondary deny. For example, a partial mask of the third segment significantly increases the range addresses affected.

## Example: Secondary Element Processing—Carve Out Exception

You can also use the secondary element to specify an exception for a range of IP addresses that are in the midst (rather than at the end) of a larger range defined by the primary element, to carving out a list of denied addresses. This approach also requires use of partial segment masking.

- You can deny access to IP addresses 48 through 63 (instead of the IP addresses 192 through 255 shown in [Example: Secondary Element Processing—Address Range Exception](#)) using the following deny element:

```
<deny ip="141.206.35.48/28"/>
```

This deny element is equivalent to the following binary IP and mask:

```
10001101.11001110.00100011.00110000 (141.206.35.48)
11111111.11111110.11111111.11110000 (255.255.255.240 or /28"/>)
_____1000110
1.11001110.00100011.00110000 (141.206.35.48)
```

- The result of ANDing the IP and mask indicates (in bold) that only the first four bits of the fourth segment are considered for denial within the 141.206.35 subnet. These bits must match exactly for an address to be denied. The deny element has the following effects:
  - Addresses 141.206.35.64 through 141.206.35.255 all have a value of 1 for one or both of the first two bits, which causes them to not match the deny mask, and not be denied.
  - Addresses 141.206.35.1 through 141.206.35.47 have a zero value for either the third or fourth bit, and are also not denied.
  - Addresses 141.206.35.48 through 141.206.35.63 all have 0011 for the first four bits, and are denied.

---

**Note:**

Because the restriction processes masks from left to right, in binary notation, you cannot isolate some addresses with a carve out type of mask, for example, 141.206.35.51 through 141.206.35.62. Carve out exceptions always affect all addresses that contain the mask value of the IP filter.

---

## Related Information

For information on...	See...
Using elements and masks to create filters	<ul style="list-style-type: none"> <li>• <a href="#">About Permissive Filters.</a></li> <li>• <a href="#">About Restrictive Filters.</a></li> </ul>
Placing filters in an XML IP restriction document	<a href="#">Creating XML-Based IP Restrictions.</a>

## About Permissive Filters

A permissive filter without a deny element permits logons from all IPs regardless of which IPs are explicitly allowed by the allow element. You can use permissive filter deny elements to define denied IPs for a list of users, and then optionally use an allow element to enable some IPs within the denied range.

The Teradata Vantage gateway first processes permissive filter deny elements, and then processes the allow elements. As a result, the gateway denies any IP address listed in the deny element unless it also appears in an allow element, in which case the gateway allows the IP to access the database.

## Gateway Processing of Permissive Filters

1. The Teradata Vantage gateway processes each incoming IP address against the permissive filter deny element.
  - a. The filter masks the incoming IP address under test with the mask from the deny element.
  - b. The filter masks the IP address in the deny element with the same mask.
  - c. If the two masked IP addresses match, the filter identifies the IP address under test as a candidate for denial. The filter then ends the deny phase of testing.
2. The gateway does allow-testing only if deny-testing identifies an IP address as a candidate for denial. If allow-testing does not override the denial, the gateway rejects the IP.
  - a. The filter masks the incoming IP address under test with the mask from the allow element.
  - b. The filter masks the IP address in the deny element with the same mask.
  - c. If the two masked IP addresses match, the filter allows the IP address under test to access the database and then ends the allow phase of testing.

### Example: Permissive Filter

```
<ipfilter name="filter2" type="permissive">
  <deny ip="141.206.35.0/255.255.255.0"/>
  <allow ip="141.206.35.175/255.255.255.255"/>
  <appliesto tagref="samoht"/>
  <appliesto tagref="noside"/>
</ipfilter>
```

where:

Term	Description
ipfilter name="filter2"	The filter name. Uniquely identifies the filter.
type="permissive"	The filter type. This term identifies whether the filter is permissive or restrictive and indicates the order of IP testing (deny and allow) that it can perform on an incoming IP address.
<deny ip="141.206.35.0/255.255.255.0"/>	The deny element appears first in a permissive filter. The deny element is divided into two segments, separated by a slash (/): <ul style="list-style-type: none"> <li>• The filter: &lt;deny ip="141.206.35.0/</li> <li>Denies access to the database from any IP address within the 141.206.35 subnet, unless the allow element explicitly allows the address. The filter allows access to all IPs not covered in the deny element.</li> <li>• The mask: 255.255.255.0"/&gt;</li> </ul>

Term	Description
	<p>Determines the extent to which the filter tests an incoming IP address against restrictions defined in the deny element. A mask of 255.255.255.0"/&gt; is equivalent to a mask of 24"/&gt;. It tests the first 24 bits (all but the last decimal segment) of the IP address.</p> <p><b>Note:</b> You can use the deny element in a permissive filter to specify a higher network tree level than what you specify in the allow element.</p>
<pre>&lt;allow ip="141.206.35.175 /255.255.255.255" /&gt;</pre>	<p>The allow element must appear after the deny element in a permissive filter.</p> <p>The allow element is divided into two segments, separated by a / :</p> <ul style="list-style-type: none"> <li>• The filter: &lt;allow ip="141.206.35.175/ Explicitly allows the 141.206.35.175 IP address access to the database, even though it is within the subnet denied access by the deny element. The filter denies access to any other IP addresses that appear in the deny element.</li> <li>• The mask: 255.255.255.255"/&gt; Determines the extent to which the filter tests an incoming IP address against restrictions defined in the allow element. A mask of 255.255.255.255 tests all the decimal segments of the IP address for an exact match.</li> </ul> <p><b>Note:</b> You can use the allow element in a permissive filter to specify a lower level in the network tree than what you use for the deny element, to allow exceptions to the IPs that the filter explicitly denies. If necessary, you can use multiple allow elements to define the exceptions.</p>
<pre>appliesto tagref="samoh"</pre>	<p>Identifies a user affected by this set of filter rules.</p> <p>Each appliesto tagref value must correspond to a tag attribute for an individual Vantage user listed in a user element of the XML IP restriction document.</p>
<pre>appliesto tagref="noside"</pre>	<p>Identifies a second user affected by this set of filter rules.</p>

## About Restrictive Filters

A restrictive filter type denies all incoming IPs except those specifically allowed by the allow element. A restrictive filter without an allow element denies logons from all IPs regardless of which IPs are explicitly denied by the deny element.

You can use restrictive filters to define the allowed IPs for a list of users, while retaining the ability to use the deny element to disable some of the allowed IPs.

## Gateway Processing of Restrictive Filters

1. The Teradata Vantage gateway tests the incoming IP address against each allow element.
  - a. The filter masks the incoming IP address with the mask from the allow element.
  - b. The filter masks the IP address in the allow element with the same mask.
  - c. If the result for the two masked IP addresses match, the filter identifies the IP address under test as a candidate for approval. The filter then ends the allow phase of testing and begins deny testing of the incoming IP address.
2. The gateway tests the incoming IP address against each deny element.
  - a. The filter masks the incoming IP address under test with the mask from the deny element.
  - b. The filter masks the IP address from the deny element with the same mask.
  - c. If the result for two masked IP addresses match, the filter allows the IP address under test to access the database and ends the denial phase of testing.

---

### Note:

Although you can construct a restrictive filter using both allow and deny elements, you do not have to use both elements in a restrictive filter. A restrictive filter must contain at minimum either an allow or a deny element. If you use only a single element, it should be the primary element type for the filter type, that is, an allow element in a restrictive filter.

---

## Example: Restrictive Filtering

```
<ipfilter name="filter1" type="restrictive">
  <allow ip="141.206.0.0/255.255.0.0"/>
  <deny ip="141.206.35.0/255.255.255.0"/>
  <appliesto tagref="xyzyzy"/>
  <appliesto tagref="shazam"/>
</ipfilter>
```

where:

Term	Description
ipfilter name="filter1"	The unique name of an IP filter.
type="restrictive"	The filter type. This term identifies whether the filter is a restrictive or permissive type, and indicates the order in which testing takes place when the filter evaluates an incoming IP address.

Term	Description
<code>&lt;allow ip="141.206.0.0/255.255.0.0"/&gt;</code>	<p>The allow element appears first in a restrictive filter.</p> <p>The allow element is divided into two segments, separated by a / :</p> <ul style="list-style-type: none"> <li>• The filter: <code>&lt;allow ip="141.206.0.0/</code> Allows users to access the database from any IP address within the 141.206 subnet, unless the address explicitly appears in the deny element that follows. The filter denies access to all IPs not included in the allow element.</li> <li>• The mask: <code>255.255.0.0"/&gt;</code> Determines the extent to which the filter tests an incoming IP address against access allowed in the allow element. A mask of 255.255.0.0 tests the first two decimal segments of each IP address seeking access, to determine whether or not it falls within the allowed range.</li> </ul> <p><b>Note:</b> You can use the allow element in a restrictive filter to specify a higher level in the network tree than what you use for the deny element.</p>
<code>&lt;deny ip="141.206.35.0/255.255.255.0"/&gt;</code>	<p>The deny element appears second in a restrictive filter.</p> <p>The deny element is divided into two segments, separated by a / :</p> <ul style="list-style-type: none"> <li>• The filter: <code>&lt;deny ip="141.206.35.0/</code> Explicitly denies all addresses in the 141.206.35 subnet, even though it is within the range of IPs allowed by the allow element. The filter denies access to other IP addresses if they appear in a subsequent deny element.</li> <li>• The mask: <code>255.255.255.0"/&gt;</code> Determines the extent to which the filter tests an incoming IP address against access denied in the deny element. A mask of 255.255.255.0 tests the first three decimal segments of each IP address seeking access, to determine whether or not it falls within the denied range.</li> </ul> <p><b>Note:</b> You can use the deny element in a restrictive filter to specify a lower level in the network tree than you use for the allow element, to define exceptions to the IPs explicitly allowed in the allow element. You can use multiple deny elements, if necessary.</p>
<code>&lt;appliesto tagref="xyzzzy"/&gt;</code>	<p>Identifies a user affected by this set of filter rules.</p> <p>The appliesto tagref values must correspond to tag attributes assigned to individual users listed in user elements of the XML IP restriction document.</p>
<code>&lt;appliesto tagref="shazam"/&gt;</code>	<p>Identifies a second user affected by this set of filter rules.</p>

## Creating an IP XML Restriction Document

After you design the needed IP filters, you can use them to create an XML restriction document.



**Note:**

The examples in the following procedure use the filters created in [Designing IP XML Restrictions](#).

1. Open a text editor, such as vi or Notepad.
2. Create the framework for the XML document, specifying the required information and element tags, using this syntax:

```
<?xml version="xml_version" encoding="encoding"?>
<tdat name="tdat">
  <system name="system_name">
    <users>
      user [...]
    </users>
    <ipfilters>
      primary_filter_definition
      secondary_filter_definition
    </ipfilters>
  </system>
</tdat>
```

***user***

```
<user name="user_name" tag="user_tag"/>
```

***primary\_filter\_definition***

```
<ipfilter name="primary_filter_name" type="restrictive">
  <allow ip="primary_filter_ip_range"/>
  <deny ip="primary_filter_deny_range"/>
  <appliesto tagref="user_tag"/> [...]
</ipfilter>
```

***secondary\_filter\_definition***

```
<ipfilter name="secondary_filter_name" type="permissive">
  <deny ip="secondary_filter_deny_range"/>
  <allow ip="secondary_filter_ip_range"/>
  <appliesto tagref="user_tag"/> [...]
</ipfilter>
```

***xml\_version***

Indicates the version of XML you are using to generate the document. This specification is for reference only. Example: 1.0

***encoding***

Defines the character set you are using in the XML document. Example: UTF-8

***tdat***

Specifies the name of XML document root element. Example: tdat. See [tdat](#).

***system\_name***

Specifies the name of the system to which the IP restrictions apply. The name must correspond to the tdpid that affected users specify when they log on to the database. Example: gizmo. See [users](#).

***user***

User to which the restrictions in the XML document apply. Examples in a later step. See [users](#).

***primary\_filter\_definition***

Filter definition that defines the restrictions in the XML restrictions. Example in a later step.

***secondary\_filter\_definition***

Filter definition that defines the restrictions in the XML restrictions. Example in a later step.

***user\_name***

Vantage username. Examples in a later step.

***user\_tag***

An XML document tag that links the corresponding Vantage username to an IP filter, when the tag value appears in the appliesto tagref attribute of the filter.

Each appliesto tagref value must correspond to a tag attribute for an individual Vantageuser listed in a user element in the document.

Examples in later steps.

***primary\_filter\_name***

Specifies the name of the primary filter listed in the restriction document, a restrictive filter. Example: filter1.

***primary\_filter\_ip\_range***

Specifies the IP range allowed by *primary\_filter\_name*.

Example: 141.206.0.0/255.255.0.0

- 141.206.0.0/ allows access to the database for any IP addresses within the 141.206 subnet unless they explicitly appear in *primary\_filter\_deny\_range*.

Because the filter is restrictive, it automatically denies access to all IPs outside those specified in the allow element.

- 255.255.0.0 defines the allow element mask.

The zeros in the third and fourth segments cause the filter to test only the first 16 bits of the incoming IP address against the allowed IP.

***primary\_filter\_deny\_range***

Specifies the range of the primary deny filter.

Example: 141.206.35.0/255.255.255.0

- 141.206.35.0/ denies access to IPs in the 141.206.35 subnet, even though the subnet is within the 141.206 range specified in *primary\_filter\_ip\_range*.
- 255.255.255.0 defines the deny element mask.

This mask causes the filter to test the first 24 bits of an incoming IP address against the denied IP. The zero indicates the filter does not use the last 8 bits of an incoming IP address in deny-testing.

***secondary\_filter\_name***

Specifies the name of the secondary filter listed in the restriction document, a permissive filter. Example: filter2

***secondary\_filter\_deny\_range***

Specifies the range of the secondary deny filter.

Example: 141.206.35.0/255.255.255.0

- 141.206.35.0/ denies access to the database for any IP addresses within the 141.206.35 subnet unless they explicitly appear in *secondary\_filter\_ip\_range*.

Because the filter is permissive, it allows access to all other IPs outside those specified the deny elements.

- 255.255.255.0 defines the deny element mask.

The 255 in each decimal-separated segment of the mask indicates the filter tests the corresponding segment of the IP address for access denial.

This mask causes the filter to test the first 24 bits of the incoming IP address against the denied IP.

### ***secondary\_filter\_ip\_range***

Specifies the IP exceptions allowed by *secondary\_filter\_name*.

Example: 141.206.35.175/255.255.255.255

- 141.206.35.175/ allows access to the database for IP address 141.206.35.175 even though it is otherwise disallowed by the more general parameters of the deny element.
- 255.255.255.255 defines the allow element mask.

The 255 in each decimal-separated segment of the mask indicates that the filter tests the corresponding segment of the IP address for allowed access.

This mask causes the filter to test all 32 bits of the incoming IP address against the allowed IP.

3. Add each user that is affected by the restrictions. For example:

```
<users>
  <user name="drct01" tag="xyzyz"/>
  <user name="perm01" tag="noside"/>
  <user name="extuser" tag="shazam"/>
</users>
```

4. Add the IP filters that define the IP restrictions for all users. For example:

```
<ipfilters>
  <ipfilter name="filter1" type="restrictive">
    <allow ip="141.206.0.0/255.255.0.0"/>
    <deny ip="141.206.35.0/255.255.255.0"/>
    <appliesto tagref="xyzyz"/>
    <appliesto tagref="shazam"/>
  </ipfilter>
  <ipfilter name="filter2" type="permissive">
    <deny ip="141.206.35.0/255.255.255.0"/>
    <allow ip="141.206.35.175/255.255.255.255"/>
    <appliesto tagref="noside"/>
    <appliesto tagref="xyzyz"/>
  </ipfilter>
</ipfilters>
```

The primary filter, filter1, applies to users drct01 and extuser, because it specifies their user tags, xyzyz and shazam.

The secondary filter, filter2, applies to users perm01 and drct01, because it specifies their user tags, noside and xzyzy.

## Example: Completed IP XML Restriction Document

The following example applies two IP filters to individual Teradata Vantage users.

```
<?xml version="1.0" encoding="UTF-8"?>
<tdat name="tdat">
  <system name="gizmo">
    <users>
      <user name="drct01" tag="xzyzy"/>
      <user name="perm01" tag="noside"/>
      <user name="extuser" tag="shazam"/>
    </users>
    <ipfilters>
      <ipfilter name="filter1" type="restrictive">
        <allow ip="141.206.0.0/255.255.0.0"/>
        <deny ip="141.206.35.0/255.255.255.0"/>
        <appliesto tagref="xzyzy"/>
        <appliesto tagref="shazam"/>
      </ipfilter>
      <ipfilter name="filter2" type="permissive">
        <deny ip="141.206.35.0/255.255.255.0"/>
        <allow ip="141.206.35.175/255.255.255.255"/>
        <appliesto tagref="noside"/>
        <appliesto tagref="xzyzy"/>
      </ipfilter>
    </ipfilters>
  </system>
</tdat>
```

## About Enabling IP Restrictions

IP restrictions do not take effect until you use the ipxml2bin utility to convert the XML to binary and transfer it to either the IP GDO (on database nodes) or the bin file . See [Enabling XML-Based IP Restrictions with the ipxml2bin Utility](#).

The initial setup of IP restrictions requires a system tpareset, but if you rerun the ipxml2bin utility the tpareset is not required. See [Editing or Disabling IP Restrictions](#).

## When Multiple Filters Exist

When multiple filters exist that reference a single user, the system considers all filters as if they were a single, large filter. If any filter denies the user access to the database, the user logon fails.

## When No Filter Exists

To maintain backward compatibility with previous versions of Teradata Vantage, if no IP filter exists for a user, the user can access the database from any IP address.

## Applying a Filter to All Users

Instead of listing individual user tags in the document IP filter(s), you can apply an IP filter to all users.

1. Create an XML document with only the users element, and no individual user elements.
2. Assign a value of allusers to the users tag attribute.
3. Assign the same value to the appliesto tagref attribute

The system tests an allusers filter first when it processes a logon for applicable restrictions.

## Example: XML Document Linking a Filter to All Users

```
<?xml version="1.0" encoding="UTF-8"?>
<tdat name="tdat">
  <system name="gizmo">
    <users tag="allusers">
    </users>
    <ipfilters>
      <ipfilter name="filter1" type="permissive">
        <deny ip="141.206.35.0/255.255.255.0"/>
        <allow ip="141.206.35.175/255.255.255.255"/>
        <appliesto tagref="allusers"/>
      </ipfilter>
    </ipfilters>
  </system>
</tdat>
```

The following table provides an explanation of the terms used in an allusers filter. The descriptions cover only the differences from standard filters. For the terms not covered in this table, refer to the examples on [About Permissive Filters](#) and [About Restrictive Filters](#).

Term	Description
<code>&lt;users tag="allusers"&gt;</code>	Provides the means to apply the filter to all users. The term allusers is for reference only, and you can use another term in its place, if you also reference the same term in the appliesto tagref.
<code>&lt;appliesto tagref="allusers"/&gt;</code>	Applies the filter to all users without the need to list them individually, if the users tag also specifies allusers.

## Saving a Completed XML IP Restriction Document

When you complete an XML IP restriction document, save the document to the `/opt/teradata/tdat/tdgss/` site directory on the lowest numbered Teradata Vantage node in preparation for testing the restrictions, for example, `ipfilter.xml`. You must specify this file name when running the `ipxml2bin` utility to enable the restrictions.

## Enabling XML-Based IP Restrictions with the `ipxml2bin` Utility

You must run the `ipxml2bin` utility to transfer the saved restrictions to the GDO. The utility looks for the file in the `/opt/teradata/tdat/tdgss/site` directory.

### Syntax

```
ipxml2bin {-f output_file_name | -G } input_file_name
```

### Syntax Elements

#### **-f output\_file\_name**

[Deprecated] An alternate file location for the `ipxml2bin` output, for use when testing the restrictions before committing them to the IP GDO.

#### **-G**

Causes the output to be written to the IPFILTER GDO.

#### **input\_file\_name**

The saved IP XML document file.

### Procedure

1. From the `/site` directory on the lowest numbered Vantage node, run the `ipxml2bin` utility to commit IP restrictions to the GDO.

```
$ ipxml2bin -G input_file_name
Parse successful
784 bytes written to the ipfilter GDO.
```

The command populates the GDO and distributes it to all database nodes.

2. Check for errors.

XML errors that indicate syntax errors in the IP XML document.

Non-XML errors, for example:

- GDO support not available  
The user specified the -G utility option on a system where PDE is not installed.
- GDO size limit exceeded; need #, limit #.  
The data in the XML file exceeds the GDO size limit (128K bytes). You must either reduce the amount of data in the XML file or switch to a directory-based solution.

3. Run the tpareset utility to enable the restrictions.

---

**Note:**

This step is only necessary for the initial implementation of IP restrictions, and does not apply to revising the XML document.

---

## Testing XML-Based IP Restrictions

From the command prompt, run the `tdgssauth` utility to determine if the restrictions contained in the GDO affect users as expected. Test several users with different IP addresses; test users who should, and users who should not, be restricted from logging on.

This example shows a failure specific to IP addresses. When IP restrictions prevent a log on attempt, minor status `0xe10000ed` is displayed in the last line of the output.

```
$ tdgssauth -m ldap -u diperm01 -i 141.206.3.15
TDGSS_BIN_FILE not set.
TDGSSCONFIG GDO used in tdgss.
Please enter a password:
                Status: authenticated, not authorized
                Database user: perm01 [permanent user]
                Profile: profile01
                External roles: extrole01perm01, extrole02perm01, extrole03perm01
                Authenticated user: ldap://
esroot.example.com:389/CN=diperm01,OU=people,OU=testing,DC=example,DC=com
                Audit trail identifier: diperm01
                Authenticating service: esroot1
```



```
Actual mechanism employed: ldap [OID 1.3.6.1.4.1.191.1.1012.1.20]
Mechanism specific data: diperm01
```

```
Security context capabilities: replay detection
                             out of sequence detection
                             confidentiality
                             integrity
                             protection ready
                             exportable security context
```

```
The TDGSS function tdgss_inquire_policy_for_user returned an error:
Major status 0x000d0000 - Failure
Minor status 0xe10000ed - The user is not permitted to log on from the
IP address.
```

## Creating IP Restrictions in a Directory

Instead of using the IP XML document method, you can create IP restrictions in a supported directory that serves the Teradata Vantage system(s).

If directory users log on through Unity, IP restrictions must be configured exactly the same for each database system managed by Unity. IP restrictions are not configured on Unity, which only passes the user IP address to Vantage.

---

### Note:

You must use Teradata schema extensions to configure IP filter directory objects. Directories configured using only native directory schema cannot employ directory-based IP restrictions. See [Using Native Directory Schema to Provision Directory Users](#).

---

## Prerequisites

- Install Teradata schema extensions, including the special IP extensions, in the directory. See [Installing Teradata Schema Extensions in a Certified Directory](#).
- Configure the LDAP mechanism properties required for your directory strategy. See [Directory Management of Database Users](#).
- Configure directory objects to enable directory authentication and authorization of database users. See the topics beginning with [Creating the Top-Level Objects in the DIT](#).

## Implementation Process for Directory-Based IP Restrictions

1. Review the concepts in [Designing Directory-Based IP Restrictions](#).

2. Review the [About Standard Teradata Schema Objects in IP Restrictions](#), [About Special IP Filter Schema Objects in IP Restrictions](#), and [Working with IP Filter Attributes](#) that you must use to define directory-based IP restrictions.
3. Create IP filter containers and IP filter objects in the directory, listing the database users (tdatUser objects) that are affected in the tdatIPFilterMember attributes for each filter. See [Creating IP Filters Containers and Inserting IP Filters](#).

**Note:**

Directory-based IP restrictions initially apply only to tdatUser objects, which are directory representations of users defined in the database. To apply IP restrictions to directory users, you must map the directory users to the tdatUser objects affected by the filters. See [Applying IPFilters to Directory Users](#).

4. Save the IP restriction-related objects and mappings in the directory.
5. Test the restrictions. See [Testing Directory-Based IP Restrictions](#).
6. After you complete testing and any necessary revisions, implement the restrictions in the database GDO. See [Enabling Directory-Based IP Restrictions with the ipdir2bin Utility](#).
7. Use tpareset to restart the database to enable the directory-based restrictions.

**Note:**

You only need to restart the database for the initial implementation of IP restrictions. Subsequent changes to the restrictions do not require a restart.

## Designing Directory-Based IP Restrictions

You must use Teradata schema extensions, including both standard Teradata schema objects and special Teradata IP filter schema objects, to create directory-based IP restrictions. The function of the IP filter schema objects and attributes is similar to the IP filter elements required for an XML IP restriction document.

**Note:**

This topic presents the differences between directory IP filter objects and the corresponding XML restriction elements, principally name differences. Make sure you understand the function of the corresponding XML elements before you begin to implement directory-based IP restrictions.

## About Standard Teradata Schema Objects in IP Restrictions

The following standard Teradata schema objects required for IP restrictions should already exist in the directory, as part of the normal directory setup shown in [Provisioning Directory Users with Teradata Schema Extensions](#).

## tdatRootNode

The tdatRootNode object identifies the root in the directory information tree (DIT). It corresponds to the tdat element in an IP XML document, and must appear in lower level objects to bind the object definitions to the Teradata root node. The cn attribute of the tdatRootNode object corresponds to the name attribute of the tdat element.

You must include the cn of the tdatRootNode in any IP filter object in the directory information tree.

## tdatSystem

The tdatSystem object corresponds to the system element in an IP XML document. It has a name attribute that corresponds to a Teradata Vantage system, or Unity server.

The tdatSystem object must be a child of the tdat object, and the parent of the authorization hierarchy for LDAP users.

The LdapSystemFQDN property for a system points to the tdatSystem object that defines LDAP authorizations (including IP filters) for directory users who access the system.

## tdatUsers

The tdatUsers object is a type of tdatContainer, with a cn=users, which contains individual tdatUser objects. A tdatUsers container must be a child of a tdatSystem object.

## tdatUser

Each tdatUser object describes a Vantage user. Reference tdatUser objects in an ipFilterMember attribute of an IPFilter object to apply the filter rules to that user. A user object must be a child of a tdatContainer, and have a cn that is a Vantage username.

In a Unity environment, each user represented by a tdatUser object must exist in all Unity-managed database systems.

---

### Note:

Each tdatUser must correspond to a user defined in the database. To apply IP restrictions to directory users, you must map the directory users to Teradata user objects that have the needed restrictions. See [Applying IPFilters to Directory Users](#).

---

## About Special IP Filter Schema Objects in IP Restrictions

Some directory objects necessary for IP restrictions are not used for any other purpose and must be specially created, using the Teradata IP filter schema extensions.

## tdatIPFilters

The tdatIPFilters object is a type of tdatContainer object with the cn=ipfilters, and must be a child of a tdatSystem object. A tdatIPFilters container holds individual tdatIPFilter objects. You can use as many tdatIPFilters objects as needed, but only one appearance is typical.

## tdatIPFilter

A tdatIPFilter object describes an individual IP filter, and must be a child of a tdatIPFilters object. You can create as many tdatIPFilter objects as necessary in the directory.

The tdatIPFilter has several attributes. The value of the name attribute is arbitrary and has no intrinsic meaning except to differentiate it from other filters. Other attributes define the function of the IP filter, as shown in the topics that follow.

## Working with IP Filter Attributes

The tdatIPFilter attributes define the function of an IP filter.

### tdatAllowDeny

The tdatAllowDeny attribute is similar to the filter type element in an IP XML document. It can contain either of two values:

- A value of TRUE defines the filter as a restrictive filter. A restrictive filter:
  - Denies all IP addresses except those specifically allowed.
  - Processes the tdatAllowedIP filter first, to determine the IP address or range of addresses allowed to access the database.
  - Processes the tdatDeniedIP filter second, to identify any exceptions to the allowed IP addresses.
- A value of FALSE defines the filter as permissive. A permissive filter:
  - Allows all IP addresses except those specifically denied.
  - Processes the tdatDeniedIP filter first, to determine the IP address or range of addresses denied access to the database.
  - Processes the tdatAllowedIP filter second to identify any exceptions to the denied IP addresses.

For information on the function of restrictive and permissive filters, see [Working with the Effects of Filter Type on allow and deny Elements](#).

## **tdatAllowedIP**

The `tdatAllowedIP` attribute defines a filter, and requires two values, an IP address and a mask. The `tdatAllowedIP` is similar to an `allow` element in an IP XML document. If the IP address of the incoming session, when ANDed with the mask, matches the IP address in the attribute ANDed with the mask, the filter allows access to the incoming IP address.

For details on how to specify IP addresses and masks for IP filtering, see the topics beginning with [About IP Filters](#).

## **tdatDeniedIP**

The `tdatDeniedIP` attribute defines a filter, and requires two values, an IP address and a mask. The `tdatDeniedIP` is similar to a `deny` element in an IP XML document. If the IP address of the incoming session, when ANDed with the mask, matches the IP address in the attribute ANDed with the mask, the filter denies access to the incoming IP address.

To specify IP addresses and masks, see the topics beginning with [About IP Filters](#).

## **tdatIPFilterMember**

The value of the `tdatIPFilterMember` attribute is a user to which the containing `tdatIPFilter` applies. The `tdatIPFilterMember` attribute corresponds to the `appliesto` element in an IP XML restriction document. The value of the `tdatIPFilterMember` attribute is a unique identifier for a `tdatUser` object, and it binds the IP filter to a database user.

You can use as many instances of the `tdatIPFilterMember` attribute in a `tdatIPFilter` object as needed, but you can specify only one user per instance.

## **Mapping IP Filters to Directory Users**

When you create IP filter objects in the directory, the restrictions initially apply only to database users, as named in the `TdatIpFiltermember` attribute(s).

To apply directory-based IP restrictions to directory users, you must map each directory user needing restrictions to a database user object that is affected by the related IP filter. See [Applying IPFilters to Directory Users](#).

After the directory entries and mappings are complete, save them in the directory.

## **Enabling Directory-Based IP Restrictions with the ipdir2bin Utility**

The `ipdir2bin` utility transfers the directory-based IP address restrictions to the IP GDO.

1. From the /site directory on the lowest numbered Teradata Vantage SQL Engine node, run the ipdir2bin utility, to commit directory IP restrictions to the database GDO:

```
$ ipdir2bin -u dir_username [-w dir_password ] [-h dir_server_name ]
[-S system_name ]
Enter LDAP password:
Parse successful
608 bytes written to the ipfilter GDO.
```

***dir\_username***

Specifies the FQDN of the directory user running the utility.

***dir\_password***

[Optional] Specifies the password for the user *dir\_username*.

Default behavior: System prompts you for a password.

***dir\_server\_name***

[Optional] Identifies the directory server.

The administrator specifies *dir\_server\_name* when doing either of the following:

- Adding a system to a domain
- Explicitly naming the server in the etc/ldap.conf file on a Teradata Vantage system.

Default: LdapServerName property value (see [LdapServerName](#)).

***system\_name***

[Optional] Identifies the FQDN of the Vantage system, as it appears in the tdatSystem object in the directory. See [LdapSystemFQDN](#).

If restrictions are configured for a single Vantage system, the tdatSystem object has the name of the system.

If directory users log on through Unity, the IP restrictions must be configured identically for all Vantage systems. IP restrictions for all database systems are the children of a single tdatSystem object.

Default: LdapSystemFQDN property value from TDGSS configuration files. (If the LdapSystemFQDN property also contains no value, the utility exits with an error.)

The command populates the GDO and distributes it to all database nodes.

2. To enable the committed restrictions, run the tpareset utility. For more information, see *Teradata Vantage™ - Database Utilities*, B035-1102.

**Note:**

This step is only necessary for the initial implementation of IP restrictions, and does not apply to revisions.

3. If the fully enabled IP restrictions do not function as needed, you can:
  - Edit the restrictions, retest them, as shown in [Testing XML-Based IP Restrictions](#), and then re-enable them as shown in the first step.
  - Disable the restrictions, as shown in [Removing All IP Restrictions in an Emergency](#).

In most cases, [Testing XML-Based IP Restrictions](#) should uncover any problems before you enable them on the system.

4. In a Unity environment, repeat this procedure for each Vantage system.

## About Ipdird2bin Errors

Because ipdir2bin accesses the directory using an external LDAP client, any errors that the utility produces are LDAP client errors, specific to the client and server involved.

## Testing Directory-Based IP Restrictions

If you map a directory user to database user object in the directory, the directory user inherits all the IP restrictions that are applicable to the mapped database user, as defined in the IP GDO. You can use `tdgssauth` to check whether the GDO applies the expected IP restrictions to a mapped directory user.

```
$ tdgssauth -m ldap -u diperm01 -i 141.206.3.15
TDGSS_BIN_FILE not set.
TDGSSCONFIG GDO used in tdgss.
Please enter a password:
                Status: authenticated, not authorized
                Database user: perm01 [permanent user]
                Profile: profile01
                External roles: extrole01perm01, extrole02perm01, extrole03perm01
                Authenticated user: ldap://
esroot.example.com:389/CN=diperm01,OU=people,OU=testing,DC=example,DC=com
                Audit trail identifier: diperm01
                Authenticating service: esroot1
                Actual mechanism employed: ldap [OID 1.3.6.1.4.1.191.1.1012.1.20]
                Mechanism specific data: diperm01

Security context capabilities: replay detection
                             out of sequence detection
                             confidentiality
                             integrity
```

```

        protection ready
        exportable security context

```

The TDGSS function `tdgss_inquire_policy_for_user` returned an error:

```

    Major status 0x000d0000 - Failure
    Minor status 0xe10000ed - The user is not permitted to log on from the
    IP address.

```

Based on the results, if the restrictions do not function as needed, you can do one or both of the following:

- Disable the restrictions.  
See [Removing All IP Restrictions in an Emergency](#).
- Edit the restrictions to correct any problems and then enable the revised restrictions.  
See [Editing IP Restrictions in the Directory](#).

When the restrictions pass the test without problems, the IP restrictions are complete.

## Example: Test of IP Access Restrictions for Directory Users

These examples test user `djl` for IP addresses 141.206.35.87, 141.206.35.88, and 141.206.35.89, and that the user is not permitted to log on from 141.206.35.88.

```

$ tdgssauth -m ldap -u djl -i 141.206.35.87
TDGSS_BIN_FILE not set.
TDGSSCONFIG GDO used in tdgss.
Please enter a password:
        Status: authenticated, not authorized
        Database user: perm01 [permanent user]
        Authenticated user: ldap://
esroot.example.com:389/CN=djl,OU=people,OU=testing,DC=example,DC=com
        Audit trail identifier: djl
        Authenticating service: esroot1
        Actual mechanism employed: ldap [OID 1.3.6.1.4.1.191.1.1012.1.20]
        Mechanism specific data: djl

Security context capabilities: replay detection
                             out of sequence detection
                             confidentiality
                             integrity
                             protection ready
                             exportable security context

```



```
Minimum quality of protection: none
Options: none
```

In this example, the last line of the output indicates that logon is denied.

```
$ tdgssauth -m ldap -u djl -i 141.206.35.88
TDGSS_BIN_FILE not set.
TDGSSCONFIG GDO used in tdgss.
Please enter a password:
                Status: authenticated, not authorized
                Database user: perm01 [permanent user]
                Authenticated user: ldap://
esroot.example.com:389/CN=djl,OU=people,OU=testing,DC=example,DC=com
                Audit trail identifier: djl
                Authenticating service: esroot1
                Actual mechanism employed: ldap [OID 1.3.6.1.4.1.191.1.1012.1.20]
                Mechanism specific data: djl

Security context capabilities: replay detection
                             out of sequence detection
                             confidentiality
                             integrity
                             protection ready
                             exportable security context
```

The TDGSS function `tdgss_inquire_policy_for_user` returned an error:

```
Major status 0x000d0000 - Failure
Minor status 0xe10000ed - The user is not permitted to log on from the
IP address.
```

```
$ tdgssauth -m ldap -u djl -i 141.206.35.89
TDGSS_BIN_FILE not set.
TDGSSCONFIG GDO used in tdgss.
Please enter a password:
                Status: authenticated, not authorized
                Database user: perm01 [permanent user]
                Authenticated user: ldap://
esroot.example.com:389/CN=djl,OU=people,OU=testing,DC=example,DC=com
                Audit trail identifier: djl
                Authenticating service: esroot1
                Actual mechanism employed: ldap [OID 1.3.6.1.4.1.191.1.1012.1.20]
                Mechanism specific data: djl
```

```

Security context capabilities: replay detection
                             out of sequence detection
                             confidentiality
                             integrity
                             protection ready
                             exportable security context

Minimum quality of protection: none
Options: none

```

## Editing or Disabling IP Restrictions

You can edit the IP restrictions in an XML document or directory.

If an error in the IP restrictions creates serious system problems (for example, restricting logons for all users, including user DBC), you can disable all IP restrictions immediately.

### Editing IP Restrictions in an XML Document

1. Retrieve the existing IP XML restriction document from the /opt/teradata/tdat/tdgss/site directory.
2. Save the existing XML restriction document to another directory for future reference.
3. Edit a copy of the existing IP XML document to revise or delete unwanted restriction(s), according to the rules for constructing an XML restriction document. See [Designing IP XML Restrictions](#).
4. Save the revised IP XML restriction document in the /site directory.

See [Saving a Completed XML IP Restriction Document](#).

5. Commit the revised restrictions to the GDO. See [Enabling XML-Based IP Restrictions with the ipxml2bin Utility](#).

---

#### Note:

You do not need to run tpareset for edits to the IP restrictions. A reset is only necessary for the initial setup of IP restrictions. GDO updates take effect immediately.

---

6. Test the revised restrictions. See [Testing XML-Based IP Restrictions](#).
7. Rerun these editing steps until the testing yields the desired results.

### Editing IP Restrictions in the Directory

1. You can edit directory entries to:
  - Add or remove an IP filter object
  - Change the applicable IP addresses in an IP filter object
  - Add or remove database users from an IP filter object

- Map or remove the mapping between directory users and database user objects, which applies IP restrictions to the directory users.

For information, see [Creating IP Filters Containers and Inserting IP Filters](#) and [Applying IPFilters to Directory Users](#).

2. Save the revisions to the directory entries.
3. Use the ipdir2bin utility to enable the revised directory-based restrictions. See [Enabling Directory-Based IP Restrictions with the ipdir2bin Utility](#).
4. Test the revised restrictions. See [Testing Directory-Based IP Restrictions](#).

---

**Note:**

You do not need to run the tpareset for edits to the IP restrictions. A reset is only necessary for the initial setup of IP restrictions.

---

5. Rerun these editing steps until the testing yields the desired results.

## Removing All IP Restrictions in an Emergency

In the event of an error in the IP restrictions that creates serious system problems, for example, restricting logons for all users (including user DBC), you can remove all IP restrictions immediately. If you follow the test procedure shown in [Testing XML-Based IP Restrictions](#) or [Testing Directory-Based IP Restrictions](#), this emergency procedure should never be necessary.

1. Save a backup copy of the current GDO file for possible use in diagnostics:

```
cp /etc/opt/teradata/tdconfig/ipfilter.gdo /etc/opt/teradata/tdconfig/ipfilter.gdo.save
```

2. Disable IP filters by replacing the existing GDO file with the blank.xml file:

```
/opt/teradata/tdgss/bin/ipxml2bin -G /etc/ipfilter.xml
```

---

**Note:**

A tpareset is not required.

---

3. Once you edit the IP restrictions to fix the problems, you can test and enable the revised restrictions. See one of the following, depending your restriction implementation:
  - [Editing IP Restrictions in an XML Document](#)
  - [Editing IP Restrictions in the Directory](#)

# Encryption

Teradata Vantage provides the following protection features:

- Password encryption. See [About Password Encryption](#).
- Message encryption and integrity checking. See [About Message Encryption](#).
- Optional quality of protection settings to increase protection strength. See [Working with Quality of Protection Options](#).
- Full disk data encryption.

## About Password Encryption

When users log on to the database, the system transmits passwords in secure form.

### Teradata Vantage Passwords Stored in the Teradata Vantage System

Teradata Vantage stores user passwords in the database in cryptographically hashed form, using SHA-256 (256-bit) hashes.

### Teradata Vantage Passwords Stored in Teradata Wallet

Teradata Wallet stores all string values, including Vantage passwords, in protected form:

- On Linux and other UNIX clients, using AES-256.
- On Windows clients, using Triple-DES, the standard for the Data Protection API (DPAPI).

## About Message Encryption

Users can enable encryption for each transaction with the database, using selectable encryption options from within Teradata Vantage client applications.

### Message Encryption for Teradata (TD2) and LDAP Authentication

When a user authenticated by the TD2 or LDAP mechanism enables message encryption from a TTU 14.10 client, the system uses either the default encryption algorithm or the algorithm defined by applicable QOP security policy, whichever is stronger. For details, see [Working with Quality of Protection Options](#).

### Message Encryption for Kerberos Authentication

For users authenticated by Kerberos (KRB5 or SPNEGO mechanism) who enable message encryption, the system uses the encryption method automatically negotiated by Kerberos, and in accordance

with Internet Engineering Task Force (IETF) RFC 4121 Kerberos Version 5 Generic Security Service Application Program Interface (GSS-API).

For details on determining the encryption method when using Kerberos, see RFC 3961 and RFC 4121. The mechanism QOP is ignored.

## Message Encryption and Performance

Message encryption is CPU intensive; more so for the QoPs that use stronger encryption keys. Depending on the application, the type of session, and the workload, message encryption may adversely impact system performance.

For tactical workloads, there will be negligible performance impact when message encryption is enabled.

For bulk data movement workloads such as Load, Export, Archive, Restore, etc., there may be significantly higher performance impact on throughput, elapsed time to complete the job, and CPU utilization.

However, the higher impact due to CPU utilization will likely be more pronounced on the Client ETL and/or BAR server, which typically has less CPU capacity than the Teradata Vantage server.

## Enabling Encryption

To enable message encryption:

- Set the ConfidentialityDesired property for the authentication mechanism to yes (default).

For details on how to view or edit security mechanism property settings, see [General Rules for Editing the TDGSS Configuration](#).

- Enable encryption from the client application that establishes the session. For instructions on how to enable encryption, see the user guide for the application.

---

### Note:

If you use certain third-party database activity monitoring software, enabling encryption may prevent the monitoring software from detecting the username for a session. Teradata Vantage activity monitoring functions are not subject to this restriction. See [Monitoring Database Access](#).

---

## Data Integrity

By default, Teradata Vantage checks all received encrypted messages for data integrity, that is, to make sure the data was not changed or corrupted during transmission.

You can also configure clients running .NET Data Provider for Teradata to separately enable integrity checking without enabling encryption. See the Help for Teradata .NET Data Provider.

# Working with Quality of Protection Options

## About Quality of Protection

Quality of Protection (QOP) determines the strength of the algorithm used to:

- Encrypt message traffic between a Teradata Vantage system and its clients (confidentiality)
- Calculate the check sum that protects message data from being altered during transmission (integrity checking)

The system examines any applicable QOP policy and the DEFAULT QOP and uses the strongest QOP.

## Default QOP Settings

The system default QOP is defined by the TdgssLibraryConfigFile.xml.

### Note:

You can configure a stronger encryption algorithm for the DEFAULT QOP. See [QOP Configuration Options](#).

```
<!-- Low, Medium and High QOP values are all set to "Default"
unless the Low, Medium and High values are explicitly set
in TdgssUserConfigFile.xml -->
<!-- To update security uncomment one or more QOPs and edit. -->
<!-- DEFAULT QOP
<MechQop Value="Default">
    AES-K128_GCM_PKCS5Padding_SHA2_DH-K2048
    AES-K128_CBC_PKCS5Padding_SHA1_DH-K2048
    AES-K192_GCM_PKCS5Padding_SHA2_DH-K2048
    AES-K192_CBC_PKCS5Padding_SHA1_DH-K2048
    AES-K256_GCM_PKCS5Padding_SHA2_DH-K2048
    AES-K256_CBC_PKCS5Padding_SHA1_DH-K2048
</MechQop>
-->
```

## QOP Configuration Options

You can enable QOP strength options in the TD2, JWT, PROXY, and LDAP mechanisms, to use in enforcing QOP security policy. For information about how QOP strength options are used in QOP security policy, see [Network Security Policy](#).

**Note:**

For users authenticated by Kerberos (KRB5 or SPNEGO mechanism) who enable message encryption or are subject to a confidentiality QOP policy, the system uses the encryption method and strength automatically negotiated by Kerberos. See [Message Encryption for Kerberos Authentication](#).

The following table describes QOP configuration options for each element in the QOP section of the TDGSS configuration file.

Configuration File Element	Configuration Options
DEFAULT QOP	<p>You can edit the DEFAULT QOP to a different encryption strength.</p> <ul style="list-style-type: none"> <li>• AES-128 (default)</li> <li>• AES-192</li> <li>• AES-256</li> </ul>
<ul style="list-style-type: none"> <li>• LOW SECURITY QOP</li> <li>• MEDIUM SECURITY QOP</li> <li>• HIGH SECURITY QOP</li> </ul>	<p>If you plan on configuring QOP security policies, you can:</p> <ul style="list-style-type: none"> <li>• Enable the LOW, MEDIUM, and HIGH SECURITY QOP entries for use in confidentiality and integrity policies.</li> <li>• Optionally edit the LOW, MEDIUM, and HIGH SECURITY QOP entries to specify a different encryption strength.</li> </ul>

## Working with TdgssUserConfigFile.xml QOP Entries

Because the TdgssUserConfigFile.xml is not overwritten when you upgrade to a new Vantage version, you may have to import part or all of the QOP configuration from outside the TdgssUserConfigFile.xml if you want to edit QOP entries. The release version of the last fresh install of Vantage indicates which entries you may have to import.

## Teradata Vantage Fresh Install

For a fresh installation of Vantage, the TdgssUserConfigFile.xml includes all currently available QOP options. You can uncomment QOP entries (if not already uncommented) and edit them as needed. See [Changing the QOP Configuration](#).

```
<!-- To update security uncomment one or more QOPs and edit. -->
<!-- DEFAULT QOP
<MechQop Value="Default">
  AES-K128_GCM_PKCS5Padding_SHA2_DH-K2048
  AES-K128_CBC_PKCS5Padding_SHA1_DH-K2048
  AES-K192_GCM_PKCS5Padding_SHA2_DH-K2048
  AES-K192_CBC_PKCS5Padding_SHA1_DH-K2048
  AES-K256_GCM_PKCS5Padding_SHA2_DH-K2048
  AES-K256_CBC_PKCS5Padding_SHA1_DH-K2048
```

```
</MechQop>
-->
```

## QOP Configuration Change Guidelines

- The system attempts to use the first DEFAULT QOP listed, but tries others if the first QOP does not work. For example, Java clients do not support encryption stronger than AES-128 without installation of a special security policy package, and will use AES-128 regardless of what QOP is listed first.

To allow Java clients to use stronger encryption, download the JAVA Cryptography Extension (JCE) Unlimited Strength Jurisdiction Policy Files and copy the jar files to the <JRE home>/lib/security directory.

- You can delete QOPs from the DEFAULT QOP list to narrow the encryption options.
- Changing the TDGSS configuration on a database system requires a tpareset, during which the database is temporarily unavailable. Plan to make QOP changes along with other TDGSS configuration changes to minimize downtime.
- If users log on through Unity you must also configure QOP on the Unity server. If you want the QOP setting to be the same for users who log on through Unity as it is for users logging directly on to the database, the QOP configuration should match between the Unity server and each connected database.
- Guidelines for Galois/Counter Mode (GCM) and Counter with Cipher Block Chaining-MAC (CCM):
  - CCM mode is not supported in Java.
  - GCM and CCM are authenticated encryption modes.
  - GCM mode is supported in Java 1.8 and later.

## Preparing to Change the QOP Configuration

- On the Vantage node with the lowest ID number, navigate to the directory that contains the TdgssUserConfigFile.xml.

```
cd /opt/teradata/tdat/tdgss/site
```

- Make a backup copy of the TdgssUserConfigFile.xml and save it according to your site standard backup procedures. If the changes you make do not work as expected, you can revert to the previous configuration.
- Open a text editor, such as vi, and bring up a working copy of the TDGSS user configuration file.

```
vi TdgssUserConfigFile.xml
```

## Changing the QOP Configuration

If the default QOP settings do not meet site needs, you can edit the TdgssUserConfigFile.xml to change the settings.



**Note:**

Changing the TDGSS configuration requires a tpareset, during which the database is temporarily unavailable. Make all QOP changes, and any other needed TDGSS configuration changes, at the same time so they can all be included in a single run of the run\_tdgssconfig tool and the tpareset utility. See [Changing the TDGSS Configuration](#).

## Changing the Default QOP Strength

If the default QOP strength does not meet site needs, you can edit the DEFAULT QOP configuration for the LDAP, TD2, and JWT mechanisms so sessions that enable encryption default to a stronger algorithm.

1. Uncomment the DEFAULT QOP in TdgssUserConfigFile.xml (if not done previously) and edit it by reordering the list to put the needed encryption strength at the top of the list or remove a value, for example:

```
<!-- To update security uncomment one or more QOPs and edit. -->
<!-- DEFAULT QOP
<MechQop Value="Default">
    AES-K128_GCM_PKCS5Padding_SHA2_DH-K2048
    AES-K128_CBC_PKCS5Padding_SHA1_DH-K2048
    AES-K192_GCM_PKCS5Padding_SHA2_DH-K2048
    AES-K192_CBC_PKCS5Padding_SHA1_DH-K2048
    AES-K256_GCM_PKCS5Padding_SHA2_DH-K2048
    AES-K256_CBC_PKCS5Padding_SHA1_DH-K2048
</MechQop>
-->
```

**Note:**

If you remove AES-128 from the list and the Legacy QOP is still enabled, execution of the run\_tdgssconfig utility in the following step exits with an error.

2. After editing, use the run\_tdgssconfig utility to update the TDGSSCONFIG GDO.

```
/opt/teradata/tdgss/bin/run_tdgssconfig
```

3. Run tpareset to activate the changes to the TDGSS configuration.

```
tpareset -f "use updated TDGSSCONFIG GDO"
```

For more information, see [Global QOPs](#).

## Enabling and Changing Low, Medium, and High QOP Entries

You can enable the LOW, MEDIUM, and HIGH QOP entries for the TD2, PROXY, JWT, and LDAP mechanisms to support the use of QOP security policies. For information about configuring a QOP security policy, see [Network Security Policy](#).

You can change the encryption strength for any entry by substituting another algorithm.

1. Uncomment the LOW, MEDIUM, and HIGH QOP entries to enable them for use with QOP security policies.

```
<!-- LOW SECURITY QOP -->
<MechQop Value="Low">
    AES-K128_CBC_PKCS5Padding_SHA1_DH-K2048
</MechQop>
<!-- MEDIUM SECURITY QOP -->
<MechQop Value="Medium">
    AES-K192_CBC_PKCS5Padding_SHA1_DH-K2048
</MechQop>
<!-- HIGH SECURITY QOP -->
<MechQop Value="High">
    AES-K256_CBC_PKCS5Padding_SHA1_DH-K2048
</MechQop>
```

2. You can optionally edit the LOW, MEDIUM, and HIGH QOP entries by changing to a stronger encryption algorithm, for example:

```
<!-- LOW SECURITY QOP -->
<MechQop Value="Low">
    AES-K192_GCM_PKCS5Padding_SHA2_DH-K2048
</MechQop>
```

3. After you complete editing, run the `run_tdgssconfig` utility to update the TDGSSCONFIG GDO.

```
/opt/teradata/tdgss/bin/run_tdgssconfig
```

4. Run `tpareset` to activate the changes to the TDGSS configuration.

```
tpareset -f "use updated TDGSSCONFIG GDO"
```

For more information, see [Changing the TDGSS Configuration](#).

## Full Disk Encryption

Full Disk Encryption (FDE) is supported on the following platforms:

- Teradata Active Enterprise Data Warehouse 6800/6805 platform

- Teradata® IntelliFlex® 1.x / 2.x / 2.x.x

With FDE, the disk controller automatically encrypts all data written to the disk and decrypts the data when it is accessed, without performance degradation. FDE is available to meet requirements of the medical and banking industries that require safe guards against data theft if a disk drive is lost or stolen.

# Monitoring Database Access

You can monitor database activity to collect information on user access to the database and possible attempts to perform unauthorized activities.

## Access-Monitoring Implementation Process

1. Review the logging that occurs automatically. See [About Default Logging](#).
2. Implement logging for permanent database users.
  - a. Set up the DBC.AccLogRule macro. See [Setting Up the DBC.AccLogRule Macro](#).
  - b. Use the BEGIN LOGGING statement to define logging rules and enable logging. See [Enabling Logging with the BEGIN LOGGING Statement](#).
  - c. Check the DBC.AccLogRuleTbl table after executing each BEGIN LOGGING statement to make sure the rule is correct. See [Verifying that the Access Log Rule Is Correct](#).
3. If directory-based users have access to the database, you can implement directory user logging. See [Using Access Logging for Directory-Based Users](#).
4. If you set up middle-tier applications as trusted users, review [Using Access Logging for Proxy Users](#) to understand how the database logs proxy users.
5. Review the sample implementation to see a typical setup for access logging. See [Sample Implementation of Access Logging](#).
6. Review access logs and investigate suspect entries. See [Investigating Database Access Attempts](#).
7. Periodically purge access logs to limit the space devoted to storing log data. See [About Access Log Maintenance](#).

---

### Note:

For information about logging access to objects protected by row level security, see [Using Access Logging with Row Level Security](#).

---

## Network Encryption Auditing

You may audit the security level used by the client interfaces when communicating with the gateway. This audit shows the security level that client interfaces are using on the network when sending messages to the database. The messages are logged to the gateway log. This feature is enabled from gtwcontrol.

For more information, see [Using Network Encryption Auditing](#).

## About Default Logging

By default, Teradata Vantage logs all permanent user attempts to log on and log off in the Event Log.

For information on how to access the Event Log, see [Specifying Parameters to Narrow the Search of System Views](#).

## Working with Access Logging

To ensure accurate logging, make sure that all users log on to the database with their own unique user name. Never allow multiple users to log on with the same user name.

### Setting Up the DBC.AccLogRule Macro

When you execute a BEGIN LOGGING statement, it creates logging rules which it stores in the DBC.AccLogRuleTbl table. Then for each session, Vantage looks in the DBC.AccLogRuleTbl to determine which privilege checks must generate log entries for a specified database user, action, or object.

Before enabling logging, you must do the following to create the DBC.AccLogRule table.

1. Run the DIPACC script to create the special security DBC.AccLogRule macro, which populates the DBC.AccLogRule table with logging information. The DBC.AccLogRule macro may already exist as a result of Vantage setup.
2. Run the tpareset utility to initialize the macro.

For instructions on use of the Database Initialization Program (DIP), see *Teradata Vantage™ - Database Utilities*, B035-1102.

See also [Disabling Access Logging with the END LOGGING Statement](#).

### Enabling Logging with the BEGIN LOGGING Statement

You can use the BEGIN LOGGING statement to start access logging. BEGIN LOGGING statements have the following results:

- Each BEGIN LOGGING statement creates a logging rule, in the form of an entry in the DBC.AccLogRule table.
- Each time a user request matches one of the logging rules found in the DBC.AccLogRuleTbl, the macro creates an entry in the DBC.AccLogTbl.

#### Basic Syntax

```
BEGIN LOGGING ON [ frequency ] [ operation ] BY username ON object_name ;
```

#### Syntax Elements

##### *frequency*

Frequency with which logging of an event takes place during a session: FIRST, LAST, FIRST AND LAST, or EACH.

**operation**

Action or list of actions which, if present in a user request, triggers logging.

**username**

Name of the user (or users) for which the system logs requests.

**object\_name**

Name of a database object which, if present in a user request, triggers logging.

**Note:**

BEGIN LOGGING parameters are optional. The absence of some parameters causes the system to use default values.

For a detailed description of BEGIN LOGGING syntax and options, see *Teradata Vantage™ - SQL Data Definition Language Syntax and Examples*, B035-1144.

## Database Level Logging

When a BEGIN LOGGING statement specifies logging at the database level, actions requested against all objects in that database generate log entries.

To illustrate this, assume that DatabaseA contains several tables, and that PUBLIC has the INSERT privilege only on Table1.

A current access log rule is based on:

```
BEGIN LOGGING ON FIRST INSERT ON DatabaseA ;
```

Subsequently, if users other than the owner submit INSERT statements against tables in DatabaseA, the log entries show:

- “Granted” for the first insert into Table1
- “Denied” for the first attempt to insert into any other table in DatabaseA

## Logging Sequence

When multiple actions take place in one session, the logging rules consider the sequence of the actions. For example, if the BEGIN LOGGING statement specifies logging on FIRST INSERT, a request that applies an INSERT to two separate tables in a database logs only the first INSERT.

## Table Level Logging

If only table-level access logging is in force, and a user submits a request against a related object, for example, the containing database, or a related view or macro, the privilege check does not generate a log entry.

## Using BEGIN LOGGING With GRANT

Most sites limit certain privileges to specialized users and administrators, for example, CREATE and DROP USER, or CREATE and DROP DATABASE statements. You can monitor use of the GRANT statement with those privileges to make sure unauthorized users do not receive them.

```
BEGIN LOGGING WITH TEXT ON EACH USER, DATABASE, GRANT ;
```

## Logging MODIFY Statements

While most SQL statements require database privilege checks and therefore generate log entries, MODIFY is not a privilege that you can grant, so the system does not check MODIFY privileges.

Because users must have the DROP privilege to use MODIFY, you can specify the DROP privilege in BEGIN LOGGING statements for the DATABASE, PROFILE, or USER modified, to log MODIFY activities.

## Verifying that the Access Log Rule Is Correct

After you execute a BEGIN LOGGING statement, you can check the DBC.AccLogRulesV view to make sure the rule is correct.

For example:

```
SELECT * FROM DBC.AccLogRulesV ;
```

For further information about DBC.AccLogRulesV, see *Teradata Vantage™ - Data Dictionary*, B035-1092.

## About DBC.AccLogTbl Entries

Teradata Vantage generates a log entry only if a logging rule is present and active for the object, action, or user for which it performs a privilege check. When the database finds one or more active rules, it logs the associated privilege checks in the DBC.AccLogTbl table for each user request that matches a rule.

A log entry does not necessarily indicate that the statement executed, instead, it indicates that the system checked on the privileges required to execute the statement. The corresponding row in DBC.AccLogTbl shows an entry of either “denied” or “granted.”

Log entries may contain one or two user names:

- The log entry always shows the logon username for the user that initiates the session.
- The log entry may show a second name for some entries. For example, if a user submits an EXECUTE statement for a macro, the system checks the database privileges of the logon username for the EXECUTE statement, and also checks the database privileges required of the macro owner for individual statements within the macro. The log entry lists both the logon user and the macro owner.

## Disabling Access Logging with the END LOGGING Statement

You can use the END LOGGING statement to terminate access logging on any user, action, or object for which one or more logging rules are currently active.

An END LOGGING statement generates an error if a corresponding BEGIN LOGGING statement is not currently in effect for the database name specified in the END LOGGING statement.

Also consider the following END LOGGING characteristics:

- You cannot omit the BY username clause to apply the END LOGGING to all users.
- The END LOGGING statement terminates access logging on the database objects included in the statement. If the END LOGGING statement terminates all logging rules for a particular user, database, or object, the system deletes the row from DBC.AccLogRuleTbl.

To disable access logging:

1. Query the DBC.AccLogRules view to display the list of BEGIN LOGGING statements currently in effect:

```
SELECT * FROM DBC.AccLogRulesV;
```

2. Submit an END LOGGING statement for each BEGIN LOGGING statement in the list that you want to end.
3. Issue a DROP MACRO statement for DBC.AccLogRule macro if you are ending all logging.

---

### Note:

If you do not issue an END LOGGING statement for every rule in DBC.AccLogRules, access logging continues even if you drop the DBC.AccLogRule macro and reset the database.

---

4. Execute the tpareset command to set the change.

For information on use of the tpareset command, see *Teradata Vantage™ - Database Utilities*, B035-1102.

## Using Access Logging for Directory-Based Users

Access logging of directory users generally conforms to the rules for use of access logging of database users, with the following exceptions:



- A SELECT USER request normally returns the current user for a session. When a directory-based user is logged on, a SELECT USER request returns either:
  - The name of the permanent user to which the directory user is mapped
  - The authcid (logon username) of the directory user, if not mapped to a permanent user
- A SELECT ROLE request returns the current role for the session. If the directory user is mapped only to EXTUSER, the initial current role for a directory-based logon is a dummy role called EXTERNAL. Any time the directory-assigned roles are enabled, a SELECT ROLE request returns EXTERNAL as its result.

During access logging, the system identifies directory users by their authcid, which it stores in DBC.SessionTbl.AuditTrailId when it establishes the session.

The format of stored authcid is the same for all directory types.

---

**Note:**

If the authcid exceeds 128 bytes in length (as converted), it truncates at 128 bytes. Therefore, all authcids should be unique for the first 128 bytes.

---

## Identifying Directory Users in Access Logs

Users from several directories or domains with similar names may have access to the same database. To help clarify which user is actually logged on, directory users can log on using the UPN logon format, which allows inclusion of the domain name as part of the username and should uniquely identify each user. See [About Network Logons](#).

## About DBC.SessionTbl Information for Directory User Sessions

The DBC.SessionTbl logs information about each session conducted in the database. This table contains columns with directory related information.

- The LDAP column denotes how the user for a given session is mapped. The column displays one of these options:
  - "P" appears for sessions of a directory user mapped to a permanent database user. This includes auto provisioned users because they are permanent database users.
  - "X" appears for sessions of a directory user not mapped to a permanent database user.
  - "N" appears for sessions that do not involve directory-based users, so they are not externally authenticated.
- The UserName column displays the username for a given session. For a directory user, the username is the authcid, the logon username by which the directory authenticates the user.
- The AuthUser column displays a URL that includes the directory and the user DN, for example:

```
'ldap://directory1.domain1.com/uid=userX,ou=users,dc=domain1,dc=com'
```

In the event users in different directories have the same name, the logged user can be isolated back to a specific directory.

For further information about the DBC.SessionTbl, see *Teradata Vantage™ - Data Dictionary*, B035-1092.

## Using Access Logging for Auto-Provisioned Users

When a user is created by auto provisioning, an event log is generated in DBC.EventLog indicating an AP user has been successfully created.

## Using Access Logging for Proxy Users

By default, the system logs all users that log on through a middle-tier application as a single identity, the application logon username. You can optionally use the GRANT CONNECT THROUGH statement to set up trusted user applications to identify individual proxy users.

When proxy users request services from the trusted user application that involve access to the database, Vantage logs the activities individual users, if logging functions are active, for example, in the DBC.AccessLog and DBC.QueryLog. To assist in investigating security events that involve proxy users, these logs show the entire query band string in effect for the request, and includes the proxy user logon username.

For more on creating proxy users, see [Working with Middle-Tier Application Users](#).

## Example: Investigating Proxy User Activity in DBC.AccessLogV

```
SEL ProxyUser(FORMAT 'X(10)'), QueryBand(FORMAT 'X(45)'),
  StatementText(FORMAT 'X(25)')
from DBC.AccessLogV order by LogDate, LogTime;

*** Query completed. 3 rows found. 3 columns returned.
*** Total elapsed time was 1 second.
```

ProxyUser	QueryBand	StatementText
DG708	=S> PROXYUSER=DG708;PROXYROLE=web;App=eCRM	SEL * from table5;
DG708	=S> PROXYUSER=DG708;PROXYROLE=web;App=eCRM	
	INS table5(c1Val,c2Val);	
DG708	=S> PROXYUSER=DG708;PROXYROLE=web;App=eCRM	SEL * from table5;

**Note:**

=S> indicates that this is a session query band.

## Example: Investigating Proxy User Activity in DBC.QryLogV

```
SEL ProxyUser(FORMAT 'X(10)'), ProxyRole(FORMAT 'X(10)'),
querytext(FORMAT 'X(40)')
from DBC.QryLogV
where username='eCRM' order by StartTime, queryid;
```

```
*** Query completed. 4 rows found. 3 columns returned.
*** Total elapsed time was 1 second.
```

ProxyUser	ProxyRole	QueryText
DG708	Web	SET QUERY_BAND = 'PROXYUSER=DG708'
DG708	Web	SEL * FROM table5;
DG708	Web	INS table5(c1Val,c2Val);
DG708	Web	SEL * FROM table5;

For more information on Query Logging, see *Teradata Vantage™ - Database Administration*, B035-1093.

## Example: Finding the Query Band for an Active Session

You can use the MonitorQueryBand function to find the query band for an active session.

```
BTEQ -- Enter your DBC/SQL request or BTEQ command:
SELECT t1.sessionno, MonitorQueryBand(Hostid, sessionNo, runvprocno)
FROM TABLE (MonitorSession(1,'',0)) AS t1;
```

```
SELECT t1.sessionno, MonitorQueryBand(Hostid, sessionNo, runvprocno)
FROM TABLE (MonitorSession(1,'',0)) AS t1;
```

```
*** Query completed. 2 rows found. 2 columns returned.
*** Total elapsed time was 1 second.
```

SessionNo	MonitorQueryBand(HostId,SessionNo,RunVprocNo)
-----------	---

1299	
------	--

1298	=S> REPJOBID=495;REPORT=Sales;CLIENTMACHINE=XYZ12;REPTYPE=10;
------	---

## Access Logging for Viewpoint Users

Teradata Viewpoint maintains a separate log from Teradata Vantage. This log provides a record of the Viewpoint user ID that performs administrative functions, for example, configuring LDAP or adding a Vantage system.

Viewpoint logging must be explicitly enabled. After logging is enabled, log information appears in `/opt/teradata/viewpoint/portal/logs/action.log`.

For information on configuring and using the Viewpoint log, see *Teradata Viewpoint User Guide*, B035-2206.

## Sample Implementation of Access Logging

The Access Log captures auditable records about the database objects that users access, the frequency of access, and whether or not the access is denied. Various government and industry regulations, such as the Sarbanes-Oxley law in the United States, require logging to trace user activity in databases that contains confidential information.

### Sample Logging Requirements

A common logging requirement is the Payment Card Industry Data Security Standard (PCI-DSS), which requires all retailers, financial institutions, and payment processors that store credit cardholder information to provide audit trails that allow reconstruction of all instances of certain events:

- Attempts to access cardholder data
- Actions of any type taken by any individual with administrative privileges
- Unsuccessful access attempts (denials)
- Creation and deletion of system-level objects
- Initialization of the audit logs (use of the BEGIN/END LOGGING statement)
- Attempts to access audit trails
- Administration of identification and authentication mechanisms

### Sample Logging Setup

The following procedure is based on the PCI-DSS requirements shown in [Sample Logging Requirements](#), but you can use them as a general guide to set up access logging for any sensitive data, by adjusting them to your site security policy.

---

#### Note:

The examples that follow are for reference only and do not in themselves guarantee compliance with PCI-DSS requirements.

---

1. Run the DIPACC script to create the DBC.AccLogRule macro, which must exist before initiating access logging.

For information on running DIPACC, see the Database Initialization Program (DIP) in *Teradata Vantage™ - Database Utilities*, B035-1102.

2. Initiate logging on tables or views that contain sensitive data:

```
BEGIN LOGGING DENIALS ON EACH SELECT, INSERT, UPDATE, DELETE, STATISTICS,
INDEX, REFERENCES, DUMP, RESTORE, SHOW, GRANT, CREATE TRIGGER, DROP TABLE,
DROP TRIGGER ON Table "Tables_Database"."Table_Name" ;
```

Repeat this step for each table or view that contains sensitive data.

3. Initiate logging of any action taken by all users with administrative privileges:

```
BEGIN LOGGING ON EACH ALL BY "DBADMIN","DBC","SECADMIN","admin1","admin2" ;
```

where DBADMIN, SECADMIN, admin1, and admin2 are the database usernames of administrators.

4. Initiate logging of all invalid access attempts:

```
BEGIN LOGGING DENIALS ON EACH ALL ;
```

5. Initiate logging of any query that creates or deletes a system level object, that is, in the space directly owned by user DBC:

```
BEGIN LOGGING ON EACH INDEX, REFERENCES, ALTER PROCEDURE, ALTER FUNCTION,
ALTER EXTERNAL PROCEDURE, CREATE OWNER PROCEDURE, CREATE TABLE, CREATE
VIEW, CREATE MACRO, CREATE DATABASE, CREATE USER, CREATE TRIGGER, CREATE
PROCEDURE, CREATE FUNCTION, CREATE EXTERNAL PROCEDURE, CREATE AUTHORIZATION,
DROP TABLE, DROP VIEW, DROP MACRO, DROP DATABASE, DROP USER, DROP TRIGGER,
DROP PROCEDURE, DROP FUNCTION, DROP AUTHORIZATION ON Database "DBC" ;
```

6. Log the initialization of the audit logs:

```
BEGIN LOGGING WITH TEXT ON EACH ALL ON MACRO DBC.AccLogRule;
```

7. Initiate logging on all attempts to access audit trails:

```
BEGIN LOGGING ON EACH ALL ON TABLE DBC.AccessLogTbl;
BEGIN LOGGING ON EACH ALL ON VIEW DBC.AccessLog;
BEGIN LOGGING ON EACH ALL ON VIEW DBC.DeleteAccessLog;
BEGIN LOGGING ON EACH ALL ON TABLE DBC.EventLog;
BEGIN LOGGING ON EACH ALL ON VIEW DBC.LogOnOff;
```

## Monitoring Query Band Logs

Transaction, session, and profile query bands are logged in DBC tables. The information can be viewed in: DBC.AccessLogV, DBC.QryLogV, and DBC.ProfileInfoV.

For more information, see *Teradata Vantage™ - Data Dictionary*, B035-1092.

## Monitoring Security Policy Violations

When the system is set up to enforce QOP security policy, it logs policy-related information for the session in the DBC.EventLog. See [Monitoring QOP Security Policy](#).

## Using Network Encryption Auditing

Network encryption auditing logs the security level that client interfaces use on the network when sending messages to the database. The messages are logged to the gateway log.

Network encryption auditing allows internal and external security auditors to know which security levels are being used on all connections and when. It allows security administrators to identify IP addresses and users not in compliance with security policy. It reports the following encryption security levels:

- Plaintext
- Integrity, Default
- Integrity, Low
- Integrity, Medium
- Integrity, High
- Confidentiality, Default
- Confidentiality, Low
- Confidentiality, Medium
- Confidentiality, High

There are three auditing options:

- No auditing: The feature is disabled, so nothing is logged.
- Audit everything: Any change in the security level used by a client interface for a session is logged in the current gateway log file.
- Audit cleartext: This only logs security level changes that result in a security level that does not guarantee confidentiality.

This feature is enabled from gtwcontrol; it is disabled by default. For more information about gtwcontrol, see *Teradata Vantage™ - Database Utilities*.

### About the Gateway Log

Network encryption auditing is logged to the gateway log. The auditing information is scattered throughout the log file because it is logged (along with other entries) by the session's gateway to its current log on its own node.

A new log is opened on restart or when the old log reaches a certain size. Logs that are older than seven days old are deleted when a gateway opens a new log.

The auditing information is not entered into database tables, so it must be extracted from the logs and imported to the database to allow sophisticated analysis.

### **Example: Enabling Network Encryption Auditing to Log Every Change in Security**

The example shows the command to enable network encryption auditing to log every change in the security level of incoming messages. Once enabled, this logs the security level of the first message after a session logs on and also logs subsequent security level changes for each session. Run:

```
gtwcontrol --auditnetsecurity=yes
```

---

#### **Note:**

Only sessions that log on after the flag is set will have their security level logged.

---

### **Example: Enabling Network Encryption Auditing to Log Every Cleartext Change in Security Level**

The example shows how to determine which client software is not using encryption. The following command enables network encryption auditing to log all cleartext security levels, that is, any level that is not explicitly a Confidentiality level. Run:

```
gtwcontrol --auditnetsecurity=ct
```

If any sessions are not using encryption, a message similar to the following will be logged:

```
gtwnetio.cpp @1816 (117455456): Thu Jan  5 20:10:30 2017
      Client Security Level: Plaintext for Request 2 from HG 1, Session 1115,
IPAddr 192.0.2.2, Port 50117, User "TESTUSER"
```

### **Example: Disable Network Encryption Auditing**

The example shows the command to disable network encryption auditing. Run:

```
gtwcontrol --auditnetsecurity=no
```

## **Auditing Logons by Clients that Cannot Automatically Follow Security Policy**

The Teradata Vantage gateway can be configured to audit your system to identify the client interfaces or proxies that cannot automatically follow security policy. These are the client interfaces or proxies that need to be manually configured, upgraded, or replaced if RequireConfidentiality is to be set or network security policy is to be used. To audit for this, use the Gateway Control `--secpynotsupported log=` flag to enable

logging. For more information, see *Teradata Vantage™ - Database Utilities*, B035-1102 and [Configuring the Gateway to Allow Logons from Older Interfaces or Proxies](#).

### About the Default Setting for `--secpynotsupported log`

The default setting for the `--secpynotsupported log` flag is `no`, which means the gateway does not log a message to the gateway log files to identify older clients or proxies that are unable to automatically support security policy. To restore the default setting, use:

```
gtwcontrol --secpynotsupported log=no
```

## Examples: Enable Logging for Clients and Proxies that are Unable to Automatically Support Security Policy

### Example: Enabling Logging for All

Setting the `--secpynotsupported log` flag to `all` means the gateway logs a message in a gateway log file when an attempt is made to log on using a client or proxy that is unable to automatically support security policy:

```
gtwcontrol --secpynotsupported log=all
```

### Example: Enabling Logging for Clients

Setting the `--secpynotsupported log` flag to `client` means the gateway logs a message in a gateway log file when an attempt is made to log on using a client that is unable to support security policy:

```
gtwcontrol --secpynotsupported log=client
```

#### Note:

The gateway does not log a message if a logon attempt uses a proxy that is unable to support security policy, because such a proxy is incapable of telling the gateway when the client does not support security policy.

### Example: Enabling Logging for Proxies

Setting the `--secpynotsupported log` flag to `proxy` means the gateway logs a message in a gateway log file when an attempt is made to log on using a proxy that is unable to automatically support security policy:

```
gtwcontrol --secpynotsupported log=proxy
```

## Related Information

For more information, see *Teradata Vantage™ - Database Utilities*, B035-1102.



For information on how to configure the Teradata Vantage gateway to allow logons by clients that cannot automatically follow security policy, see [Configuring the Gateway to Allow Logons from Older Interfaces or Proxies](#).

## Using External Monitoring Software

Some sites may use third-party database activity monitoring software in addition to, or instead of, using the built in Teradata Vantage monitoring capabilities.

If your site uses third-party database activity monitoring software, see your vendor for installation and configuration instructions. Your vendor may instruct you to use Gateway Control (gtwcontrol) options during configuration of the third-party software.

## Investigating Database Access Attempts

### About the Investigation Process

1. Investigate the log of user access privilege checks that result from the logging rules set up.
2. Become familiar with the security-related tables and views that can assist you in understanding aid in investigations of user behavior and security events. See [Investigating Database Access Attempts](#).
3. Identify users who make unsuccessful attempts to access data, and if necessary, take remedial action according to your site security policy.

### About Access Logging Information in System Views

Teradata Vantage stores various types of security information in data dictionary tables residing in the DBC system database. System views based on these tables contain access logs and other security-related data.

View Name	Description
<b>Access Logging Views</b>	
DBC.AccessLogV	Indicates the results of a privilege check performed against a Teradata SQL request. The system logs privilege checks based on the criteria defined in the current logon rules.
DBC.AccLogRulesV	Lists the current logging rules, which the system derives from BEGIN LOGGING statements, and uses to determine which privilege checks should create entries in the DBC.AccLogTbl table.
DBC.DeleteAccessLogV	Lists the entries from the access log by date and time, to help you identify aged data that you should remove. You can only remove entries more than 30 days old. To remove all entries over 30 days old, you can enter: <code>DELETE FROM DBC.DELETEACCESSLOGV ALL ;</code>

View Name	Description
DBC.LogOnOffVX	Lists logon and logoff activity, the associated user, session number, and attempted logon events. Event data indicates the reasons for unsuccessful logon attempts. For unsuccessful logons, the table stores the string "Non-existent User," instead of the username used in the logon, unless the DBS Control ShowAllUserNames flag is set to TRUE.
DBC.LogonRulesV	Lists the users named in previous GRANT LOGON or REVOKE LOGON statements, and indicates which users have WITH NULL PASSWORD privileges, which allows them to be externally authenticated. The system uses these entries to determine whether to allow access.
DBC.SecurityLogVX	Lists a subset of the data on privilege checking from DBC.AccLogTbl, limited to username, table, database, logon time, and account.
<b>User and Privilege Views</b>	
DBC.AllRightsVX	Lists all automatically or explicitly granted privileges for a user or database, and the objects to which the privileges apply.
DBC.AllRoleRightsV	Lists all privileges granted to each role.
DBC.RoleInfoVX	Lists the name of the creator for each role.
DBC.RoleMembersVX	Lists each role, all of its members, and whether it is the default role for each of the members.
DBC.UsersV	Lists information about all users defined in the database. The information is derived from system table DBC.DBase.
DBC.UserGrantedRightsV	Lists the explicit privileges that a user grants to other users.
DBC.UserRightsV	Lists all database privileges explicitly granted to each user. It does not list the implicit privileges for the users.
DBC.UserRoleRightsV	Lists all roles, including any nested roles, available to each user, along with the privileges granted to each role. Does not list directory users or their mapped external roles.
<b>Password Control Views</b>	
DBC.ProfileInfoVX	Lists all profiles and associated parameter settings. Use this view to check the password control settings for a profile.
DBC.RestrictedWordsV	Lists words that are not allowed in a password string when the RestrictedWords control is enabled.
DBC.SecurityDefaultsV	Lists the current global password controls and associated values.

## Accessing System Views

You can access security-related system views to review such things as security settings, user privileges, and user activities using an SQL statement, for example:

```
SELECT * FROM <view_name> ;
```

## Specifying Parameters to Narrow the Search of System Views

You can focus the search of a system view by specifying critical parameters in the SQL statement, for example:

```
SELECT a.UserName, a.LogDate, a.LogTime, StatementText, LogonSource
FROM DBC.AccessLogV AS a, DBC.LogOnOffX AS b
WHERE a.UserName = 'DBADMIN'
AND a.SessionNo = b.SessionNo
ORDER BY 2, 3 ;
```

This query displays the SQL statements, along with the logon IP address and date/time information, for a particular username and session number.

## Using MONITOR Related Queries

You can monitor the use of production control features using SELECT queries.

### Example: Identifying Which Users Can Force Other Users Off the System

Use this query to identify users that have the ABORT SESSION (AS) privilege, which allows them to force other users off the system:

```
SELECT DISTINCT UserName FROM DBC.AllRightsX
WHERE AccessRight = 'AS' ;
```

### Example: Identifying Users Recently Forced Off the System

Use this query to identify users that were forced off the system in the past two days:

```
SELECT DISTINCT UserName FROM DBC.LogOnOffX
WHERE Event = 'Forced Off'
AND LogDate > DATE - 3 ;
```

## Example: Identifying Current MONITOR Function Users

Use this query to identify users currently using the MONITOR function:

```
SELECT UserName, IFPNo FROM DBC.SessionInfoX
WHERE Partition = 'MONITOR' ;
```

## Querying Session-Related Views

You can monitor database access at the session level using one or more of these session-related views:

- DBC.LogOnOffVX
- DBC.SessionInfoX

For more information on all system views, see *Teradata Vantage™ - Data Dictionary*, B035-1092.

## DBC.LogOnOffV

The DBC.LogOnOffV view provides information about the success and duration of user sessions, in addition to LogonSource information. This view is helpful to investigate failed logon attempts.

This query returns a list of failed logon attempts that occurred in the last seven days:

```
SELECT LogDate, LogTime, UserName, Event
FROM DBC.LogOnOffV
WHERE Event NOT LIKE ('%Logo%') AND LogDate GT DATE - 7

ORDER BY LogDate, LogTime ;
```

LogDate	LogTime	UserName	Event
07/07/30	08:55:22	Non-existent User	Auth Failed
07/10/21	08:59:53	BRM	Bad Account

For more information on possible values for DBC.LogOnOffV.Event, see *Teradata Vantage™ - Data Dictionary*, B035-1092.

## DBC.SessionInfoX

The DBC.SessionInfoX view provides information about currently logged on users, including the session source (host connection, logon name, and application), query band, the current partition, collation, role, password status, type of transaction, LDAP status, and audit trail ID (user authcid).

For information about collections of sessions initiated under a session pool, use the `DISPLAY POOL` command. For details, see *Teradata® Director Program Reference*, B035-2416.

If you use a multi-tier client architecture, the `LogonSource` field of this view can provide distinct source identification because it originates from the server tier, including the user ID and application name.

**Note:**

CLv2 inserts data strings for TCP/IP sessions into the `LogonSource` field, which truncates strings that exceed 128 bytes.

**Example: Querying Session DBC.SessionInfoX Using BTEQ**

```
Teradata BTEQ 13.00.00.00 for z/OS. Enter your logon or BTEQ command:
.logon tdpv/socal
.logon tdpv/socal Password:
*** Default Character Set Name 'EBCDIC '
*** Logon successfully completed.
*** Transaction semantics are BTET.
*** Total elapsed time was 0.58 seconds.
BTEQ -- Enter your DBC/SQL request or BTEQ command:
sel logonsource from dbc.SessionInfoX;
*** Query completed. 5 rows found. One column returned.
*** Total elapsed time was 0.44 seconds.

LogonSource
-----
(TCP/IP) 05BF 155.64.116.42 IETTST      818 DBADMIN BTEQ 01 LSS
(TCP/IP) 06F9 208.199.59.157 NAG2N2    1396 POINT  BTEQ 01 LSS
(TCP/IP) 04DC 10.243.71.25 PW_OLD      2462 ROOT   ARCMAN 01 LSS
z/OS TDRT      D48734      BATCH      CS210041      Socal  BTQMAIN
z/OS TDPV      AN1005      TSO        AN1005      Socal  IKJFT01
```

The first three lines report network-connected sessions. The last two lines report mainframe-connected sessions, where:

Field	Value in Querying Example Output	Description
First	(TCP/IP)	Connection name or type (literal)
	(TCP/IP)	
	(TCP/IP)	
Second	05BF	Port or socket identifier

Field	Value in Querying Example Output	Description
	06F9	
	04DC	
Third	155.64.116.42	IP address of the client (host)
	208.199.59.137	
	10.243.71.25	
Fourth	IETTST	Teradata Director Program Identifier (TDPID)
	NAG2N2	
	PW_OLD	
Fifth	818	Client process/thread identifier
	1396	
	2462	
Sixth	DBADMIN	Username used in this session
	POINT	
	ROOT	
Seventh	BTEQ	Name of the application through which the session was initiated
	BTEQ	
	ARCMAN	
Eighth	01	All network sessions (literal)
	01	
	01	
Ninth	LSS	All network sessions (literal)
	LSS	
	LSS	

## About Access Log Maintenance

The DBC.AccessLogTbl collects and retains data while access logging is enabled. To save database space and shorten searches of the log, you should periodically purge the DBC.AccessLogTbl of old data. The purge interval varies based on the type and amount of logging that you enable.

Teradata recommends that you transfer log data into a separate log history table or archive it to tape, so that log history remains available in the event of a subsequent security investigations.

For information about managing accumulated log data, see *Teradata Vantage™ - Data Dictionary*, B035-1092.

# Implementing Row Level Security

## About Row-Level Security

Access to Teradata Vantage objects is controlled primarily by object level user privileges. Object level privileges are discretionary, that is, object owners automatically have the right to grant access on any owned object to any other user.

In addition to object level privileges, you can use row level security (RLS) to control user access by table row and by SQL operation. RLS access rules are based on the comparison of the RLS access capabilities of each user and the RLS access requirements for each row.

Object owners do not have discretionary privileges to grant row access to other users. Only users with security constraint administrative privileges can manage row level access controls.

When multiple Vantage systems are managed by Unity, the same row level security constraints and access privileges should exist on all database systems.

## Row Level Security Compared to View and Column Access Controls

Implementation of row level security can be complicated compared to standard discretionary access controls. Before you commit to using row level security, determine whether or not you can meet access control needs by more conventional means, for example:

- Grant user access to views that do not include columns with sensitive data, instead of granting user privileges on the entire base table.
- Grant or revoke access privileges only on selected columns in the base table.

When comparing access control methods, consider that view and column level access controls:

- Are usually adequate for controlling SELECT statements, but users cannot execute INSERT, UPDATE, and DELETE statements on columns they cannot see, and must revert to accessing the base tables for these operations.
- Are discretionary, that is, the object owner can grant access to any user.

## Related Information

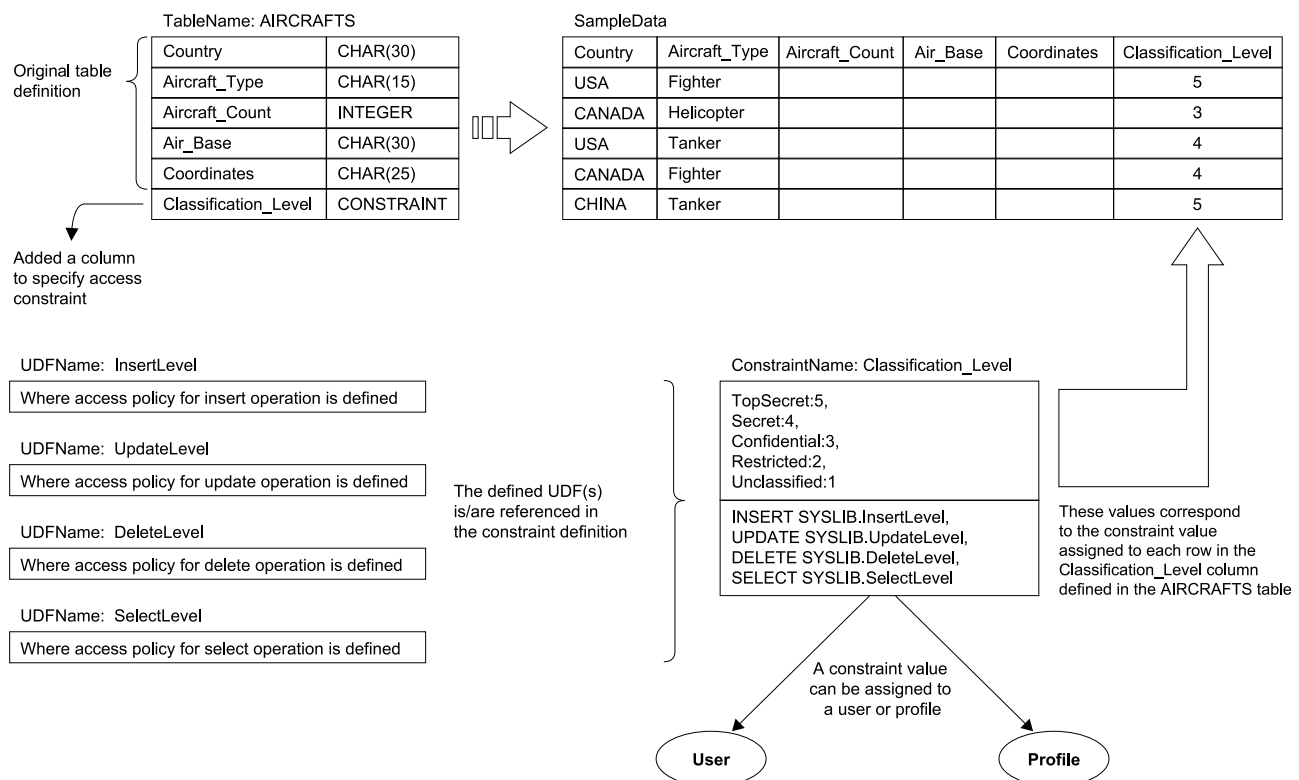
For additional information on the use of view and column level privileges, see [Other Options for Restricting Database Access](#).



## Elements of Row Level Security

Element	Description
Security classification category	A set of labels (access levels or compartments) used to define user access capabilities and row access requirements.
Security CONSTRAINT	<ul style="list-style-type: none"> <li>A CONSTRAINT object named for a security classification system, which: <ul style="list-style-type: none"> <li>Defines the range of valid label values for the classification system</li> <li>Specifies 1 to 4 security constraint UDFs</li> <li>Can be assigned to users to define their row access capabilities</li> </ul> </li> <li>A table column named for a CONSTRAINT object, in which the column value for each row determines the row access requirement</li> </ul>
Security constraint user defined function (UDF)	Defines and enforces RLS policy on each incoming INSERT, SELECT, UPDATE, or DELETE statement.

The drawing shows the components required to define row-level security.



## Row-Level Security Implementation Process

1. Create the security classifications that define security labels for users and data rows. See [Defining Security Labels for Users and Rows](#).
2. Create user-defined functions to define and enforce row level security restrictions. [Creating Row Level Security UDFs](#).
3. Grant the necessary administrator privileges for working with row level security constraints. See [Granting Security Constraint Administrative Privileges](#).
4. Create security constraint objects. See [Working with Security Constraints](#).
5. Assign security constraints and constraint values to database users. See [Working with Constraint Assignments](#).
6. Create/Alter tables to define security constraint columns. See [Working with Security Constraint Columns](#).
7. Assign constraint OVERRIDE privileges to users who need to bypass the enforcement of security constraints. See [Working with Constraint OVERRIDE Privileges](#).
8. Evaluate database objects and processes that interface with RLS tables, and where necessary, rework them to ensure conformance with RLS requirements. See [Working with Row-Level Security Effects](#).
9. Learn how the system derives the session constraint values under various conditions, and how to set alternate values. See [Determining the Session Constraint Values](#).
10. Enable logging of user attempts to access RLS tables, views, and administrative functions. [Using Access Logging with Row Level Security](#).
11. Access system tables and views that contain security constraint information. See [About Constraint-Related System Tables and Views](#).

## About Security Labels

You must set up a system of labels for each security classification category you want to use in defining user access levels and row access requirements.

A security classification system consists of:

- The name of the classification
- The valid labels for use in classification, where each label is a name:value pair

The labels within a classification system may represent a value hierarchy, or they may be a series of compartments with no hierarchical relationship, for example:

- Security clearance (hierarchical): Top Secret, Secret, Classified, Unclassified
- User function (non-hierarchical): Administrator, Programmer, Batch User, End User
- Division/location (non-hierarchical): Canada, China, France, Germany, United States

Each classification system is the basis for:

- A security CONSTRAINT object, which defines a set of applicable access restrictions
- A security constraint column, which apply the restrictions defined in the corresponding CONSTRAINT object to each table in which the column appears

## Defining Security Labels for Users and Rows

Before implementing row level security, you should define the security classification systems and associated labels required to support your site security policy.

1. Define each classification system and identify the labels in the system.

Each system is the basis for a security CONSTRAINT object, which defines a set of access controls. Each user can be assigned up to 6 hierarchical and 2 non-hierarchical constraints.

2. For each table requiring RLS protection, determine which of the classification system (security constraints) should apply to the range of users who access the table.

A table can contain up to 5 constraint columns.

3. Identify how security labels for each system should apply to table rows, and define the user access level required to perform each SQL operation (INSERT, SELECT, UPDATE, and DELETE).

You can use this analysis to help:

- Determine the level of protection required for each row
- Define the SQL access rules used in creating security constraint UDFs
- Determine which UDFs should be used in a security CONSTRAINT object

## Working with Row Level Security UDFs

### About Security Constraint UDFs

A security constraint UDF defines and enforces the rules that determine whether to allow or deny the execution of an INSERT, SELECT, UPDATE, or DELETE statement on a table row.

Each constraint UDF restricts an SQL operation based on a coded rule. UDF rules vary by SQL operation and whether the constraint system is a hierarchical (level) or non-hierarchical (compartment) labeling system.

Hierarchical and non-hierarchical constraints require different kinds of UDFs.

Each time a user accesses a row, the system invokes the UDF associated with the SQL operation (for example, INSERT) to determine if the user can perform the operation. If the requesting user does not have the access level required to perform the operation on the row, the UDF denies the request and request processing moves on to the next applicable row.

---

#### Note:

The system does not require that the user have EXECUTE FUNCTION privileges on the automatically invoked UDFs.

---

If the CONSTRAINT object does not specify a UDF for an SQL operation, the operation succeeds only if the user has the corresponding OVERRIDE privilege.

If a requesting user has the **OVERRIDE** privilege for an SQL operation, the request bypasses the UDF that restricts the operation.

## Basic SQL Access Control Guidelines

The following guidelines, based on the Bell-Lapadula Model, are commonly used for enforcement of access control in government and military applications.

No Read Up (for SELECT operations):

- The session hierarchical level must be  $\geq$  the row hierarchical level.  
Users cannot read a row with a higher classification.
- The session non-hierarchical label must include all compartments found in the row label.  
The user can read a row only if assigned to all compartments used to classify the row.

No Write Down (INSERT/UPDATE operations)

- The row hierarchical level must be  $\geq$  the session hierarchical level.  
New or updated rows inherit the session level. This rule prevents an updating user from accidentally reclassifying the row to a lower level.
- The row label must include all non-hierarchical compartments in the session label.  
New or updated rows inherit all session compartments. This rule prevents an updating user from accidentally adding excess compartmental classifications to a row.

---

### Note:

The sample rules do not contain a **DELETE** policy, but it is common to require that a row be set to the lowest classification level or to **NULL** (declassified), before it can be deleted.

---

## Example: Hierarchical Rules

Operation	Example Rule
INSERT	<p>The current session must have a security label for the security constraint. The session label is entered as the constraint column value for the new row.</p> <p>Purpose: Allows any user with table level insert privileges to insert a new row. Assumes that the user security level is appropriate for the new data.</p>
SELECT	<p>The session security label must equal or exceed the row label or the operation fails.</p> <p>Purpose: Only users classified equal to or higher than a row can read row data.</p>
UPDATE	<p>The session security label must equal or exceed the row label or the operation fails. If the operation is allowed, the updated row uses the session security label for the constraint column value.</p> <p>Purpose: Restricts access to equivalent/higher level users. The row is automatically reclassified to the user level in case the user adds data with a higher classification.</p>

Operation	Example Rule
DELETE	The row cannot be deleted unless the constraint column value is at the lowest security level. Purpose: Ensures that a row is reviewed and declassified before it can be deleted.

## Example: Non-Hierarchical Rules

Operation	Example Rule
INSERT	The current session must have a security label (1 or more compartments). All compartments in the session label are entered as the row constraint column value. Purpose: Forces predictable row classification based on the user label.
SELECT	The session security label must include all the compartments in the row label or the operation fails. Purpose: If a row is classified with several compartments, ensures that the accessing user is a member of all of the compartments.
UPDATE	The row label must include all the compartments contained in the session label. Purpose: Prevents the user from inadvertently adding classifications to the row.
DELETE	The row can be deleted only if the constraint column value is NULL. Purpose: Ensures that a row is reviewed and declassified before it can be deleted.  <b>Note:</b> You must have OVERRIDE UPDATE privileges to reclassify a row as NULL, so that it can be deleted.

## Related Information

For information on how the system determines the session constraint value, see [Determining the Session Constraint Values](#).

## Creating Row Level Security UDFs

Coding a set of UDFs for a security constraint must be coordinated with the purpose and structure of the corresponding CONSTRAINT object. See [Working with Security Constraints](#).

1. Code and file each security constraint UDF as shown in *Teradata Vantage™ - SQL External Routine Programming*, B035-1147.

### Note:

You can create more than one UDF for an SQL operation type, but you can only specify one UDF of each type in a security CONSTRAINT object.

2. Use the CREATE FUNCTION statement to define a FUNCTION object for each UDF, as documented in *Teradata Vantage™ - SQL Data Definition Language Syntax and Examples*, B035-1144.
3. To avoid performance problems that occur when running the UDF in protected mode, use the ALTER FUNCTION statement to set non-protected mode execution for the function:

```
ALTER FUNCTION SYSLIB.function_name EXECUTE NOT PROTECTED
```

#### Note:

If a UDF fails to operate correctly in non-protected mode, it can cause a database restart. Be sure to test each UDF thoroughly before deploying it.

## Related Information

For information on...	See...
Coding a security constraint UDF, including sample code for both hierarchical and compartmental UDFs	<i>Teradata Vantage™ - SQL External Routine Programming</i> , B035-1147.
Required syntax, options, and usage notes for the CREATE/ALTER/REPLACE/DROP FUNCTION statements	<ul style="list-style-type: none"> <li>• <i>Teradata Vantage™ - SQL Data Definition Language Syntax and Examples</i>, B035-1144.</li> <li>• <i>Teradata Vantage™ - SQL Stored Procedures and Embedded SQL</i>, B035-1148.</li> </ul>

## Altering and Replacing Row Level Security UDFs

You can ALTER FUNCTION or REPLACE FUNCTION for a row level security UDF if the resulting UDF retains all parameters specified in the constraint definitions that use the UDF.

## Dropping Row Level Security UDFs

You can drop a row level security UDF with a DROP FUNCTION statement, but you must first drop all CONSTRAINT objects that specify the UDF or ALTER the CONSTRAINT objects to specify a different UDF.

## Working with Security Constraint Administrative Privileges

### About Security Constraint Administrative Privileges

You must grant certain system level privileges to users who administer row level security. By default, user DBC has all security constraint administration privileges, WITH GRANT OPTION.

## Granting Security Constraint Administrative Privileges

To assign system level security constraint administrative privileges to a user or role, use the GRANT statement (SQL form), documented in *Teradata Vantage™ - SQL Data Control Language*, B035-1149.

### Related Information

For information on...	See...
Required syntax, options, and usage notes for the GRANT (SQL form) statement	<i>Teradata Vantage™ - SQL Data Control Language</i> , B035-1149
Assigning CONSTRAINT values and granting and revoking OVERRIDE privileges	<a href="#">Working with Constraint Assignments.</a>

## Revoking Security Constraint Administration Privileges

To remove security constraint administrative privileges from a user or role, use the REVOKE statement (SQL form), documented in *Teradata Vantage™ - SQL Data Control Language*, B035-1149.

## Working with Security Constraints

### About CONSTRAINT Objects

Each CONSTRAINT object defines a set of row level security constraints, including:

- A security label category and the range of values (levels or compartments) for the category.
- Specification of up to 4 UDFs, INSERT, SELECT, UPDATE and DELETE, which define and enforce the security constraints.

You can apply the security constraints in a CONSTRAINT object to a:

- User, by specifying the CONSTRAINT object in a:
  - CREATE USER or MODIFY USER statement
  - CREATE PROFILE or MODIFY PROFILE statement, and then assigning the profile to the user
- Table, by defining a constraint column that is named for the CONSTRAINT object in a CREATE TABLE or ALTER TABLE statement.

## Security Classification Types and Required CONSTRAINT Object Settings

Settings for some options in a security CONSTRAINT object depend on the type of security classification it represents.

Classification Type	Description/Settings
Hierarchical (Non-Set)	<p>All label values are hierarchically related members of the classification category defined by the CONSTRAINT object name.</p> <p>Required settings:</p> <ul style="list-style-type: none"> <li>• Data type: smallint The corresponding constraint column automatically inherits the data type. The smallint data type allows a range of 1 to 10,000 values.</li> <li>• VALUES: The set of name:value pairs in the hierarchy, for example, for the security clearance category: Top Secret:4, Secret:3, Classified:2, Unclassified:1 One integer value from the name:value pairs is assigned to each row in a corresponding constraint column. Users may be assigned multiple values. Row access is based on comparing the session value(s) to the row value(s).</li> </ul>
Non-Hierarchical (Set)	<p>All label values are individual compartments in the classification category defined by the CONSTRAINT object name, and have the same relative weight.</p> <p>Required settings:</p> <ul style="list-style-type: none"> <li>• Data type - byte(n). Allows specification of 1 to 8 times the number of values (compartments) as the number of bytes defined by (n), up to 256 values (32 bytes).</li> <li>• VALUES - The set of name:value pairs that represent all compartments in the category, for example, for the country category: USA:1, UK:2, Canada:3, Japan:4...[country:value] A system uses the user constraint values as the default session value. Row access is based on comparing the session value(s) to the row value(s). The system automatically encodes the applicable constraint values as a binary string that represents the value as a unique bit position (rather than a numeric value), allowing up to 256 values to appear in the column.</li> </ul> <p><b>Note:</b> When doing an INSERT or UPDATE to a table, if the user has OVERRIDE privileges, the operation must supply the hex values. See <a href="#">Example: Loading Tables with User OVERRIDE Privileges</a>.</p>

## Creating CONSTRAINT Objects

Create constraint objects using the CREATE CONSTRAINT statement, documented in *Teradata Vantage™ - SQL Data Definition Language Syntax and Examples*, B035-1144.

See [Security Classification Types and Required CONSTRAINT Object Settings](#).

## Related Information

For information on...	See...
Requirements, options, and usage notes for CREATE CONSTRAINT	<ul style="list-style-type: none"> <li>• <i>Teradata Vantage™ - SQL Data Definition Language Syntax and Examples</i>, B035-1144.</li> </ul>



For information on...	See...
	<ul style="list-style-type: none"> <li>• <i>Teradata Vantage™ - SQL Data Definition Language Detailed Topics</i>, B035-1184.</li> </ul>
System processing of constraints	<a href="#">Determining the Session Constraint Values</a>

## Altering or Dropping CONSTRAINT Objects

The process for altering or dropping a CONSTRAINT object requires that you first remove all constraint assignments.

### Note:

Modifying the name:value pairs in a CONSTRAINT object after a table that contains a corresponding constraint column is archived can have unintended effects on COPY and RESTORE operations. Such modifications may compromise security, depending on which parameters are changed. See [Archive, Copy, and Restore Requirements](#).

1. Determine the database objects to which a security constraint is assigned.
  - a. Find the tables and indexes that have the security constraint. See [Finding Tables and Indexes with a Security Constraint](#).
  - b. Find views that have the security constraint. See [Finding Views that Include a Security Constraint](#).
2. DROP the indexes identified in step 1a.
3. Remove the constraint columns from all tables identified in step 1a using the ALTER TABLE statement. For example:

```
ALTER TABLE table_name
  DROP constraint_column_name ;
```

4. REPLACE all views identified in step 1b to remove the constraint column.
5. Identify the users and profiles to which the constraint is assigned. See [Finding Users or Profiles with an Assigned Constraint](#).
6. Remove the security CONSTRAINT object assignments from:
  - All users to whom the constraint is assigned, using the MODIFY USER statement. See [Changing or Dropping Security Constraints in a MODIFY USER Statement](#).
  - All profiles to which the constraint is assigned, using the MODIFY PROFILE statement. See [About Changing or Dropping Security Constraints in a Profile](#).
7. After removing all assignments for a security constraint, you can either:
  - Use the ALTER CONSTRAINT statement to change the constraint object.

**Note:**

If you do not specify one of the available options in the ALTER CONSTRAINT statement, the CONSTRAINT retains the existing values for the option.

- Use the DROP CONSTRAINT statement to drop the constraint object.

For information about the ALTER CONSTRAINT and DROP CONSTRAINT statements, see *Teradata Vantage™ - SQL Data Definition Language Syntax and Examples*, B035-1144.

8. After altering or dropping a constraint, reverse the actions done in steps 2 through 6 to reapply security constraints to tables, views, users and profiles.

## Related Information

For information on...	See...
Syntax requirements, options, and detailed usage notes for ALTER TABLE, MODIFY PROFILE, and ALTER CONSTRAINT	<ul style="list-style-type: none"> <li>• <i>Teradata Vantage™ - SQL Data Definition Language Syntax and Examples</i>, B035-1144.</li> <li>• <i>Teradata Vantage™ - SQL Data Definition Language Detailed Topics</i>, B035-1184.</li> </ul>
Syntax requirements, options, and detailed usage notes for REVOKE (SQL form)	<i>Teradata Vantage™ - SQL Data Control Language</i> , B035-1149.

## Investigating Security Constraint Object Definitions

Users with the CONSTRAINT ASSIGNMENT or CONSTRAINT DEFINITION privilege can use the SHOW CONSTRAINT statement to display the SQL DDL syntax that defines a CONSTRAINT object, for example:

```
SHOW CONSTRAINT constraint_name ;
```

## Working with Constraint Assignments

To ensure uninterrupted access to data, you should assign security constraints to users before you define security constraint columns in database tables.

## About Assigning Security Constraints

You can assign up to 6 hierarchical (non-set) constraints and 2 non-hierarchical (set) constraints to a user or profile. You must specify at least 1 name:value pair for each constraint.

Each assigned CONSTRAINT object must exist in the database, and the assigned security label values must correspond to valid name:value pairs defined in the object.

When more than one hierarchical label value is assigned to a user or profile, you can include the keyword **DEFAULT** after the label to specify that it is the default value for user sessions. If **DEFAULT** is not specified, the first constraint name listed in the user/profile definition is the default. **DEFAULT** is not applicable to non-hierarchical security constraints.

If a user is the member of a profile that assigns one or more security constraints, the assignments in the profile supersede any assignments made directly to the user.

## Assigning Security Constraints in a CREATE USER Statement

Use the **CREATE USER** statement to assign security constraints to a new user. For example:

```
CREATE USER
  Joe_Smith AS PERM = 1e6, PASSWORD=JoePassword,
  CONSTRAINT = Classification_Level (Secret, Unclassified DEFAULT),
  CONSTRAINT = Classification_Country (US, UK, GER);
```

where:

Syntax Element	Description
Classification_Level (Secret, Unclassified DEFAULT)	Names a hierarchical security <b>CONSTRAINT</b> object and assigns the applicable name:value pairs (levels) according to user needs. In this example the user has two classification levels: <ul style="list-style-type: none"> <li>The Unclassified level (default), allows the user to <b>INSERT</b> new rows without automatically classifying them as secret.</li> <li>The user can also use <b>SET SESSION CONSTRAINT</b> to assume the Secret level, to read highly classified rows.</li> </ul>
Classification_Country (US, UK, GER)	Names a non-hierarchical security <b>CONSTRAINT</b> object and assigns the applicable name:value pairs (compartments) to the user. In this example the user is from the US, but also coordinates operations with the UK and Germany.

### Note:

A new or changed security constraint assignment takes effect at the next user login.

## Related Information

For more information on **MODIFY USER** syntax and options, see:

- Teradata Vantage™ - SQL Data Definition Language Syntax and Examples*, B035-1144
- Teradata Vantage™ - SQL Data Definition Language Detailed Topics*, B035-1184

## About Changing Security Constraints for a User

You can change security constraint assignments for a user in a `MODIFY USER` statement. When you specify a security constraint that is:

- Not currently assigned to the user, the constraint is added to any existing constraints
- Already assigned to the user, the new specifications replace the existing specifications
- Already assigned to the user, followed by the keyword `NULL`, the constraint assignment is dropped from the user

## Changing or Dropping Security Constraints in a `MODIFY USER` Statement

You can use the `MODIFY USER` statement to change or drop `CONSTRAINT` object assignments for an existing user.

The following example changes assignments made in [Assigning Security Constraints in a `CREATE USER` Statement](#):

```
MODIFY USER
  Joe_Smith AS
  CONSTRAINT = Classification_Level (TopSecret, Unclassified DEFAULT),
  CONSTRAINT = Classification_Country (NULL)
  CONSTRAINT = Classification_Job (Analyst) ;
```

where:

Syntax Element	Description
<code>Classification_Level(TopSecret, Unclassified DEFAULT)</code>	Raises the user classification level by replacing <code>Secret</code> with <code>TopSecret</code> , while retaining the <code>DEFAULT Unclassified</code> level.
<code>Classification_Country(NULL)</code>	Drops the <code>Classification_Country</code> constraint assignment from the user.
<code>Classification_Job (Analyst)</code>	Adds the new hierarchical category <code>Classification_Job</code> , and assigns the <code>Analyst</code> classification level.

### Note:

A new or changed security constraint assignment takes effect at the next user login.

## Related Information

For more information on `MODIFY USER` syntax and options, see:

- *Teradata Vantage™ - SQL Data Definition Language Syntax and Examples*, B035-1144
- *Teradata Vantage™ - SQL Data Definition Language Detailed Topics*, B035-1184

## Assigning Security Constraints in a CREATE PROFILE Statement

You can use the CREATE PROFILE statement to assign security constraints to a profile, and then use the CREATE USER or MODIFY USER statement to assign the profile to users.

---

### Note:

You can also use a GRANT CONNECT THROUGH statement to assign a profile to an application proxy user.

1. Create the profile and assign one or more security constraints, for example:

```
CREATE PROFILE
  profile_name AS ... ,
  CONSTRAINT = Classification_Level (Secret, Unclassified DEFAULT),
  CONSTRAINT = Classification_Country (US, UK, GER);
```

---

### Note:

See [Assigning Security Constraints in a CREATE USER Statement](#).

2. Assign the profile constraints to users by naming the profile in a CREATE USER or MODIFY USER statement. For example:

```
[CREATE|MODIFY] USER AS PROFILE = profile_name
```

A new or changed security constraint assignment takes effect at the next user login.

---

## About Changing or Dropping Security Constraints in a Profile

When changing security constraint assignments in a profile, if you specify a constraint that is:

- Not currently assigned to the profile, the constraint is added to any existing constraints.
- Already assigned to the profile, the new specification replaces the existing specification.
- Already assigned to the profile, followed by the keyword NULL, the constraint assignment is dropped from the profile.

---

### Note:

A new or changed security constraint assignment takes effect at the next user login.

---

## Adding, Changing, and Dropping Security Constraints in a Profile

Use the MODIFY PROFILE statement to change the security constraints shown in the profile created in [Assigning Security Constraints in a CREATE PROFILE Statement](#), for example:

```
MODIFY PROFILE Profile_Name AS
  CONSTRAINT = Classification_Level (TopSecret, Unclassified DEFAULT),
  CONSTRAINT = Classification_Country (NULL)
  CONSTRAINT = Classification_Job (Analyst) ;
```

where:

Syntax Element	Description
Classification_Level (TopSecret, Unclassified DEFAULT)	Raises the classification level for the constraint by replacing Secret with TopSecret, while retaining the DEFAULT Unclassified level.
Classification_Country(NULL)	Drops the Classification_Country constraint assignment from the profile.
Classification_Job (Analyst)	Adds the new hierarchical category Classification_Job, and assigns the Analyst classification level.

## Working with Security Constraint Columns

### About Security Constraint Columns in Tables

A security constraint column provides the means to assign a security label to each row in a table. The column name must match the name of a CONSTRAINT object, and each row must contain a valid column value defined in the CONSTRAINT object definition.

#### Note:

You must have the CONSTRAINT ASSIGNMENT privilege to add or drop a security constraint column for a table.

### Limits on Applying Security Constraints to Tables

A maximum of 5 security constraints can be defined for a table.

A security constraint column cannot be part of a primary index or a partitioning expression, but can participate in a secondary index.

Join indexes and hash indexes must include all constraint columns in the base table.

Limits on CHECK or UNIQUE table constraints:

- If a table protected by row level security is defined with a CHECK or UNIQUE constraint, enforcement of the constraint bypasses any security constraints defined for the table.
- You cannot define a CHECK or UNIQUE constraint on a security constraint column.

Do not define row level security on a parent or child referential integrity (RI) table.

If either or both the parent and child RI tables define security constraint columns, the security constraint UDFs are disabled and execution of any request that accesses either of the RI tables continues as if the tables had no row level security constraints.

## Related Information

For information on...	See...
CHECK and UNIQUE table constraints	<ul style="list-style-type: none"> <li>• <i>Teradata Vantage™ - SQL Data Definition Language Syntax and Examples</i>, B035-1144.</li> <li>• <i>Teradata Vantage™ - SQL Data Definition Language Detailed Topics</i>, B035-1184.</li> </ul>
Referential integrity	<ul style="list-style-type: none"> <li>• <i>Teradata Vantage™ - Database Design</i>, B035-1094.</li> <li>• <i>Teradata Vantage™ - SQL Data Definition Language Syntax and Examples</i>, B035-1144.</li> <li>• <i>Teradata Vantage™ - SQL Data Definition Language Detailed Topics</i>, B035-1184.</li> </ul>

## Creating a Table with Security Constraint Columns

You can define security constraint columns for both SET and MULTiset base tables using the CREATE TABLE statement. For example:

```
CREATE [SET|MULTISET] TABLE table_name
  (col1 integer, col2 integer, col3 varchar(30),
   Security_Classification CONSTRAINT, Job_Category CONSTRAINT)
  primary index(col1);
```

where Security\_Classification (a hierarchical level constraint) and Job\_Category (a non-hierarchical category constraint) are security constraint columns that match the names of existing security CONSTRAINT objects.

### Note:

Tables need not be defined with both types of constraints.

A constraint column inherits the data type and the NULL/NOT NULL specification from the corresponding CONSTRAINT object.

You cannot specify the COMPRESS phrase for a security constraint column.

## Related Information

For detailed CREATE TABLE syntax and options used for applying security constraints, see:

- *Teradata Vantage™ - SQL Data Definition Language Syntax and Examples*, B035-1144
- *Teradata Vantage™ - SQL Data Definition Language Detailed Topics*, B035-1184

## Adding a Security Constraint Column to a Table

To add a security constraint column, use the ADD CONSTRAINT clause of the ALTER TABLE statement, documented in *Teradata Vantage™ - SQL Data Definition Language Syntax and Examples*, B035-1144.

## Removing a Security Constraint Column from a Table

1. Remove the constraint column from any views in which the column appears using the REPLACE VIEW statement and omitting the column name in the new view.
2. Drop indexes that include the constraint column, then recreate the indexes without the column.
3. Remove the constraint column from all tables in which the column appears, using the DROP clause of the ALTER TABLE statement, documented in *Teradata Vantage™ - SQL Data Definition Language Syntax and Examples*, B035-1144.

## Working with Constraint OVERRIDE Privileges

### About OVERRIDE Privileges

When a CONSTRAINT object defines no UDF for an SQL DML operation, only users with the OVERRIDE or OVERRIDE *operation* privilege can perform the operation.

Manual entry or update of a constraint column value requires that the user have the OVERRIDE privilege to bypass the automatic use (by the enforcing UDF) of the current\_session parameter to set the constraint column value.

Users who perform load/export jobs may need OVERRIDE privileges to ensure that they can access all rows required to complete a load/export operation.

You can grant OVERRIDE privileges to users to bypass enforcement of RLS constraints. OVERRIDE privileges can apply to:

- A database, table, or column
- Some or all SQL DML operation types
- Some or all security constraints



- One or more users, roles, or external roles

---

**Note:**

The granting user must have the system level CONSTRAINT ASSIGNMENT privilege.

---

**Note:**

OVERRIDE privileges take effect at the next user request after they are granted/revoked.

---

## Granting SQL DML OVERRIDE Privileges

You can grant OVERRIDE privileges using the GRANT (SQL form) statement, documented in *Teradata Vantage™ - SQL Data Control Language*, B035-1149.

### Related Information

For information on GRANT (SQL form), see *Teradata Vantage™ - SQL Data Control Language*, B035-1149.

## Revoking SQL DML OVERRIDE Privileges

You can revoke OVERRIDE privileges using the REVOKE (SQL form) statement, documented in *Teradata Vantage™ - SQL Data Control Language*, B035-1149.

### Related Information

For information on REVOKE (SQL form), see *Teradata Vantage™ - SQL Data Control Language*, B035-1149.

## Working with Row-Level Security Effects

All objects and operations that interface with a row level security table must conform to applicable requirements and restrictions.

## Aggregate Function Requirements

When a request that references a row level security table also includes an aggregate function, for example SUM, COUNT, MAX, MIN or AVG, the aggregation is based on only the rows that are available to the requesting user.

## Required Action

To ensure that an aggregation includes all related rows, the executing user must have either:

- Security labels sufficient to overcome all SELECT access restrictions on the table rows
- The OVERRIDE SELECT CONSTRAINT privilege on all security constraints in the table

## Archive, Copy, and Restore Requirements

Users performing archives and restores must have OVERRIDE DUMP CONSTRAINT and OVERRIDE RESTORE CONSTRAINT privileges in addition to the normal DUMP and RESTORE privileges.

Archive and restore source and destination tables must define the same security constraints.

If a RESTORE of a PPI row level security table specifies the LOG WHERE option, it includes constraint columns in the resulting error table. Note that security constraints are not enforced for error row insertion, but are enforced for any subsequent access to the error table.

## Using COPY and RESTORE After a Constraint Change

---

### Note:

Changing or deleting name:value pairs in a CONSTRAINT object after tables that contain the corresponding constraint column are archived may result in incorrect or unpredictable row level protection. Addition of new name:value pairs without changing existing constraint parameters does not have these unpredictable effects.

---

If the system determines that the constraint definition has a later timestamp (due to change) than the archive copy of the data used to COPY or RESTORE a row level security table, it issues a warning: Restored/Copied data table security may be compromised because a security constraint is inconsistent between the backup and disk. The COPY or RESTORE operation is allowed to complete, but security may be compromised.

Make sure you understand the possible security effects before you COPY or RESTORE a row level security table.

## Using COPY or RESTORE to Migrate the Row Level Security Setup to Another System

When migrating the RLS setup from one system to another, a COPY of the SYSLIB database containing all the constraint UDFs automatically re-assigns new internal function IDs to the UDFs on the target system. These re-assigned IDs no longer match the IDs stored in DBC.ConstraintFunctions copied over to the target system during the migration.

Prior to migrating RLS tables to the target system, you must re-assign the UDFs in each constraint on the target system, using the procedures in [Altering or Dropping CONSTRAINT Objects](#), as follows:

1. Use ALTER CONSTRAINT to drop all UDFs from each constraint.
2. Use ALTER CONSTRAINT to re-specify the UDFs in the CONSTRAINT object. The ALTER CONSTRAINT statement then inserts the corresponding rows in DBC.ConstraintFunctions with the newly assigned function ids.

## Compound Statement SQL Requirements

Compound statement SQL requests (such as INSERT ... SELECT, MERGE ... INTO, or ALTER TABLE ... INSERT INTO) are supported for row level security tables only if all tables in the statement include the same security constraint columns.

The security constraint UDFs do not execute for security constraints in the target table, only for those in the source table.

When the system generates security constraint values for the target table:

- If the user *does not* have the OVERRIDE privilege for the operation on the table, the values for all constraint columns are taken from the corresponding columns of the source table.
- If the user *has* the OVERRIDE privilege for the operation, the constraint values are taken from the source table, unless an alternate value is provided by the request.

## Example: Setting an Alternate Constraint Value

You can provide an alternate constraint value in the SQL statement. For example:

```
MERGE INTO table1 AS tgt USING table2 AS src
  ON (tgt.col1 = src.col1) WHEN MATCHED THEN UPDATE SET Level = 2;
```

where SET Level = 2 sets the security constraint column “Level” to a value of 2 for all updated rows.

You can alternately specify:

- That a value is taken from another column. For example:

```
SET Level = src.column_name ;
```

- Values for multiple constraint columns. For example:

```
SET Level = 2, Class = 4, Job = 7, ... ;
```

## Example: Using the Results of a UDF to Set the Constraint Value

A user with the OVERRIDE CONSTRAINT privilege can specify a UDF in the request, the execution of which is inserted as the constraint column value, if it is a valid value for the constraint.

```
INSERT table1 SELECT col1, col2, ..., NonMacUDF(Level), ... FROM table2;
```

where the value for the security constraint column "Level" (in table 1) is the result of executing the UDF "NonMacUDF."

## COLLECT STATISTICS AND HELP STATISTICS Requirements

Users must have the `OVERWRITE SELECT CONSTRAINT` privilege to execute the `COLLECT STATISTICS` or `HELP STATISTICS` statement on a row level security table.

## Error Table Requirements

When you use the `CREATE ERROR TABLE` statement, all security constraint columns in the base table are automatically added to the generated error table.

You can prevent the creation of constraints columns in error tables using the `NO RLS` clause.

When an error inserts rows into an error table, the system does not enforce row level security constraints because they are already enforced on the base table during the execution of the `MERGE INTO` or `INSERT ... SELECT` operation that generated the error.

## Indexing and Partitioning Requirements

Type	Requirement/Restriction
Primary Indexes and Primary Keys	The primary index and primary key for a table cannot contain a security constraint column.
Secondary Indexes	Both unique and non-unique secondary indexes can contain one or more security constraint columns, but it is not required.
Hash Indexes and Join Indexes	Join and hash indexes can reference only a single row level security base table, and only if the index definition contains all constraint columns in the table. Recommended action: Replace affected single-table hash and join indexes to include security constraint columns.
Table Partitioning	A table partitioning expression cannot contain a security constraint column.

## Join Requirements

Joins of row level security tables, and of views that reference the tables, are supported in SQL statements, but the tables must all define the same security constraint columns.

## Load/Export Utility Requirements

Load utilities that support the creation of database target tables as part of the job script also support the definition of security constraint columns in the target tables. TPump error table and Acquisition error

tables for MultiLoad/FastLoad are not protected. However, Application error tables for MultiLoad/FastLoad are protected.

If the user does not have OVERRIDE privileges on a constraint when inserting into or updating a table, constraint column values default to the session constraint values for the user.

Granting OVERRIDE privileges to a user prevents execution of the corresponding row level security constraint UDF(s), and allows the user to specify the constraint values.

---

**Note:**

Users with OVERRIDE privileges must specify the values for non-hierarchical constraint columns as hex code instead of numeric values. See [Specifying Non-Hierarchical Constraint Values when Loading Tables](#).

---

## Recommended Action

Grant load/export utility users OVERRIDE privileges for the operations they perform on RLS tables.

## Macro Requirements

DDL and DCL statements for row level security administration are not allowed in a macro.

## Macro Privileges

The system normally executes macros based on the object-level privileges of the user or database that owns the macro.

In addition, for row level security tables, the system determines row access for a macro based on the security access level of the executing user, not the macro owner.

## OLAP Function Requirements

When a request that includes an OLAP function (such as, MAVG, CSUM, or RANK) references a row level security table, the returned answer set is based on only the rows that are available to the requesting user.

## Required Action

To ensure inclusion of all table rows in an OLAP function, ensure that the user has either:

- A security label high enough to overcome all SELECT access restrictions on table rows
- The OVERRIDE SELECT privilege on all security constraint columns for the table

## Pooled Session Requirements

When users access the database through an application that uses pooled sessions, then:

- if the application is not a trusted user, the row level security constraints for all end users derive from the application logon user. Teradata recommends to not allow users to access tables protected by row level security through non-trusted user applications, because the system cannot apply row level security to the individual end users or track their actions in access logs.
- if the application is a trusted user:
  - If the proxy user is linked to a permanent database user by using the TO PERMANENT clause in the GRANT CONNECT THROUGH statement, the row level security constraints derive from the permanent user, including the user profile.
  - If the proxy user is not linked to a permanent database user, but has a profile assigned through the GRANT CONNECT THROUGH statement, the row level security constraints derive from the assigned profile.
  - If the proxy user is not linked to a permanent database user, and does not have a profile assigned, the system cannot set a row level security constraint for the session, so attempts to access row level security tables fail.

To provide safe access to row level security tables, set up applications for trusted sessions.

- For setup instructions, see [Setting Up Trusted User Applications and Proxy Users \(Recommended\)](#).
- For information on how the system authorizes proxy user access to row level security tables, see [Session Constraint Values for Trusted User Applications and Proxy Users](#).

## Statistics Requirements

A user must have OVERRIDE privileges on a row level security table to execute the COLLECT STATISTICS, HELP STATISTICS, or SHOW STATISTICS statement if the statement includes the VALUES clause. For information about statistics-related privilege requirements, see *Teradata Vantage™ - SQL Data Definition Language Syntax and Examples*, B035-1144.

Using a SELECT statement to access statistics data in DBC views usually requires only PUBLIC privileges. However, a user can access statistics data on a row level security table (for example row count) if they have the OVERRIDE SELECT CONSTRAINT privilege on the table.

## Stored Procedure Requirements

DDL and DCL statements for row level security are not allowed in a stored procedure.

## Stored Procedure Privileges

In addition to the normal privileges required to execute a stored procedure, for row level security tables, the system determines row access for a stored procedure based on the access level for the executing user, not for the procedure owner.

## Temporary Table and Volatile Table Restrictions

Definition of row level security constraints is not supported for global temporary tables or volatile tables, and returns an error.

## Unity Requirements

When multiple Teradata Vantage systems are managed by Unity, all Vantage systems should be subject to the same row level security constraints and access privileges.

## View Requirements

When a user accesses a view, the system enforces all security constraints in the base table(s) referenced by the view, whether or not the view contains the security constraint columns.

A CREATE VIEW or REPLACE VIEW statement can reference both RLS and non-RLS base tables, but all of the RLS base tables must contain the same security constraints or the system returns an error.

If you add or change security constraints in base tables referenced by a view, and as a result the RLS tables no longer contain matching sets of security constraints, subsequent requests to access the view fail with an error.

## Determining the Session Constraint Values

The security constraint values used by the system for a session vary based on several factors. In some cases, the user must manually supply the values.

### Session Constraint Values for Permanent Database Users

When a permanent database user accesses a row protected by a security constraint, the system determines the session constraint value(s) as follows:

1. If the user profile has a corresponding security constraint assignment, the session uses the constraint value(s) specified in the profile definition.
2. If the user profile does not have a security constraint specification, the system derives the session constraint value(s) from the user definition.
3. Whether user constraint values are defined in a profile or user definition, if multiple constraint values are defined, the system determines the session value(s) as follows:

- For hierarchical (non-set) constraints, the system uses the DEFAULT value specified in the profile, or if there is no profile, in the user definition. If no DEFAULT is specified, the system uses first value listed for the constraint.
- For non-hierarchical (set) constraints, the session uses all constraint values in the profile, or if there is no profile, in the user definition. No DEFAULT can be specified.

**Note:**

Requesting users can use SET SESSION CONSTRAINT to change the session constraint value to any constraint value specified in the profile or user definition.

4. If neither the user profile nor the user definition contains a security constraint assignment, the constraint value for the session is NULL, and the user can access rows controlled by a security constraint only if assigned the necessary OVERRIDE privileges.
5. If a user has OVERRIDE privileges on the object and the operation being performed, the system ignores constraints assigned in the profile or user definition. The session derives security constraint values in one of the following ways:
  - For simple inserts, the user must supply the constraint value(s).
  - For compound statements, for example, INSERT ... SELECT or MERGE, the system derives constraint value(s) from the security constraint columns in the source table.
  - The user can use SET SESSION CONSTRAINT to specify constraint values.

## Session Constraint Values for Directory Users

- A directory user who is mapped to a permanent Teradata Vantage user inherits the permanent user privileges and constraint assignments.
- A directory user who is not mapped to a permanent database user, but who uses Sign-On As to log on as a permanent user, inherits the user privileges and constraint assignments.
- Directory users who are mapped to multiple constraint value sources (users or profiles) must use the `user=user_name` or `profile=profile_name` option in the logon string to specify the source for default constraint values and OVERRIDE privileges.
- Directory users can use the SET SESSION CONSTRAINT option to access any mapped or inherited constraint assignments.
- For directory users with no mappings and only PUBLIC or EXTUSER privileges, the session constraint value is NULL, and the user cannot access row level security tables.

## Session Constraint Values for Application Pooled Users

For mainframe and middle-tier application pooled sessions (not trusted sessions), users inherit the row level security privileges for the mainframe or application logon user, or its profile, if used. System processing of constraint values is the same as for any other permanent database user.



## Session Constraint Values for Trusted User Applications and Proxy Users

### SET QUERY\_BAND Processing

If the trusted user application submits an initial SET QUERY\_BAND statement (a standard operation for trusted user applications) the system resets the session constraint values to an empty set, and takes the following actions:

- If the statement does not set a proxy user, the session uses the default constraint value for the trusted user according to the normal permanent user constraint hierarchy. See [Session Constraint Values for Permanent Database Users](#).
- If the statement sets a proxy user, and the user is also a permanent database user, the system uses the default constraint value for the permanent user, according to the normal permanent user constraint hierarchy, including profile. See [Session Constraint Values for Permanent Database Users](#).
- If the application proxy user does not have a profile or the profile has no security constraint, the system is unable to set a session constraint value and also will not accept a SET SESSION CONSTRAINT statement. If the application proxy user has a profile with a security constraint, the session is set to that constraint.

### Session Constraint Values in OVERRIDE Sessions

The user conducting a session that accesses an RLS table may need OVERRIDE privileges if:

- The CONSTRAINT object associated with a constraint column in the table does not define a UDF for the SQL operation.
- Accessing table rows or setting row level constraint values cannot be done effectively with the assigned user defaults.

---

#### Note:

For additional information on OVERRIDE privileges, see [Working with Constraint OVERRIDE Privileges](#).

---

When an OVERRIDE is in effect for a session:

- The OVERRIDE privileges bypasses enforcement of the affected UDFs.
- If the operation is an INSERT or UPDATE, the user SQL must supply the value(s) to be entered into the named constraint column in the table, either using data from the source table or values specified in a SET SESSION CONSTRAINT statement.

**Note:**

Loading tables with non-hierarchical constraints in an OVERRIDE session requires special handling. See [Example: Loading Tables with User OVERRIDE Privileges](#).

## Specifying Non-Hierarchical Constraint Values when Loading Tables

To provide the capacity for the largest number of values in the smallest space, non-hierarchical constraint values are defined in RLS tables as bit positions rather than as numeric values. The bit positions are represented in the table as hex code.

The method by which hex-encoded constraint values are loaded into an RLS table varies depending on the presence or absence of the OVERRIDE privilege.

### Example: Loading Tables without User OVERRIDE Privileges

When a user without the OVERRIDE privilege performs an INSERT or UPDATE on an RLS table the system converts the session constraint value(s), defined as byte(n) in the assigned user constraint, to hex code and loads them into the table.

For example, assume that:

1. A BYTE(1) non-hierarchical constraint named Countries is defined with these values:

- USA: 1
- UK: 2
- Canada: 3

2. User U1 is assigned the constraint.

```
CONSTRAINT = Countries (USA, UK, Canada)
```

3. User U1 defines a table to include the Countries constraint column:

```
CT rls_table (x INT, Countries CONSTRAINT);
```

4. The security policy defined in the related INSERT UDF does not alter the session constraint for the user.

At logon, the session constraint value for user U1 is calculated by the system as follows:

Constraint Value	Bit Position	Binary Value
USA:1	1	1
UK:2	2	1
Canada:3	3	1

Constraint Value	Bit Position	Binary Value
Not applicable	4	0
	5	0
	6	0
	7	0
	8	0

The system evaluates the assigned user constraints and calculates a binary string to represent each set of non-hierarchical values, in the example above, 11100000, which translates to the hex string 'E0'xb.

If user U1 inserts a row into the table rls\_tbl, the system automatically enters the calculated hex value 'E0'xb in the Countries CONSTRAINT column for the table.

## Resetting the Session Constraint

Users have the option to change the default constraint values available for a session using the SET SESSION CONSTRAINT statement.

Based on [Example: Loading Tables without User OVERRIDE Privileges](#), U1 might reset the session default Countries constraint:

```
SET SESSION CONSTRAINT = Countries (UK, Canada);
```

The session constraint value is changed from the 'E0'xb shown above to '60'xb (hex representation of 01100000). Subsequent inserts during the session default to '60'xb for the Countries constraint column.

### Note:

You can display the hex string for the default constraint values using the HELP SESSION CONSTRAINT statement.

## Example: Loading Tables with User OVERRIDE Privileges

When a user with OVERRIDE privileges uses an INSERT or UPDATE to load a table that has one or more non-hierarchical constraint columns, the system does not calculate and insert the needed hex values. The user must supply the hex values for each non-hierarchical constraint column as part of the load script. For example:

1. Calculate the binary string for the bit position that represents the constraint values, as shown in [Example: Loading Tables without User OVERRIDE Privileges](#).
2. Convert the binary string to a hex value.
3. Load the hex values into the source data column(s).

4. Specify the column(s) containing the hex data in the load script.
5. When you load the destination table, the hex values are entered into the constraint columns.

**Note:**

The database checks all constraint column input data for validity, by internally converting the hex string to binary and checking the indicated bit position against dictionary rows in DBC.ConstraintValues, before inserting them into the target table.

## Using SET SESSION to Change the Session Security Constraint Value

Users assigned more than one value for a security constraint can use SET SESSION CONSTRAINT to replace the default value with another assigned value. For information about SET SESSION CONSTRAINT, see *Teradata Vantage™ - SQL Data Definition Language Syntax and Examples*, B035-1144.

## Using HELP SESSION to Investigate Session Constraint Values

A user who is unsure of the session constraint value(s) can use the HELP SESSION CONSTRAINT command to get a list of the security constraints and associated constraint values for the session.

## About Row-Level Security and Bulk Table Loads

By default, moving data between protected and unprotected tables is restricted. Source and target tables are required to have identical constraints to perform a bulk load to or from a table with RLS restrictions. This restriction guarantees all constraint column values are valid before moving them. If an invalid value is encountered, the bulk load terminates.

The system can be configured to allow client loader applications and SQL statements to move data between protected and unprotected tables.

To enable this functionality, contact the Teradata Support Center.

## About Session Constraints and Bulk Table Loads

By default, row inserts into RLS-protected tables done by users with OVERRIDE privileges require values to be specified for the constraint columns. This prevents these users from doing bulk loads through client loaders because data is read from external files.

The system can be configured to allow a user with OVERRIDE privileges and non-null session constraints to perform simple and bulk inserts without explicitly specifying constraint values for the constraint columns.

To enable this functionality, contact the Teradata Support Center.

## Using Access Logging with Row Level Security

### About Access Logging for Row-Level Security

In addition to logging the privilege checks on discretionary rights, as described in [Monitoring Database Access](#), you can also monitor RLS-related privilege checks.

RLS Monitoring Category	SQL Statements Monitored
Security constraint UDF enforcement actions	<ul style="list-style-type: none"> <li>• INSERT</li> <li>• SELECT</li> <li>• UPDATE</li> <li>• DELETE</li> </ul>
SQL OVERRIDE privilege check	<ul style="list-style-type: none"> <li>• OVERRIDE INSERT</li> <li>• OVERRIDE SELECT</li> <li>• OVERRIDE UPDATE</li> <li>• OVERRIDE DELETE</li> </ul>

### Beginning RLS Access Logging

Use the BEGIN LOGGING statement to enable access logging of row level security privilege checks. For details, see *Teradata Vantage™ - SQL Data Definition Language Syntax and Examples*, B035-1144.

### Example: Logging Denials of Access Attempts

Log all denials of attempts to access any table with the Security\_Classification constraint, for example:

```
BEGIN LOGGING DENIALS
ON EACH INSERT, SELECT
FOR CONSTRAINT <name>;
```

### Example: Logging the First Operation on a Specified Object

Log the first insert to the employee\_record table in the secure\_database, for a request, whether or not the insert is denied, for example:

```
BEGIN LOGGING
ON FIRST INSERT
FOR CONSTRAINT <name>
ON TABLE <dbname>.<tblname>;
```

## Example: Logging Denials for a Specified User on a Specified Object

Log all denials of Joe Smith to archive the employee\_record table in the secure\_database, to track any rows not included in the archive, for example:

```
BEGIN LOGGING DENIALS
ON EACH DUMP
FOR CONSTRAINT Security_Classification
BY JoeSmith
ON TABLE secure_database.employee_record;
```

## Ending RLS Access Logging

You can selectively end some or all access logging for a security constraint using the END LOGGING statement. For details, see *Teradata Vantage™ - SQL Data Definition Language Syntax and Examples*, B035-1144.

## About Constraint-Related System Tables and Views

### DBC Security Constraint Tables and Views

Table and Related Views	Description
<ul style="list-style-type: none"> <li>• DBC.SecConstraints</li> <li>• DBC.SecConstraintsV and VX</li> </ul>	Contains information about all security constraint objects defined for the system.
<ul style="list-style-type: none"> <li>• DBC.ConstraintFunctions</li> <li>• DBC.ConstraintFunctionsV</li> </ul>	Identifies the UDFs for each constraint object. <b>Note:</b> Details for the UDFs are in DBC.UDFInfo.
<ul style="list-style-type: none"> <li>• DBC.ConstraintValues</li> <li>• DBC.ConstraintValuesV</li> </ul>	Defines the set of valid name:value pairs for each constraint object.
<ul style="list-style-type: none"> <li>• DBC.AsgdSecConstraints</li> <li>• DBCUsrAsgdConstraintsV and VX</li> <li>• DBC.ProfileAsgdConstraintsV and VX</li> </ul>	Defines the name:value pairs assigned to each user or profile, by constraint object name.
<ul style="list-style-type: none"> <li>• DBC.TVFieldsV</li> <li>• DBC.ColumnsV</li> </ul>	Contains a row for every column in a database table, view, or index. The ConstraintID column indicates whether a column is a security constraint.

Table and Related Views	Description
DBC.AccLogRulesV	Indicates whether a logging rule is for a security constraint related privilege, and contains a column for each row level security privilege.

## Other DBC Tables with Security Constraint Columns

Table	Description
DBC.TVFields	Contains a row for every column in a database table, view, or index. The ConstraintID column indicates whether a column is a security constraint.
DBC.TVM	The MACFlag column contains a value of either: <ul style="list-style-type: none"> <li>• 'Y' for RLS tables</li> <li>• 'N' or NULL for non-RLS tables</li> </ul>
DBC.AccessRights	Contains abbreviations that define user privileges in the database. The table includes the following abbreviations for row level security privileges: <ul style="list-style-type: none"> <li>• SD - CONSTRAINT DEFINITION</li> <li>• SA - CONSTRAINT ASSIGNMENT</li> <li>• OI - OVERRIDE INSERT</li> <li>• OS - OVERRIDE SELECT</li> <li>• OU - OVERRIDE UPDATE</li> <li>• OD - OVERRIDE DELETE</li> <li>• OA - OVERRIDE DUMP</li> <li>• OR - OVERRIDE RESTORE</li> </ul>
DBC.SessionTbl	Provides information about a session based on the session ID. A session can be labeled with up to 6 non-set and 2 set constraints. The SessionTbl identifies each active constraint for each session by: <ul style="list-style-type: none"> <li>• Constraint ID: A four-byte internal identifier used by the system for recovery. You can determine the associated constraint name by joining with DBC.SecConstraints.</li> <li>• Constraint Value: Either a numeric value for hierarchical constraints, or a byte string for non-hierarchical constraints. You can determine the name that corresponds to a constraint value by joining with DBC.ConstraintValues.</li> </ul>
DBC.AccLogRuleTbl	Indicates whether a logging rule is for a security constraint related privilege, and contains a column for each row level security privilege.
DBC.AccLogTbl	Indicates whether a column level access rule applies to a security constraint column.

## Restricted Access to Statistics in DBC Views

Accessing DBC views that contain statistics data normally requires only PUBLIC privileges. However, because users might possibly derive protected information about row level security tables from statistics data, access to statistics on row level security tables is restricted.

The following views do not return statistics information for row level security tables:

- DBC.ColumnStatsV
- DBC.IndexStatsV
- DBC.MultiColumnStatsV

To provide convenient access to statistics on row level security tables, users having **OVERRIDE SELECT CONSTRAINT** privileges on the DBC.StatsTbl table can create custom views.

---

**Note:**

The DBC.StatsTbl.StatsType column carries a value of P (protected) to identify tables that have the statistics restrictions.

---

## Finding Security Constraint Assignments

You must remove security constraint assignments before you can alter or drop a **CONSTRAINT** object. Use the following example SQL requests to determine which database objects have constraint assignments.

### Finding Tables and Indexes with a Security Constraint

1. Query the DBC.Dependency table to find tables that have a security constraint column:

```
SELECT database1name, object1name from dbc.dependency
where object2name='constraint_name';
```

2. Run the **SHOW TABLE** command for each table returned by the query.

The system displays the standard **CREATE TABLE** for the table, including any indexes that are defined on the table.

### Finding Views that Include a Security Constraint

```
SELECT databasename, tvmlname from dbc.tvml, dbc.dbase
where
tablekind='V' and
tvml.databaseid=dbase.databaseid and
tvmlid in (select tableid from dbc.tvfields where constraintid in
(select constraintid from dbc.secconstraints
where constraintname='constraint_name'));
```

### Finding Users or Profiles with an Assigned Constraint

For users:



```
SELECT UserName from DBC.UsrAsgdSecConstraintsV
where
ConstraintName='constraint_name'
```

For profiles:

```
SELECT ProfileName from DBC.ProfileAsgdSecConstraintsV
where
ConstraintName='constraint_name'
```

## Determining Whether a Column is a Security Constraint

You can execute a HELP COLUMN statement against a column to find out whether the column is a security constraint. See *Teradata Vantage™ - SQL Data Definition Language Syntax and Examples*, B035-1144.

## Related Information

For information on...	See...
Using security-related tables and views	<a href="#">Investigating Database Access Attempts.</a>
Querying data dictionary tables and views	<i>Teradata Vantage™ - Data Dictionary</i> , B035-1092.

# Implementing Teradata Secure Zones

Secure zones separate the access to data from the database administration duties in an exclusive database hierarchy inside a Teradata Vantage system.

## Teradata Secure Zones Overview

The Teradata Secure Zones feature allows you to create one or more exclusive database hierarchies, called zones, within a single Teradata Vantage system. Access to the data in each zone and the database administration is handled separately from the Vantage system and from other zones.

Secure zones are useful in situations where the access to data must be tightly controlled and restricted. You can also use secure zones to support some regulatory compliance requirements for the separation of data access from database administration duties.

For example, consider the following use of secure zones. Suppose you have a multinational company or conglomerate enterprise with many subsidiaries. You can create a separate zone for each of the subsidiaries. If your company has divisions in different countries, you can create separate zones for each country to restrict data access to the personnel that are citizens of that country. Your corporate personnel can manage and access data across multiple zones while the subsidiary personnel in each zone have no access to data or objects in the other zones. A system-level zone administrator can manage the subsidiary zones and object administration can be done by either corporate DBAs or zone DBAs, as required.

With Teradata Secure Zones, you can ensure the following:

- Users in one subsidiary have no access or visibility to objects in other subsidiaries.
- Corporate-level users may have access to objects across any or all subsidiaries.

Another typical scenario is the case of cloud companies that host multiple data customers as tenants. Companies that offer cloud-based database services can host multiple tenants within a single Vantage system, using zones to isolate the tenants from each other as if they were running on physically segregated systems. Zone DBAs can administer the objects in their own zone as required. The tenant zones can be managed by a system-level zone administrator.

With Teradata Secure Zones, you can ensure the following:

- Users in a tenant zone have no access or visibility to objects within other zones.
- Users in a tenant zone cannot grant rights on any objects in the zone to any other users, databases, or roles of other zones within the system.

## Secure Zone Objects

Zone objects are created, modified, and dropped in the same way as any other Teradata Vantage object; an object exists only inside its own zone. Tables, triggers, and macros that are created inside a zone are zone objects. Objects such as roles and profiles, which are not qualified by database names, are only accessible

inside the zone in which they are created. Security constraints are an exception. Security constraints that are created outside a zone can be assigned to zone users. Security constraints that are created inside a zone can be assigned to users who are outside the zone.

## Secure Zone User Types

The following list describes the different types of users that are associated with a zone:

- zone creator

Creates zones and assigns a user or a database as the zone root. Zone creators cannot access the objects or data in the zones that they create. Any user who has the ZONE rights with the WITH GRANT OPTION privilege can grant CREATE ZONE and DROP ZONE privileges.

Only the zone's creator can add a root and primary DBA to a zone or drop a root and primary DBA from a zone.

If the zone creator creates the zone with a user as root, then the zone creator must have DROP USER privilege on that user. Once the root is assigned to a zone, all privileges on the root user are revoked from the zone creator.

If the zone creator creates the zone with a database as root, then the zone creator must have CREATE USER privilege on the database that becomes a root. Once the root is assigned to a zone, all privileges except CREATE USER privilege on the root database are revoked from the zone creator.

A zone creator may grant zone access to users or roles that exist outside of the zone and is also responsible for revoking access to the zone.

A zone creator must have CREATE ZONE and DROP ZONE privileges. A zone creator cannot be dropped until the zone itself is dropped.

Zone creators who create multiple zones function as system-level zone administrators for those zones.

- zone root

The empty database or user on which the zone creator creates a zone.

A zone creator creates the zone and associates a database or a user as its root. The zone root database or user must be empty. It cannot have any objects, users, databases, roles, or profiles associated with it. It also cannot have privileges on any other user. Similarly, no user should have any privileges on root except for the zone creator, owner of the root, and creator of the root.

If the zone root is a database, the zone creator must subsequently assign a primary DBA to the zone. If the zone root is a user, that user automatically becomes the primary DBA for the zone.

- primary zone DBA

A primary zone DBA acts as the zone's database administrator.

The zone creator creates the primary zone DBA. The primary zone DBA can create zone users, databases, objects, and zone-level objects such as roles and profiles.

- zone user

A permanent database user with privileges in a zone. A zone user is a user that is created by another user in the zone, under the hierarchy of the zone root. Zone users are created using the existing CREATE USER syntax. A zone user cannot be a zone guest of another zone.

Only zone users can grant privileges on database objects within the zone to zone guests.

- zone guest

A zone guest is a role or user that is located outside of the zone but is granted privileges to create and access objects in the zone where he is a guest. A zone can have many zone guests and a user or a role can be a guest of more than one zone.

Zone guests cannot grant privileges on zone objects to other users.

To make an external LDAP user a zone guest, the zone creator can use the GRANT ZONE syntax to grant zone access privilege to an external role. External users that log on with that role are able to access the zone objects that they have privileges on.

Only the zone users can grant privileges on database objects in a zone to zone guests. Zone users cannot grant privileges to zone guests with the WITH GRANT OPTION privilege.

Zone guests with the required privileges can create users, databases, and TVM objects inside the zone but they cannot add another guest to the zone.

Zone guests can create views, triggers, and macros on the zone objects in their perm space.

## Privileges in Teradata Secure Zones

User Type	Who creates them and the privileges that they have	Privileges they can and cannot grant, and users that they can create
zone creators	<p>A Vantage user who has the following rights with the WITH GRANT privilege may explicitly grant the following privileges to zone creators:</p> <ul style="list-style-type: none"> <li>• CREATE ZONE</li> <li>• DROP ZONE</li> <li>• DROP USER privilege on the user who becomes the zone root</li> <li>• CREATE USER privilege on the database that becomes the zone root</li> </ul>	<p>Zone creators cannot grant any privileges to zone users.</p> <p>Zone creators can create zone guests from users or roles that were previously created outside the zone.</p>
primary zone DBA	<p>The zone creator either:</p> <ul style="list-style-type: none"> <li>• Creates a zone with a user as the root, which by default makes that user the primary DBA and implicitly grants them all privileges.</li> <li>• Creates a zone with a database as the root, and then creates a user who is the primary DBA by using CREATE USER FROM <i>database_name</i> syntax to implicitly grant all privileges to that user.</li> </ul>	<p>The primary DBA can do the following:</p> <ul style="list-style-type: none"> <li>• Create zone users, databases, and TVM objects inside the zone using existing DDL syntax.</li> <li>• Grant privileges to zone guests. No privileges can be granted to a zone guest with the WITH GRANT OPTION privilege.</li> </ul>

User Type	Who creates them and the privileges that they have	Privileges they can and cannot grant, and users that they can create
zone user (includes the primary DBA)	A primary DBA or any previously created zone user creates other users in a zone under the hierarchy of zone root, using the existing CREATE USER syntax.	Zone users can create zone users, databases, and TVM objects using existing DDL syntax. Only zone users can grant privileges on database objects in a zone to zone guests. No privileges can be granted to a zone guest with the WITH GRANT OPTION privilege.
zone guest	The zone creator creates zone guests using the GRANT ZONE syntax. A zone guest cannot access zone objects unless a zone user explicitly grants them privileges to create objects or grants them privileges to access existing objects in the zone where they are guests.	Zone guests with the required privileges can do the following: <ul style="list-style-type: none"> <li>• Create zone users, databases, and TVM objects inside the zone using existing DDL syntax.</li> <li>• Create views, triggers, macros and so on, on the zoned objects in their perm space.</li> </ul>

## Implementing Teradata Secure Zones

To set up and use a secure zone, the following tasks need to be performed:

1. An existing user with the appropriate privileges should grant zone creation privileges to an existing user or should create a user and grant the privileges needed to create zones to that user.

See [Privileges in Teradata Secure Zones](#).

For more information, see *Teradata Vantage™ - SQL Data Control Language*, B035-1149.

2. The zone creator must create an empty user or database to be zone root or assign an existing user or database to be or zone root. If the zone creator creates the zone with a user as root, then the zone creator must have DROP USER privilege on that user. If the zone creator creates the zone with a database as root, then the zone creator must have CREATE USER privilege on the database that becomes the root.

For information about creating users and databases, see *Teradata Vantage™ - Database Administration*, B035-1093.

3. The zone creator must create a secure zone.

See [Creating a Zone](#).

4. If the creator creates the zone with ROOT as a user, skip this step. If the creator creates the zone with ROOT as a database, then assign a primary DBA to the zone.

See [Adding a Primary DBA to a Zone](#).

5. The zone creator can add zone guests to the zone, if desired.

See [Adding Zone Guests to a Zone](#).

6. The primary DBA can create at least one zone user in the zone using the existing CREATE USER syntax. Any of these zone users can then optionally create other zone users.

See [Creating Zone Users in a Zone](#).

7. If the zone creator added zone guests, a zone user must grant them the desired privileges.

For information about granting privileges, see GRANT (SQL Form) in *Teradata Vantage™ - SQL Data Control Language*, B035-1149.

8. If you want to map the users in the zone to a proxy user or to preexisting directory users, you can either set up a proxy user or set up roles.

See [About Using a Proxy User or Directory User in a Zone](#).

## Creating a Zone

A zone creator must have CREATE ZONE privilege to create a zone. The zone creator then uses the following syntax to create a zone:

```
CREATE ZONE zone_name ROOT root_name ;
```

The *root\_name* must be an existing user or database that has no preexisting objects, descendants, roles, profiles, or grants.

### Note:

You can create a secure zone without a ROOT, and use ALTER ZONE later to add a ROOT database or user. You can also use ALTER ZONE to delink the ROOT from a secure zone before you drop the zone.

For information about the CREATE ZONE and ALTER ZONE syntax, see *Teradata Vantage™ - SQL Data Definition Language Syntax and Examples*, B035-1144.

## Adding a Primary DBA to a Zone

Only the zone creator can add the primary DBA user to a zone. This user cannot be a part of any other zone.

If the zone creator creates the root of the zone on a user, that user automatically becomes the zone's primary DBA. This user must be an existing user who has no preexisting objects, descendants, roles, profiles, or grants.

If the zone creator creates the root of the zone on a database, the zone creator must use the following syntax to create the primary DBA user:

```
CREATE USER DBA_user_name FROM database_name [...];
```

The *database\_name* is the name of the database that the zone was created on.

You can add other valid syntax after the AS clause. For information about the CREATE USER syntax, see *Teradata Vantage™ - SQL Data Definition Language Syntax and Examples*, B035-1144.

## Creating Zone Users in a Zone

Any zone user with CREATE USER privilege can create other zone users. To create zone users, you use the existing standard Vantage CREATE USER syntax. For example, you can use the following syntax:

```
CREATE USER  user_name  AS ... ;
```

You can add other valid syntax after the AS clause. For information about CREATE USER syntax, see *Teradata Vantage™ - SQL Data Definition Language Syntax and Examples*, B035-1144.

## Adding Zone Guests to a Zone

Only the zone creator can add guests to a zone. PUBLIC cannot be a zone guest. To add one or more zone guests, use a syntax statement in the form of one of the following statements:

```
GRANT ZONE  zone_name_list  TO  user_name_list  ;
GRANT ZONE  zone_name_list  TO  role_name_list  ;
GRANT ZONE  zone_name_list  TO  user_name_list, role_name_list  ;
```

The user or role cannot be a member of any other zone. Zone creators cannot add themselves as zone guests.

Zone guests have no privileges on zone objects until a zone user grants them access privileges.

## About Using a Proxy User or Directory User in a Zone

You cannot map a directory user to a permanent user with zone privileges. You must use roles instead. External users and application proxy users cannot be zone users, but they can be zone guests. After the zone creator creates a zone guest, any zone user can grant access privileges on zone objects to the guests by granting them to external roles or proxy user roles.

For more information about roles and proxy users, see [Setting Up Trusted User Applications and Proxy Users \(Recommended\)](#) and [Working with Roles for Proxy Users](#).

## Granting Privileges to Zone Users and Zone Guests

Any zone user, including the primary DBA, can grant access privileges to other zone users or to zone guests. For example:

```
GRANT  privileges  ON  object_name  TO  user_name_list  ;
```

**Note:**

Zone users cannot use the WITH GRANT OPTION privilege to grant access privileges on zone objects to a zone guest.

## Dropping the Root From a Secure Zone

Only the zone creator can delink the root of a zone. The root must not have any objects, descendants, roles, or profiles assigned to it. You must remove all zone guests from a zone before you delink the zone's root.

You can use the following syntax:

```
ALTER ZONE zone_name DROP ROOT ;
```

For information about ALTER ZONE syntax, see *Teradata Vantage™ - SQL Data Definition Language Syntax and Examples*, B035-1144.

## Dropping Zone Users From a Secure Zone

Zone users can be removed using the existing standard Vantage DROP USER syntax.

You can use the following syntax:

```
DROP USER user_name ;
```

## Revoking Access From Zone Guests

Only the zone creator can remove zone guests. You can use the following syntax, as appropriate:

```
REVOKE ZONE zone_name_list FROM user_name_list ;
REVOKE ZONE zone_name_list FROM role_name_list ;
REVOKE ZONE zone_name_list FROM user_name_list, role_name_list ;
```

## Dropping a Zone

Only the zone creator can drop a zone. Before a zone can be dropped, the root, primary DBA, all zone users, all zone database objects, and all zone guests, must be dropped. You can then use the following syntax to drop the zone itself:

```
DROP ZONE zone_name ;
```



## Example Scenario

Suppose that you want to create two zones for an automotive company to use, one for the sales staff and one for the service personnel.

First, you need to create a zone creator. The DBC user, for example, can create a user named Dana and grant that user the privileges that a zone creator needs, as follows:

```
CREATE USER dana ;
GRANT CREATE ZONE, DROP ZONE TO dana WITH GRANT OPTION ;
```

Dana can now create two databases on which to create the zones, as follows:

```
CREATE DATABASE jcl_sales FROM dana AS PERM=100000 ;
CREATE DATABASE jcl_service FROM dana AS PERM=100000 ;
```

Now Dana creates the two zones. The first command creates the zone and assigns the ROOT directly. The second command creates the zone and then uses an ALTER statement to assign the ROOT.

```
CREATE ZONE jcl_sales_zone ROOT jcl_sales ;
CREATE ZONE jcl_service_zone ;
ALTER ZONE jcl_service_zone ADD ROOT jcl_service ;
```

Dana then creates a primary DBA for each zone, as follows:

```
CREATE USER jcl_sales_dba FROM jcl_sales
AS
  PASSWORD=jcl_sales_dba
  PERM = 0 ;
CREATE USER jcl_service_dba FROM jcl_service
AS
  PASSWORD=jcl_service_dba
  PERM = 0 ;
```

Now the zone DBAs can create zone users and grant rights to them, as follows:

```
CREATE USER jcl_sales_bob FROM jcl_sales
AS
  PASSWORD=jcl_sales_bob
  PERM = 0 ;
GRANT ALL ON jcl_sales TO jcl_sales_bob WITH GRANT OPTION ;
CREATE USER jcl_service_bill FROM jcl_service
AS
  PASSWORD=jcl_service_bill
```

```

    PERM = 0 ;
GRANT ALL ON jcl_service TO jcl_service_bill WITH GRANT OPTION ;

```

The new users can create tables in their respective zones, as follows:

```

CREATE TABLE jcl_sales.Auto_Inv (Vin_no INT, Auto_Desc Varchar(25));
CREATE TABLE jcl_service.Part_Inv (Part_no INT, Part_Desc Varchar(25)) ;

```

The zone creator can create users to be made as zone guests and grant them access to the zones, as follows:

```

CREATE USER jcl_sales_guest FROM dana
AS
    PASSWORD=jcl_sales_guest
    PERM = 0 ;
GRANT ZONE jcl_sales_zone TO jcl_sales_guest ;
CREATE USER jcl_service_guest FROM dana
AS
    PASSWORD=jcl_service_guest
    PERM = 0 ;
GRANT ZONE jcl_service_zone TO jcl_service_guest ;

```

(Or the zone creator could use GRANT ZONE to make existing users into zone guests.)

A zone user, in this case the zone DBA, grants the zone guests the rights needed to access objects in the zone, as follows:

```

GRANT SELECT ON jcl_sales.Auto_Inv TO jcl_sales_guest ;
GRANT SELECT ON jcl_service.Part_Inv TO jcl_service_guest ;

```

## Setting Up a Zone for Testing

Following are the steps that you can take to set up a zone for testing (a sandbox).

1. A DBA with the appropriate privileges creates a database.

For information about how to create a database, see *Teradata Vantage™ - Database Administration*, B035-1093 and *Teradata Vantage™ - SQL Data Definition Language Syntax and Examples*, B035-1144.

2. A DBA with CREATE ZONE privileges creates a zone with the new database as its root.

```

CREATE ZONE zone_name ROOT root_db_name ;

```

3. The creator of the zone creates a primary DBA from the root database.

```
CREATE USER DBA_user_name FROM database_name ... ;
```

4. The creator of the zone creates zone guests from the existing users that want to test in the zone. The following SQL syntax can be used to create zone guests:

```
GRANT ZONE zone_name TO user_or_role_name_list ;
```

5. The primary DBA for the zone grants the zone guests any privileges that they need to do experiments, prototyping, and so on in the sandbox zone. For example:

```
GRANT TABLE ON root_db_name TO list_of_sandbox_zone_users;
```

6. When testing is completed, the zone creator should drop the zone. Before a zone can be dropped, the following tasks must be performed:
  - The primary DBA should drop all zone database objects.
  - The zone creator (administrator) should revoke zone access privilege from the zone guests.
  - The zone creator (administrator) should drop the primary DBA.
  - The zone creator (administrator) should drop the zone root.

## Security Considerations

### Using External Authentication and Authorization with Zones

You can externally authenticate zone users using LDAP or Kerberos, but you cannot externally authorize zone users. The logon for any externally authorized user that is mapped to a zone user fails.

Zone guests can be externally authenticated and authorized. You can grant zone access to external roles that are mapped to groups in which zone guests are members. For external authorization to access objects within a zone to work, you must grant both of the following:

- Zone access to the zone guests
- The required discretionary access control privileges on the zone objects to external roles that are active for the zone guests' sessions.

### Using Trusted Sessions and Proxy Users with Zones

A trusted user, a permanent proxy user, a proxy role, and a proxy profile in a GRANT CONNECT THROUGH statement cannot belong to a secure zone in a Trusted Sessions configuration.

Application proxy users can access objects in a zone if you grant both zone access and the required discretionary access control privileges on the zone objects to the specified proxy role.

## Table and View Privileges

If you have implemented Secure Zones, zone users can see the information for objects in their zone in DBC views, but they cannot see the information for any objects outside their zone. If a non-zone user needs to see information for all the zones, you can grant ZONE OVERRIDE privilege. Users with ZONE OVERRIDE privilege can see information from all zones. You can also grant SELECT privilege on a specific view, but users with SELECT privileges on a view obtain different result sets, depending on their zone membership or non-zone status.

For more information, see *Teradata Vantage™ - Data Dictionary*, B035-1092.

## Using Logging to Monitor Zones

Logging syntax within zones, including BEGIN LOGGING, END LOGGING, BEGIN QUERY LOGGING, REPLACE QUERY LOGGING, END QUERY LOGGING, and SHOW QUERY LOGGING, is unchanged. Logging and query logging privileges within zones are the same as those for non-zone query logging.

Zone DBAs can use zone-level logging to monitor their zones and can use zone-level query logging to monitor the queries of the users in their zones. Non-zone users other than the DBC user cannot enable query logging on objects across all zones unless they have the ZONE OVERRIDE privilege.

The Vantage DBC user has access to all dictionary data in the database without any zone restrictions. If other users from outside a zone need to collect system-wide dictionary data, the DBC user must grant them the ZONE OVERRIDE privilege. The DBC user receives the ZONE OVERRIDE privilege during system initialization and can grant this privilege, without the WITH GRANT OPTION privilege, to other users. This privilege applies to dictionary table and view access only; it does not apply to data table access.

For information about using GRANT ZONE OVERRIDE and REVOKE ZONE OVERRIDE, see *Teradata Vantage™ - SQL Data Control Language*, B035-1149.

## Access Logging

Any zone user with EXECUTE privilege on the DBC.AccLogRule macro can begin and end logging on objects within the zone, but not on objects outside their zone. Zone users cannot create an ON ALL log rule and they cannot end system-level logging rules that were created by non-zone users.

Users who have system-wide logging privileges can begin and end logging on all non-zone database objects. They cannot create log rules on zone database objects. They cannot end zone-level logging rules that zone users created. Users with system-wide logging privileges are the DBC user and any non-zone users, including zone guests, who have ZONE OVERRIDE privilege and EXECUTE privilege on the DBC.AccLogRule macro.

## Query Logging

Users with EXECUTE privilege on DBQLAccessMacro can begin and end query logging. The following table summarizes query logging in Teradata Secure Zones:

User and Privileges	Logging Scope
Zone users with EXECUTE privilege on DBQLAccessMacro	<p>Can begin and end query logging on objects in their zone. They cannot begin and end query logging on objects outside of their zone.</p> <p>When they SHOW LOGGING, they can see only rules created by zone users.</p> <p>They cannot end system-level logging rules created by users with system-level logging privileges.</p>
The DBC user	<p>Can begin and end query logging on all objects on the system, regardless of zones. When they SHOW LOGGING, they can see all rules.</p>
Non-zone users (including zone guests) with the ZONE OVERRIDE privilege and with EXECUTE privilege on DBQLAccessMacro	<p>Can begin and end query logging on all objects on the system, regardless of zones.</p> <p>When they SHOW LOGGING, they can see all rules.</p>
Non-zone users (excluding DBC) with DBQLAccessLog privilege but without ZONE OVERRIDE privilege	<p>Can begin and end query logging on objects that are not in a zone. When they SHOW LOGGING, they can see only rules created by non-zone users.</p>

# TDGSS Configuration Files, Valid Settings, and Editing Guidelines

The following topics contain information and valid settings for the TDGSS configuration files (TdgssUserConfigFile.xml and TdgssLibraryConfigFile.xml).

The TDGSS configuration files are organized in these major sections:

- [Configuration File Header](#): Defines the file as a library or user configuration file.
- Security attributes legal values (valid values):
  - [AlgorithmName](#): The algorithms available for use by TDGSS
  - [KeyLength, KeyLengthP](#): Allowable encryption key lengths (in bits) for algorithms
  - [Mode](#): Available encryption modes for algorithms
  - [Padding](#): Supported encryption padding types for algorithms
  - [InterfaceType](#): Supported security interface types for algorithms
  - [AlgorithmType](#): The type indicates how an algorithm is used
  - [Mechanism Properties](#): Properties define the function of the containing mechanism
- [Quality of Protection \(QOP\)](#)
- [Mechanism Configurations](#)

## Configuration File Header

The Header section identifies the file type (Library or User) and the file version.

### Example: TDGSS Library Configuration File

The header for the library configuration file appears in the following default form:

```
<Header
  Version="1"
  ConfigFileType="Library">
</Header>
```

### Example: TDGSS User Configuration File

The header for the user configuration file appears in the following default form:

```
<Header
  Version="1"
  ConfigFileType="User">
</Header>
```

## AlgorithmName

The AlgorithmName class lists the encryption algorithms available for use by TDGSS.

Algorithm Name	Description
NONE	Indicates that the algorithm does not support encryption.
AES	Advanced Encryption Standard algorithm. A symmetric block cipher that uses the rijndael algorithm to process data blocks of 128 bits, using cipher keys of 128, 192, and 256 bits.
MD5	Message Digest Algorithm. TDGSS uses MD5 for digital signature applications where it must compress a large file in a secure manner.
<ul style="list-style-type: none"> <li>• SHA1</li> <li>• SHA256</li> <li>• SHA512</li> </ul>	Secure Hashed Algorithm TDGSS uses secure hashed algorithms to compute a condensed representation of a message or data file.
Diffie-Hellman	An encryption key exchange algorithm invented by W. Diffie and M.E. Hellman

## Example: Algorithm Names

Supported algorithm names appear in the AlgorithmName section of the TdgssLibraryConfigFile.xml in the following XML format:

```
<AlgorithmName
  NONE="0"
  BLOWFISH="1"
  AES="2"
  MD5="3"
  SHA1="4"
  DIFFIE_HELLMAN="5"
  <!-- DH is alias for DIFFIE-HELLMAN -->
  DH="5"
  SHA256="6"
  SHA512="7">
</AlgorithmName>
```

### Note:

Blowfish is a deprecated algorithm, and is available only to support legacy mechanisms and QOP 0.

## KeyLength, KeyLengthP

The KeyLength class lists supported Diffie-Hellman encryption key lengths. The key lengths shown in the [Quality of Protection \(QOP\)](#) must use values from this list.

### Note:

Some of the supported key lengths may not be active for a particular release of Teradata Vantage. Active key lengths are those used in QOP configurations defined in the TDGSS library configuration file. To determine which encryption key lengths are active for the current release, see [Quality of Protection \(QOP\)](#).

2048 is the only supported key length for KeyLengthP.

Key Length	Description
K0	Encryption not supported.
K128	The numeric portion of the values shown are equal to the length, in bits, of the associated encryption keys. Some key lengths are not used by currently available algorithms.
K192	
K256	
K416	
K448	
K512	
K1024	
K2048	

## Example: Key Lengths

Supported encryption key lengths appear in the KeyLength section of the TdgssLibraryConfigFile.xml in the following XML format:

```
<KeyLength
  K0="0"
  K128="128"
  K192="192"
  K256="256"
  K416="416"
  K448="448"
  K512="512"
  K1024="1024"
```



```

        K2048="2048">
    </KeyLength>
    <KeyLengthP
        K2048="2048">
    </KeyLengthP>

```

**Note:**

The numbers associated with each key length are used by the underlying mechanism code to represent the key length in bits.

## Mode

The Mode class defines the encryption mode types supported by TDGSS. The Mode attribute values shown in the [Quality of Protection \(QOP\)](#) must appear on this list.

**Note:**

If you use Java Cryptography Extensions (JCE), the mode types must match the mode types supported by TDGSS.

Mode Type	Description
NONE	Encryption not enabled
CBC	Cipher-block Chaining An operational mode for a block cipher, in which a set number of bits is encrypted as a single unit or block with a cipher key applied to the entire block.
CFB	Ciphertext Feedback An operational mode for block cipher, in which plaintext values are encrypted and transferred one at a time.
ECB	Electronic Code Book An operational mode for a block cipher in which each possible block of plaintext has a defined corresponding ciphertext value. For compatibility with legacy Teradata systems only.
OFB	Output Feedback An operational mode for a block cipher that permits encryption of differing block sizes, but the output of the encryption block function is the feedback instead of the cipher text. For compatibility with legacy Teradata systems only.
GCM	Galois/Counter Mode An operational mode for block cipher that uses universal hashing over a binary Galois field to provide authenticated encryption.
CCM	Counter with Cipher Block Chaining-MAC

Mode Type	Description
	An operational mode for block cipher that combines counter mode encryption and CBC-MAC authentication.
CTR	Counter Mode An operational mode for block cipher that turns a block cipher into a stream cipher. It generates the next key stream block by encrypting successive values of a "counter."

This list of supported modes does not change unless you install a new release of TDGSS that supports a different set of encryption modes.

## Example: Modes

Supported encryption modes appear in the Mode section of TDGSS Legal Values in the following default xml form:

```
<Mode
  NONE="0"
  CBC="1"
  CFB="2"
  ECB="3"
  OFB="4"
  GCM="5"
  CCM="6"
  CTR="7">
</Mode>
```

### Note:

The numbers associated with each mode type are used by the underlying mechanism code to represent the mode type.

## Padding

The Padding class lists the encryption padding types supported by TDGSS. The [Quality of Protection \(QOP\)](#) sections of TDGSS use the encryption padding types described in this list.

### Note:

If you run any Java Cryptography Extensions (JCE) on your system, the encryption padding must be one of the types supported by TDGSS.

Padding Type	Description
No Padding	The algorithm does not use encryption.
OAEPWithDIGESTAndMGFPadding [Deprecated]	Optimal Asymmetric Encryption Padding with DIGEST function and Mask Generation function, for Java.
PKCS1 Padding [Deprecated]	Public Key Cryptography Standard 1.
PKCS5 Padding	Public Key Cryptography Standard 5
SSL3 Padding [Deprecated]	Secure Sockets Layer A protocol that allows two parties to authenticate and establish a session key to encrypt the rest of the session.

This list of supported encryption padding types only changes if you install a new release of TDGSS that supports a different set of padding types.

## Example: Padding Types

Supported encryption padding types appear in the Padding section of TdgssLibraryConfigFile.xml as follows:

```
<Padding
  NoPadding="0"
  OAEPWithDIGESTAndMGFPadding="1"
  PKCS1Padding="3"
  PKCS5Padding="4"
  SSL3Padding="5">
</Padding>
```

### Note:

The numbers associated with each padding type are used by the underlying mechanism code to represent the padding type.

## InterfaceType

The InterfaceType class lists the encryption interfaces supported by TDGSS. The security mechanisms defined in the [Mechanism Configurations](#) section must use an encryption interface from this list.

Interface Type	Description
GSS	Standard interface for the UNIX operating system.
SSPI	Standard Windows client interface.

Interface Type	Description
Teradata	Default Teradata interface.
Custom	Reserved for future use. Allows for the creation and use of a custom security interface when the feature is implemented.
Negotiate	The security mechanism is negotiated between the client and server.

This list of supported encryption interface types only changes if you install a new release of TDGSS that supports a different set of interface types.

Supported interface types appear in the InterfaceTypes section of the TdgssLibraryConfigfile.xml in the following default form:

```
<InterfaceType>
  gss
  sspi
  teradata
  custom
  negotiate
</InterfaceType>
```

## AlgorithmType

The Algorithm Type class lists the encryption functions supported by TDGSS.

The following table describes the supported algorithm functions:

Algorithm Type	Function
Confidentiality	Encrypts transmitted data to insure confidentiality.
Integrity	Checks a message to ensure that none of the data was changed or lost.
Key Exchange	Secure exchange of encryption keys between initiator and acceptor.

This list of supported algorithm functional types only changes if you install a new release of TDGSS that supports a different set of functional types.

Supported encryption functions appear in the AlgorithmType section of TdgssLibraryConfigFile.xml in the following default xml form:

```
<AlgorithmType>
  Confidentiality
  Integrity
  KeyExchange
</AlgorithmType>
```

## Mechanism Properties

All TDGSS mechanisms contain a group of properties that define mechanism function, based on the values of the properties.

Property values have the following characteristics.

- All property values must appear in the TDGSS configuration files in double quotation marks, that is: *property="value"*
- Many properties, such as MechanismEnabled, are common to all mechanisms. Others are only meaningful for some mechanisms.
- All properties have a default value.
- Many properties allow only a yes or no value to indicate whether or not the mechanism supports the property function.
- For some properties, you do not need to edit the default property values. For others, the default values are hard coded and editing is not allowed.
- Many editable properties carry a default value of "", which may indicate that the property:
  - Has no value.
  - Defers to a default value stored in the system.

---

### Note:

You may need to configure a value other than "" to enable certain security functions.

---

Be sure you fully understand the function of a property, before you consider changing the default property value.

For information about which mechanisms support each property, see [Supporting Mechanisms](#).

For detailed guidelines on editing property values, see [About Editing Configuration Files](#).

## Modifying Mechanism Properties Without a TPA Reset

The following can be modified without a TPA reset:

- Any attribute or property whose name begins with "Ldap" for KRB5 and LDAP
- MechanismEnabled property for KRB5, LDAP, JWT, and PROXY
- AuthorizationSupported property for KRB5 and LDAP
- LDAP Service ID and password with no impact to user LDAP logons
- The following properties in the PROXY mechanism:
  - CertificateFile
  - PrivateKeyFile
  - PrivateKeyPassword
  - PrivateKeypasswordProtected

- CACertFile
- CACertDir
- SigningHashAlgorithm
- Any JWT mechanism property whose name begins with "JWT"
- All canonicalizations including the lightweight authorization structures

The following configuration changes still require a tpareset:

- Changes to any mechanism property not mentioned above require a tpareset
- QoP configuration
- Local or global policy configuration, including service name changes
- TDNEGO and SPNEGO

---

**Note:**

The run\_tdssconfig utility indicates when a TPA reset is required.

---

## Supporting Mechanisms

The following table shows which mechanisms support each property.

- Editable means the mechanism supports the property and the setting for the property may be edited.
- Keep Default means the mechanism supports the property, but do not change the default setting for the property.
- A blank cell means the mechanism does not support the property; for example, TD2 does not support the AuthenticationSupported property.
  - If no mechanisms for a property are selected, the property does not have any supporting mechanisms, is reserved for future use, or is unused. See the section for the specific property for details.

---

**Note:**

Teradata recommends viewing the Editing Guidelines for each property.

---

Basic Functional Properties	JWT	KRB5	LDAP	PROXY	SPNEGO	TDNEGO	TD2
<a href="#">AuthenticationSupported</a>	Keep Default	Keep Default	Keep Default	Keep Default	Keep Default	Keep Default	
<a href="#">AuthorizationSupported</a>		Editable	Editable		Editable	Keep Default	
<a href="#">GenerateCredentialFromLogon</a>		Keep Default	Keep Default	Keep Default		Keep Default	Keep Default
<a href="#">NegotiationSupported</a>					Keep Default	Keep Default	
<a href="#">SingleSignOnSupported</a>		Keep Default			Keep Default	Keep Default	

## A: TDGSS Configuration Files, Valid Settings, and Editing Guidelines

Confidentiality Properties	JWT	KRB5	LDAP	PROXY	SPNEGO	TDNEGO	TD2
<a href="#">DHKeyP and DHKeyG</a>	Editable		Editable	Editable			Editable
<a href="#">VerifyDHKey</a>							Editable
Directory Identification and Search Properties	JWT	KRB5	LDAP	PROXY	SPNEGO	TDNEGO	TD2
<a href="#">LdapBaseFQDN [Deprecated]</a>		Editable	Editable				
<a href="#">LdapClientDebug</a>		Keep Default	Keep Default				
<a href="#">LdapClientDeref</a>		Keep Default	Keep Default				
<a href="#">LdapClientRebindAuth</a>		Editable	Editable				
<a href="#">LdapClientReferrals</a>		Editable	Editable				
<a href="#">LdapCredentialsUPN</a>			Editable				
<a href="#">LdapGroupBaseFQDN</a>		Editable	Editable				
<a href="#">LdapServerName</a>		Editable	Editable				
<a href="#">LdapServerPort [Deprecated]</a> [Deprecated]							
<a href="#">LdapServerRealm [Deprecated]</a>		Editable	Editable				
<a href="#">LdapSystemFQDN</a>		Editable	Editable				
<a href="#">LdapUserBaseFQDN</a>		Editable	Editable				
<a href="#">UseLdapConfig</a>		Editable	Editable				
JWT Support Properties	JWT	KRB5	LDAP	PROXY	SPNEGO	TDNEGO	TD2
<a href="#">JWTClientTlsCACertDir</a>	Editable						
<a href="#">JWTClientUseTls</a>	Editable						
<a href="#">JWTDecryptionKeyFile</a>	Editable						
<a href="#">JWTDynamicKey</a>	Editable						
<a href="#">JWTKeyCacheRefreshTime</a>	Editable						
<a href="#">JWTKeyDirectory</a>	Editable						
<a href="#">JWTRestAPIMaxTimeAllowed</a>	Editable						
<a href="#">JWTRestAPITimeLimit</a>	Editable						
<a href="#">JWTSkewTime</a>	Editable						
<a href="#">JWTTokenExchange</a>	Editable						
<a href="#">JWTVerificationKeyFile</a>	Editable						
LDAP Binding Properties	JWT	KRB5	LDAP	PROXY	SPNEGO	TDNEGO	TD2
<a href="#">LdapClientMechanism</a>			Editable				
<a href="#">LdapServiceBindRequired</a>			Editable				
<a href="#">LdapServiceFQDN</a>		Editable	Editable				

## A: TDGSS Configuration Files, Valid Settings, and Editing Guidelines

<a href="#">LdapServicePassword</a>		Editable	Editable				
<a href="#">LdapServicePasswordFile</a>		Editable	Editable				
<a href="#">LdapServicePasswordProtected</a>		Editable	Editable				
<b>LDAP Policy Properties</b>	<b>JWT</b>	<b>KRB5</b>	<b>LDAP</b>	<b>PROXY</b>	<b>SPNEGO</b>	<b>TDNEGO</b>	<b>TD2</b>
<a href="#">LdapNetworkBaseFQDN</a> (configured in <LdapConfig> section in TdgssUserConfigFile.xml)							
<a href="#">LdapPolicyFQDN</a>			Editable				
<a href="#">MechanismIgnoreQOP</a>	Keep Default	Keep Default	Keep Default	Keep Default	Keep Default		Keep Default
<b>LDAP Protection Properties</b>	<b>JWT</b>	<b>KRB5</b>	<b>LDAP</b>	<b>PROXY</b>	<b>SPNEGO</b>	<b>TDNEGO</b>	<b>TD2</b>
<a href="#">LdapAllowUnsafeServerConnect</a>		Editable	Editable				
<a href="#">LdapClientTlsCACert</a>		Editable	Editable				
<a href="#">LdapClientTlsCACertDir</a>		Editable	Editable				
<a href="#">LdapClientTlsCert</a>		Editable	Editable				
<a href="#">LdapClientTlsCipherSuite</a>		Editable	Editable				
<a href="#">LdapClientTlsCRLCheck</a>		Editable	Editable				
<a href="#">LdapClientTlsKey</a>		Editable	Editable				
<a href="#">LdapClientTlsRandFile</a>		Editable	Editable				
<a href="#">LdapClientTlsReqCert</a>		Editable	Editable				
<a href="#">LdapClientSASLSecProps</a>			Editable				
<a href="#">LdapClientUseTLS</a>		Editable	Editable				
<b>Mechanism Status Properties</b>	<b>JWT</b>	<b>KRB5</b>	<b>LDAP</b>	<b>PROXY</b>	<b>SPNEGO</b>	<b>TDNEGO</b>	<b>TD2</b>
<a href="#">DefaultMechanism</a>	Editable	Editable	Editable		Editable	Editable	Editable
<a href="#">DefaultNegotiatingMechanism</a>					Keep Default	Editable	
<a href="#">MechanismEnabled</a>	Editable	Editable	Editable	Editable	Editable	Editable	Editable
<a href="#">MechanismRank</a>		Editable	Editable		Editable	Editable	Editable
<b>Operational Properties</b>	<b>JWT</b>	<b>KRB5</b>	<b>LDAP</b>	<b>PROXY</b>	<b>SPNEGO</b>	<b>TDNEGO</b>	<b>TD2</b>
<a href="#">AnonymousAuthentication</a> (Reserved for future use)							
<a href="#">ConfidentialityDesired</a>	Keep Default	Keep Default	Keep Default	Keep Default	Keep Default	Keep Default	Keep Default
<a href="#">CredentialUsage</a>	Keep Default	Keep Default	Keep Default	Keep Default	Keep Default		Keep Default
<a href="#">DelegateCredentials</a> (unused)							
<a href="#">DesiredContextTime</a> (Reserved for future use)							



<a href="#">DesiredCredentialTime</a> (Reserved for future use)							
<a href="#">IntegrityDesired</a>	Keep Default	Keep Default	Keep Default	Keep Default	Keep Default	Keep Default	Keep Default
<a href="#">MutualAuthentication</a>		Editable			Editable	Keep Default	
<a href="#">OutOfSequenceDetection</a>	Keep Default	Keep Default		Keep Default	Keep Default	Keep Default	
<a href="#">ReplayDetection</a>	Keep Default	Keep Default		Keep Default	Keep Default	Keep Default	
<a href="#">TeradataKeyTab</a>		Editable					
<b>Unity Support Properties</b>	<b>JWT</b>	<b>KRB5</b>	<b>LDAP</b>	<b>PROXY</b>	<b>SPNEGO</b>	<b>TDNEGO</b>	<b>TD2</b>
<a href="#">CACertDir</a>				Editable			
<a href="#">CACertFile</a>				Editable			
<a href="#">CertificateFile</a>				Editable			
<a href="#">PrivateKeyFile</a>				Editable			
<a href="#">PrivateKeyPassword</a>				Editable			
<a href="#">PrivateKeyPasswordProtected</a>				Editable			
<a href="#">ProxySupported</a>				Keep Default			
<a href="#">SigningHashAlgorithm</a>				Editable			

## Special Handling for New Properties

Teradata occasionally adds a new mechanism or mechanism property to the TdgssLibraryConfigFile.xml. These mechanisms and properties perform default functions without additional configuration; however, if you need to edit a new mechanism or a new property value, first check to see if it is in TdgssUserConfigFile.xml and uncomment it. If the mechanism or property is not in the user config file, you must move it to TdgssUserConfigFile.xml (copy/paste it from TdgssLibraryConfigFile.xml) before you can make the change. See, [About Editing Configuration Files](#).

## Basic Functional Properties

The following properties, from AuthenticationSupported through SingleSignInSupported, describe basic capabilities of the mechanisms in which they occur.

### AuthenticationSupported

This property indicates whether or not the mechanism supports external (non-Teradata) authentication of users.

## Valid Settings

The only valid setting for this property is yes, the preset default value.

## Editing Guidelines

The value of this property is only an indication of mechanism function. The system may ignore edits to the default value or the edits may cause a problem.

## AuthorizationSupported

This property determines whether the mechanism supports directory authorization of users.

## Valid Settings

Setting	Description
yes	The database accepts external authorization of user privileges. Yes is the default for LDAP and TDNEGO.
no	The database authorizes user privileges internally. No is the default for all mechanisms, except for LDAP and TDNEGO.

## Editing Guidelines

- AuthorizationSupported must be set to yes if the directory authorizes user privileges, that is, if directory users are mapped to database objects.
- You can edit this property in the TDGSS version of the TdgssUserConfigfile.xml on the database, and in the the Unity version of the configuration file on the Unity server (for Unity information, see *Teradata® Unity™ Installation, Configuration, and Upgrade Guide for Customers*, B035-2523). If the database configuration is set to yes, the Unity configuration must be set to yes; if the database is set to no, Unity can be set to yes or no.
- When the value of this property is set to yes, the gateway looks for authorization information from the directory specified in the LdapServerName property specified for the mechanism.
- When the value of this property is set to no, the gateway ignores any authorization information in the directory. This setting allows you to authenticate directory users with LDAP, while authorizing user privileges in the database.
- To set this property to yes for KRB5 or SPNEGO, you must copy the LdapServerName property from the TdgssLibraryConfigFile.xml into the TdgssUserConfigFile.xml, and then configure the property value.
- Do not modify the AuthorizationSupported property for the TDNEGO mechanism because it does not use this property. TDNEGO passes the entire logon string to the underlying mechanisms, which means TDNEGO always supports authorization. Note, the underlying mechanism may not support authorization.

## GenerateCredentialFromLogon

The GenerateCredentialFromLogon property tells TDGSS which part of the logon string to look in for user credential information. Some CLlv2-based third-party applications or scripts, such as a BTEQ script, do not allow use of the .logdata statement in a logon string. If GenerateCredentialFromLogon is set to yes, the gateway first looks in the .logdata statement for credential information. If it does not find a .logdata statement, the gateway uses the credential information found in the .logon statement.

### Note:

GenerateCredentialFromLogon appears only in the TdgssLibraryConfigFile.xml, but it is fully functional and you do not need to add it to the TdgssUserConfigurationFile.xml.

### Valid Settings

The only valid setting for this property is yes, the preset default value.

### Editing Guidelines

Do not edit the value of the GenerateCredentialFromLogon property.

## NegotiationSupported

The NegotiationSupported property is boolean. It signifies that a mechanism is actually a pseudo mechanism that negotiates between the client and the server to find an appropriate mechanism to use. All mechanisms that fit this description will have NegotiationSupported set to yes.

The SPNEGO mechanism is no longer considered a negotiating mechanism, because it can only be used for .NET clients that are connecting to Vantage servers using the Kerberos mechanism. Therefore SPNEGO will have the NegotiationSupported property set to no.

### Default Property Value

TDGSS sets the value of this property to yes for TDNEGO and to no for all other mechanisms in the Vantage server, Unity server, and client configuration files.

### Valid Settings

Setting	Description
yes	The mechanism is a pseudo mechanism that negotiates between the client and the server to find an appropriate mechanism to use; for example, the TDNEGO mechanism is set to yes.
no	The mechanism is not a pseudo mechanism that negotiates between the client and the server.

### Editing Guidelines

- This property should never be edited.

- Any mechanism that has the NegotiationSupported property set to yes must also have at least one NegotiatedMechanism ObjectId set for that mechanism, otherwise an error is reported by the tdgssconfig executable.

## SingleSignOnSupported

This property indicates whether or not the mechanism supports Single Sign-on.

### Valid Settings

The preset value is the only valid setting for this property.

Setting	Description
yes	The mechanism supports single sign-on (the default for mechanisms that support the property).
no	The mechanism does not support single sign-on (the default for mechanisms that do not support the property).

### Editing Guidelines

- This property only indicates mechanism capability. The system ignores the value of this property if it is not the preset value.
- Do not use this property to enable or disable single sign-on. See [About External Authentication](#) for instructions.
- Do not modify the TDNEGO setting for this property.

## Confidentiality Properties

### DHKeyP and DHKeyG

The Diffie-Hellman encryption key (DH Key) is made up of two values, P and G, which allows two hosts to create and share a secret key to ensure the confidentiality of the encryption key exchange between initiator and acceptor.

The P and G parameters are both public to the system. P is a large prime number, and G is chosen so it is a small primitive root of P, that is, G is a primitive root if and only if  $G^{(P-1)/q} \bmod P > 1$  for all prime divisors q of P-1.

The basic calculation is:  $G^X \bmod (P)$ .

The variable X is a private number that each user keeps to themselves. Each uses their private key X to calculate their public key, such that:

```
PublicKeyUser1 = G^x mod (P)
PublicKeyUser2 = G^y mod (P)
```

Each user transmits their Public key so that User 2 has PublicKeyUser1 and User 1 has PublicKeyUser2.

User1 computes:  $K1 = (\text{PublicKeyUser2})^x \bmod (P)$

User2 computes:  $K2 = (PublicKeyUser1)^y \bmod (P)$

**Note:**

There are two sets of DH keys: DHKeyP/DHkeyG and DHKeyP2048/DHKeyG2048. The first pair is 640 bit, which is only supported for compatibility with pre-TD 14.0 systems. In cases where the client and server are both TD 14.0 or higher, the 640 bit keys are never used.

## Default Property Value for DHKeyP2048

This 2048 bit DHKeyP is supplied with Teradata Vantage (represented in hex code):

DHKeyP2048="8AB3F86E8D374B782F31DAD5F27D6AFDA30150C11A20CF6346712AE2D2C6B70A5B79  
D45D4C0C232A065B207B121B2C33E147B5983C38A1087F272703B0B839CBA6F71C5D0EB51EC89093  
4EACF2C7DD2A1DF6F55E89B145A0359D35EF8FB6C561E157B13FF927A35E69963648614902B1034E  
F71197F545DEF3236244EADAE0689E624CF1245953630AE042BD797C4025E37C51D9F6CBDA0B2278  
FA7D5CA2D9CA930BE2968330C811A4BA4D0845333C0D62E3EE742154F6B62F2951CD8C73C43B5AA1  
C7819DEF1D7C9314411E465F8E4796666594AADE0AEB3F1256E5719E7AE54DD34FFDA949634E4A29  
3C5BC60AF258BB9FE558086E83B3DD3D7491966DEE93"

## Default Property Value for DHKeyG2048

This 2048 bit DHKeyG is supplied with Teradata Vantage (represented in hex code):

[illegible]

## Default Property Values for Legacy DHKeyG and DHKeyP

[illegible]

## Editing Guidelines

- In high security environments, you can replace the preset key and/or rotate keys periodically to minimize the chance that the key can be compromised.
- If you edit DHKeyP2048, you should also edit DHKeyG2048.
- You can edit this property only on all nodes and on the Unity server. Also see [Coordinating Mechanism Property Values for Unity](#).
- You can use any DH Key with a supported key length. See [KeyLength](#), [KeyLengthP](#).

## VerifyDHKey

This property indicates whether the mechanisms requests that the system perform an integrity check to verify that the DH Key is the same for both initiator and acceptor.

### Default Property Value

This property is preset to no for all supporting mechanisms, indicating that the system does not verify the integrity of the DH Key. This setting minimizes the effect of message encryption on system performance.

## Editing Guidelines

- To provide the user with early notification of whether or not the encryption is successful, you can set this property to yes for supported mechanisms. A yes setting results in a modest performance degradation of the encryption sequence, because the initiator and acceptor must exchange an extra message.
- Editing this property has meaning only for the client TdgssUserConfigFile.xml.
- Teradata recommends that you use the default “no” setting.

## Directory Identification and Search Properties

You can use directory identification and search properties to configure additional search capabilities, which TDGSS can use to enhance user authentication and authorization.

## LdapBaseFQDN [Deprecated]

The LdapBaseFQDN property specifies the FQDN of a directory object that contains directory users and groups, and provides the search base for locating user and group objects.

### Note:

This property is deprecated in favor of the LdapGroupBaseFQDN and LdapUserBaseFQDN properties.

- If you set the value of either LdapGroupBaseFQDN or LdapUserBaseFQDN (preferred), the value overrides the value of LdapBaseFQDN.

- If you do not set the values of either LdapGroupBaseFQDN or LdapUserBaseFQDN, the system uses the value of LdapBaseFQDN.

## Valid Settings

- "" (default), that is, the property does not specify a search base
- The FQDN of an object that contains all directory users and groups (not recommended)

## Editing Guidelines

- LdapSystemFQDN appears by default only in the LDAP mechanism. If the AuthorizationSupported=yes for KRB5 or SPNEGO, you can add LDAPSystemFQDN to the TDGSS configuration file for the mechanism and specify a value. See [Changing the TDGSS Configuration](#).
- If the property is set to the default "", and the directory is Active Directory, ADAM, AD LDS, or any uncertified LDAPv3-compliant directory, you must do the following:
  - For LDAP authentication only, configure the LdapUserBaseFQDN.
  - For LDAP authorization, configure the LdapGroupBaseFQDN.
- Edit this property on database nodes and on the Unity server, if used. Also see [Coordinating Mechanism Property Values for Unity](#).

## LdapClientDebug

You can use the LdapClientDebug property to assist the Teradata Support Center in debugging LDAP directory issues.

The LdapClientDebug property applies to all authentication mechanisms that support referral chasing.

## Valid Settings

The default property value is 0, the preset default.

Do not attempt to reset this value without Teradata Support Center assistance.

## Editing Guidelines

This property is not user-configurable. Do not attempt to reset this value without Teradata Support Center assistance.

## LdapClientDeref

The LdapClientDeref property tells the directory server what to do with any referral objects it encounters in the directory information tree.

The LdapClientDeref property applies to mechanisms that support referral chasing.

**Note:**

Do not change the default setting for this property without first contacting Teradata Support Center for assistance.

**Valid Settings**

Setting	Description
never (default)	Do not chase referrals of any kind to bind the user, even if LdapClientReferrals is set to on (preferred).
always	Chase referrals only if the object containing the referral is in the search base.
finding	Chase referrals only if the object that contains the referral is the search base.
searching	Chase any referral to any object that is subordinate to the search base. Return any objects found in the referred directory as if they came from the local directory

**Editing Guidelines****Note:**

Do not attempt to reset this value without Teradata Support Center assistance.

- To set a value, you must manually add this property to the TDGSS configuration file on needed mechanisms. See [About Editing Configuration Files](#).
- Edit this property on database nodes and on the Unity server, if used. Also see [Coordinating Mechanism Property Values for Unity](#).
- Use the default setting, never, for all external authentication mechanisms to prevent referral chasing, unless you have a good reason to follow referrals.
- If LdapClientReferrals is set to yes, use the LdapClientDeref property to tell the directory how to handle the referrals it finds. Also see [LdapClientReferrals](#).

**LdapClientRebindAuth**

When the LdapClientReferrals property is set to chase referrals, LDAP establishes a new connection to the directory server and continues the searches on that connection, based on the referral.

The LdapClientRebindAuth property tells the authentication mechanism how to bind to the new (referred) connection.

**Valid Settings**

Setting	Description
yes (default)	The authentication mechanism uses the user credential info to authenticate the new search connection before searching.



Setting	Description
no	The authentication mechanism does not authenticate the new search connection with user credential info, but instead uses an anonymous connection to do the search.

## Editing Guidelines

- The LdapClientRebindAuth appears by default in the library configuration file, for the LDAP mechanism only.
- To set a value, you must manually add this property to the TDGSS configuration file for each mechanism that uses it. See [About Editing Configuration Files](#).
- Edit this property on database nodes and on the Unity server, if used. Also see [Coordinating Mechanism Property Values for Unity](#).
- Edit this property to specify the how TDGSS should handle referrals, when the LdapClientReferrals property is set to yes.

## LdapClientReferrals

The setting of the LdapClientReferrals property determines whether TDGSS chases any referrals the directory presents. The directory makes referrals when the user DN that TDGSS sends to the directory is ambiguous, that is, when the DN conforms to more than one directory naming context. When the value of LdapClientReferrals is set to on, TDGSS searches the directory for each naming context (referral) it receives.

The LdapClientReferrals property applies to any mechanism that searches for users in a directory.

### Note:

Do not enable this property without first contacting Teradata Customer Service for assistance.

For additional information on chasing referrals, see [Optimizing Directory Searches](#).

## Valid Settings

Setting	Description
off (default)	TDGSS does not chase referrals
on	TDGSS chases referrals

## Editing Guidelines

- To set a value, you must manually add this property to the TDGSS configuration file on the mechanisms that require it. See [About Editing Configuration Files](#).
- Edit this property on database nodes and on the Unity server, if used. Also see [Coordinating Mechanism Property Values for Unity](#).

- The recommended setting is off.
- If you use referral chasing, you should review the function of the LdapClientDeref property, which tells the directory how to process referrals, and configure it, if required.

## LdapCredentialsUPN

For some logon forms, tells TDGSS how to interpret the portion of the logon string that identifies the user.

For example, for the BTEQ logon:

```
.logon system/user,password
```

If the *user* specification is in the form “a@b” or a/b” or “a\b”, the setting of LdapCredentialsUPN tells the system whether to treat the entire *user* specification as an Authcid or to use the special characters to construct a UPN.

Also see [LDAP Logon Format Examples](#).

### Default Property Value

The default value is yes.

### Valid Settings

Setting	Description
yes (default)	The system treats the user specification as a UPN.
no	The system interprets the entire user specification as a user Authcid.

### Editing Guidelines

- If the LdapCredentialsUPN property is absent or set to yes (the default), the system treats the user specification as a UPN, which must conform to the rules of IETF 1964.

#### Note:

When considered as a UPN, the *user* specification, as shown in the example in the introductory text for this property (above), must appear in the logon as: “a\b” or “a/b” or “a@b”, where the added backslash (\) character informs the system how to handle the character that follows.

- If the CredentialsUPN property is set to no, the system disregards the special characters and considers the entire user specification to be a string representing the user Authcid.

## LdapGroupBaseFQDN

This property helps LDAP narrow the directory search during user authorization, when directory groups are mapped to one or more Teradata Vantage external roles.

For additional information on optimizing directory searches, see [Configuring LDAP Properties to Narrow the Search Base](#).

## Valid Settings

- "" (default), that is, the property does not specify an object to narrow the search
- The FQDN of a directory object that contains the group objects that map to Teradata role objects in the directory.

## Editing Guidelines

- This property appears by default in the library configuration file for the LDAP mechanism. You can manually add it to the TDGSS configuration file for other supporting mechanisms, if needed. See [About Editing Configuration Files](#).
- You should specify a value for LdapGroupBaseFQDN if the AuthorizationSupported property for the mechanism is set to yes.
- For best results, set the value of LdapGroupBaseFQDN to the FQDN of an object one level higher in the directory tree than the highest level group object that maps to Teradata Vantage external role objects.
- If you do not edit the default value for this property (""), LDAP uses the value of the LdapBaseFQDN to search the directory, however, because LdapBaseFQDN is deprecated, this approach is not recommended.
- Edit this property on database nodes and on the Unity server, if used. Also see [Coordinating Mechanism Property Values for Unity](#).

## LdapServerName

The value of the LdapServerName property tells TDGSS which directory to use for authentication and authorization of directory users.

## Valid Settings

- "", that is, \_ldap.\_tcp (default)
- A valid URI or DNS SRV RR specification.

## Sample Configuration for a LDAP Uniform Resource Identifier

```
"resource_identifier [ ... ]"
```

*resource\_identifier*

```
scheme://server[:port]/
```

**Note:**

The resource identifiers must be separated by spaces. The entire string, including double quotation marks, cannot exceed 256 characters.

**Syntax Elements*****scheme***

A valid URL scheme: ldap, ldaps, gc, or gcs.

***server***

The FQDN or IP address of the directory server.

Do not use a server IP address with Active Directory and DIGEST-MD5. DIGEST-MD5 is deprecated.

For fail-over protection, you can specify multiple directory servers, beginning with the primary server. TDGSS selects servers from the list in the order configured. If a server is unavailable, TDGSS tries the next server on the list.

For configuring systems connected to multiple directory services, see [Creating the <LdapConfig> Section in the TdgssUserConfigFile.xml](#).

***port***

[Optional] The LDAP service port.

Default behavior: The system uses the default port designation for the specified scheme, for example:

- ldap (389)
- ldaps (636)
- gc (3268)
- gcs (3269)

**Configuring DNS SRV Resource Records (RRs)**

You can configure the LdapServerName property to tell LDAP to select an authenticating directory at random, from the DNS domain SVR RRs, if the RRs conform to IETF RFC 2782.

For details, see the following table or go to: <http://www.ietf.org/rfc/rfc2782.txt>.

Property Component and Value	Description
Specify the default domain: _scheme._tcp or "".	Directs TDGSS to select a directory from those listed in the SRV RRs for the default domain.

Property Component and Value	Description
Specify a non-default domain: <code>_scheme._tcp.domain_name</code>	Directs TDGSS to select a directory from those listed in SRV RRs for the domain you specify.
Configure a site-aware domain name, for example: <code>_ldap._tcp.site_name._sites.domain</code>	Directs TDGSS to select a directory that is local to the Teradata Vantage system to which the user logs on, from the SRV RRs for the domain. Also see <a href="#">Configuring LDAP for Site-Aware Authentication</a> .

## Editing Guidelines

- LdapServerName appears by default in the LDAP mechanism. You must add LdapServerName to KRB5 and SPNEGO and specify a value if AuthorizationSupported=yes.
- You must configure this property for any mechanism with AuthorizationSupported =yes.
- Edit this property on database nodes and on the Unity server, if used.
- If the default associated with the domain scheme is not the correct port, you can use the URI method to specify another port.
- You can use the \_ldaps.\_tcp or \_gcs.\_tcp scheme to automatically enable SSL protection.
- If the directory is not Active Directory, and you specify \_ldaps.\_tcp or \_gcs.\_tcp, you may need to manually register the location of the directory service in the DNS. For Active Directory, the process is automatic.
- You can use the LdapServerName property to provide directory fail-over protection, by specifying multiple directory servers in a space-separated list.
- If you use the LdapServerName property to configure site-aware authentication:
  - If the DNS service for the domain in which the database resides is not the one where Active Directory registers its site-aware DNS SRV RRs (that is, a “foreign” service), then you must also manually configure the site-aware SRV RRs in the foreign DNS service. See [Configuring LDAP for Site-Aware Authentication](#).
  - If configuring LDAP in a Unity environment, the configuration on the Unity server and on a connected database do not have to match if users directly logging on to the database and those logging on through the Unity server are authenticated in different directories. Also see [Coordinating Mechanism Property Values for Unity](#).
  - If users directly logging on to a database and those logging on through the Unity server are authenticated by the same directory, the LdapServerName configuration for the database and the Unity server should match.
  - If you configure multiple directory services, you need to configure an LdapServerName for each service entry. See [Configuring LDAP to Use Multiple Directory Services](#).

## LdapServerPort [Deprecated]

This property identifies the LDAP service port.

---

**Note:**

Do not use `LdapServerPort` for new implementations of LDAP. This property is deprecated and is available only to support legacy configurations.

---

**Default Property Value**

TDGSS initially sets the value of this property to 389, the default LDAP service port, but the value is deprecated.

**Valid Settings**

This property is deprecated. The only valid setting is 0.

For legacy configurations that use LDAP authentication and specify an `LdapServerPort`:

- Reset the value of `LdapServerPort` to zero.

---

**Note:**

To minimize the number of database restarts (required to enable a configuration change), you can wait and include the reset of `LdapServerPort` with your next TDGSS configuration change.

---

- Incorporate the required port designation into the URI form of `LdapServerName`. See [LdapServerName](#).

**Editing Guidelines**

- This property is deprecated. If you use LDAP authentication or authorization, specify the port designation as part of `LdapServerName`.
- Edit this property on Teradata Vantage nodes.

**LdapServerRealm [Deprecated]**

This property is only used with DIGEST-MD5 binding.

---

**Note:**

The DIGEST-MD5 authentication protocol used by LDAP is deprecated. Teradata strongly recommends you use simple binding with TLS protection, and stop using DIGEST-MD5.

---

If the directory offers more than one realm and the system uses DIGEST-MD5 binding, you must use this property to identify the default SASL realm that the directory server should use for authentication. The system ignores this property if it uses simple binding.

If a user specifies a realm in the logon string, in the form `.logdata realm=realm`, the logon specification overrides the setting for this property.

---

**Note:**

Directory users that log on to Teradata Vantage must inhabit the specified realm.

---

**Valid Settings**

- "" (default), that is, the property does not specify a realm
- A valid SASL realm that the authenticating directory server offers

**Editing Guidelines**

- If the directory server offers multiple SASL realms, you must set the value of this property to identify the default realm name.
- Edit this property on database nodes and on the Unity server, if used. Also see [Coordinating Mechanism Property Values for Unity](#).
- If the directory offers only one realm, you do not need to set a value.

**LdapSystemFQDN**

The LdapSystemFQDN property identifies the FQDN of the tdatSystem object that is the parent of the structure used for LDAP user authorization. This information helps LDAP locate objects and mappings applicable to the system without making a deep search of the directory.

---

**Note:**

This value is useful only when the AuthorizationSupported property is set to yes.

---

**Valid Settings**

- "" (default), that is, the property does not specify an object to help the search
- The FQDN of a tdatSystem directory object.

For information about the tdatSystem object, see [Creating the Top-Level Objects in the DIT](#).

**Editing Guidelines**

- The LdapSystemFQDN property appears by default only in the LDAP mechanism. You must add the LdapSystemFQDN to the TDGSS configuration file and specify a value for any mechanism where AuthorizationSupported=yes, including KRB5 and SPNEGO. See [Changing the TDGSS Configuration](#).
- You must set a value for the LdapSystemFQDN property in any authentication mechanism that has the AuthorizationSupported property set to yes.
- If the directory serves one Teradata Vantage system, the LdapSystemFQDN properties configured on that system name the tdatSystem object that represents the system.

- If the directory serves multiple Vantage systems, the LdapSystemFQDN on each system points to the tdatSystem object that contains the authorization structure for the system. Several database systems can point to the same tdatSystem object if they have identical authorization requirements.
- If users log on only through Unity, the Unity server must point to the tdatSystem object that contains the LDAP authorization structure.
- If users can log on through Unity and directly to database systems, each logon must point to the tdatSystem object that contains its authorization structure. Also see [Coordinating Mechanism Property Values for Unity](#).

## LdapUserBaseFQDN

This property helps narrow the directory search to the children of the object that contains user objects, when LDAP authenticates a user.

This property applies to all mechanisms that can specify directory authorization.

### Default Property Value

TDGSS initially sets the value of this property to "" for all mechanisms, that is, it does not define an FQDN

### Valid Settings

- "" (default), that is, the property does not specify an object to narrow the search
- The FQDN of the directory object that contains directory user objects

### Editing Guidelines

- This property appears only in the library configuration file. You must manually add it to the TDGSS configuration file before you can configure it. See [About Editing Configuration Files](#).
- You must set a value for this property if the directory is Active Directory, ADAM, AD LDS or any uncertified LDAPv3-compliant directory. See [About Certified Directories](#).
- The value of the LdapUserBaseFQDN property often corresponds to the value of the identity search or identity map Base attribute, but the Base attribute is not a substitute for the LdapUserBaseFQDN. If you configure an identity search and the search fails, LDAP uses the value of the LdapUserBaseFQDN property. See [Optimizing Directory Searches](#).
- Edit this property on database nodes and on the Unity server, if used. Also see [Coordinating Mechanism Property Values for Unity](#).

## UseLdapConfig

The UseLdapConfig property determines whether TDGSS uses the property values in the base mechanism configuration or the values in the <LdapConfig> section. You can use the LdapConfig section to configure:

- User authentication by multiple directory services. See [Configuring LDAP to Use Multiple Directory Services](#).



- Security policies. See [Network Security Policy](#).

### Default Property Value

The default value is no.

### Valid Settings

Setting	Description
no (default)	TDGSS uses the configuration contained in the base mechanism.
yes	TDGSS uses the configuration contained in the LdapConfig section

### Editing Guidelines

- The UseLdapConfig property appears by default in the TdgssLibraryConfigFile.xml. If you want to use the property you must copy the property into the TDGSS configuration file (TdgssUserConfigFile.xml for database, or TdgssUnityConfig.xml for Unity) for each mechanism through which users may need to be authenticated/authorized by multiple directory services, and reset the value to yes.
- You can configure this property on database nodes and on Unity. Also see [Coordinating Mechanism Property Values for Unity](#).
- If you import the UseLdapConfig property into the TdgssUserConfigFile.xml and reset the value to yes, the TdgssUserConfigFile.xml must contain a valid <LdapConfig> section or logons fail. For configuration information, see [Creating the <LdapConfig> Section in the TdgssUserConfigFile.xml](#).

## JWT Support Properties

### JWTClientTlsCACertDir

The JWTClientTlsCACertDir property specifies the location of the CA certificates. Specifies the full path to the site/ssl/cacerts directory. For example: /opt/teradata/tdat/tdgss/site/ssl/cacerts/.

### Default Property Value

There is no default, but it is typically here: /opt/teradata/tdat/tdgss/site/ssl/cacerts/.

### Related Information

For more information about configuring JWT, see [Configuring Single Sign-On](#).

### JWTClientUseTls

The JWTClientUseTls property enforces TLS 1.2 or higher on REST API calls. This makes sure that the REST API always uses https and that peer and host verification is done.

### Default Property Value

The default setting is "Yes". The value "No" should not be used in production.

### Related Information

For more information about configuring JWT, see [Configuring Single Sign-On](#).

## JWTDecryptionKeyFile

[Optional] The JWTDecryptionKeyFile property specifies the name of the file that contains the decryption key and is used to decrypt the JWT. This is part of JWT validation.

### Default Property Value

The default setting is "", that is, no file is specified.

### Valid Settings

Setting	Description
""	No decryption file is specified.
A file name	Specifies the name of the file that contains the decryption key.

### Related Information

For more information about configuring JWT, see [Configuring Single Sign-On](#).

## JWTDynamicKey

The JWTDynamicKey property enables dynamic key rotation.

### Default Property Value

The default setting is "No". To enable JWTDynamicKey, set the property value to "Yes".

### Related Information

For more information about configuring JWT, see [Configuring Single Sign-On](#).

## JWTKeyCacheRefreshTime

The JWTKeyCacheRefreshTime property specifies the interval (in minutes) at which the key cache is purged, so the new key cache is refreshed.

### Default Property Value

The default setting is 1440 minutes (24 hours).

**Related Information**

For more information about configuring JWT, see [Configuring Single Sign-On](#).

**JWTKeyDirectory**

The JWTKeyDirectory property specifies the name of the directory that contains the JWK file.

**Default Property Value**

The default setting is "", that is, no directory is specified.

**Valid Settings**

Setting	Description
""	No directory is specified.
A directory name	Specifies the name of the directory.

**Related Information**

For more information about configuring JWT, see [Configuring Single Sign-On](#).

**JWTRestAPIMaxTimeAllowed**

The JWTRestAPIMaxTimeAllowed property specifies the maximum (in seconds) REST API call timeout.

**Default Property Value**

The default setting is 20 seconds.

**Related Information**

For more information about configuring JWT, see [Configuring Single Sign-On](#).

**JWTRestAPITimeLimit**

The JWTRestAPITimeLimit property specifies time (in seconds) between REST API calls. Too many REST API calls causes denial of service.

**Default Property Value**

The default setting is 10 seconds.

**Related Information**

For more information about configuring JWT, see [Configuring Single Sign-On](#).

## JWTSkewTime

The JWTSkewTime property specifies the maximum skew time (in seconds) allowed during JWT validation.

### Default Property Value

The default setting is 300 seconds (5 minutes).

## JWTTokenExchange

The JWTTokenExchange enables token exchange when set to "Yes".

### Default Property Value

The default setting is "No".

### Related Information

For more information about configuring JWT, see [Configuring Single Sign-On](#).

## JWTVerificationKeyFile

The JWTVerificationKeyFile property specifies the name of the file that contains the verification key and is used to verify the signature in JWT. This part of JWT validation.

### Default Property Value

The default setting is "", that is, no file is specified.

### Valid Settings

Setting	Description
""	No verification file is specified.
A file name	Specifies the name of the file that contains the verification key.

### Related Information

For more information about configuring JWT, see [Configuring Single Sign-On](#).

## LDAP Binding Properties [Deprecated]

Use LDAP binding properties to enable simple binds, when the default SASL/DIGEST-MD5 binding does not meet site security policy.

---

**Note:**

The DIGEST-MD5 authentication protocol used by LDAP is deprecated. Teradata strongly recommends you use simple binding with TLS protection, and stop using DIGEST-MD5.

---

## LdapClientMechanism

The LdapClientMechanism property specifies the bind type TDGSS must use to bind the user, during LDAP user authentication. See [LDAP Binding Options](#).

### Valid Settings

- sasl/digest-md5 (default)

---

**Note:**

Although the sasl/digest-md5 setting is the default (for legacy compatibility), it is not recommended for use.

---

- simple (recommended)

### Editing Guidelines

- The LdapClientMechanism property appears by default in the library configuration file for the LDAP mechanism. Other mechanisms do not support this property.
- To reset the value from the default, you must manually add this property to the TDGSS configuration file for the LDAP mechanism. See [About Editing Configuration Files](#).
- Edit this property on database nodes and on the Unity server, if used. Also see [Coordinating Mechanism Property Values for Unity](#).
- Change the setting to simple to support simple binds.
- Regardless of the LdapClientMechanism setting, Teradata strongly recommends that you also setup TLS protection to guard against man-in-the-middle and other attacks. See [Using TLS with a Directory Server](#).

## LdapServiceBindRequired

Specifies whether LDAP requires the service bind, that is whether Teradata Vantage must authenticate itself to the directory. See [Using Service Binds](#).

---

**Note:**

Service binds occur automatically, and ignore the setting of this property, when the authentication mechanisms specify:

---

- An Identity search. See [Using Identity Searches](#).
- Directory Authorization, when the mechanism is not LDAP. See [About Directory User Characteristics](#).

## Valid Settings

Setting	Description
yes	A service bind is required
no (default)	A service bind is not required

## Editing Guidelines

- To set a value, you must manually add this property to the TDGSS configuration file for the LDAP mechanism. See [About Editing Configuration Files](#).
- Edit this property on database nodes and on the Unity server, if used. Also see [Coordinating Mechanism Property Values for Unity](#).
- If your system uses service binding, whether or not it requires this property to be set to yes, you should configure the [LdapServiceFQDN](#), [LdapServicePassword](#), and possibly [LdapServicePasswordProtected](#) properties. If you do not configure these companion properties, the directory automatically uses an anonymous bind.

## LdapServiceFQDN

The value of the LdapServiceFQDN property is a distinguished name (DN) that identifies a bindable object in the directory, which represents the service or application that requires binding.

The LdapServiceFQDN property is usable for all mechanisms that support service binds.

You must configure LdapServiceFQDN and LdapServicePassword for:

- All service binds, or the directory automatically uses an anonymous service bind. For an explanation of service binds and a list of features that require them, see [Using Service Binds](#).
- Service entries in the <LdapConfig> section of the TdgssUserConfigFile.xml, for example, when configuring multiple directory services for accessing a Teradata Vantage system.

### Note:

If one of these conditions exists and you do not configure the LdapServiceFQDN property, the system defaults to binding anonymously, which may produce unwanted results, including bind failure.

## Valid Settings

- "" (default), which specifies an anonymous bind
- The FQDN of a bindable directory object that represents the service or application identity.

## Editing Guidelines

- KRB5 and SPNEGO mechanisms must be edited if AuthorizationSupported is set to yes; otherwise do not edit.

- LDAP may be edited (required for use of service binds).
- To set a value, you must manually add this property to the TDGSS configuration file for needed mechanisms. See [About Editing Configuration Files](#).
- Edit this property on database nodes and on the Unity server, if used.
- You must uniquely identify each service account.
- If you configure this property, you must also specify a value for the LdapServicePassword property.

## LdapServicePassword

If the service defined by the LdapServiceFQDN property requires a password, you must either configure the LdapServicePassword property or provide an LDAP service password file. See [LdapServicePasswordFile](#) and [LdapServiceFQDN](#).

The LdapServicePassword property is usable only for mechanisms that support service binds.

---

### Note:

You can encrypt the LdapServicePassword using the LdapServicePasswordProtected property, as shown in [LdapServicePasswordProtected](#).

---

## Valid Settings

Setting	Description
no (default)	The system does not use service binds, or if it does use service binds, a password is not required.
A password	The password for the service defined by the LdapServiceFQDN property.

The password for the service FQDN, if a password is required.

## Editing Guidelines

- To set a value, you must manually add this property to the TDGSS configuration file on needed mechanisms. See [About Editing Configuration Files](#).
- Edit this property on the database and on Unity, if used. Also see [Coordinating Mechanism Property Values for Unity](#).
- If you do not specify a password, the system assumes that no password is required.
- If you configure LdapServicePassword, you must also configure LdapServiceFQDN.
- If you use this property to specify a service password, note that the password appears in the TdgssUserConfigFile.xml in plain text. To protect the LdapServicePassword, set the [LdapServicePasswordProtected](#) property to yes.

## LdapServicePasswordFile

The value of LdapServicePasswordFile names a file containing a list of encrypted passwords. Using a password file allows you to change the LDAP service password without requiring a restart of the database SQL Engine system. By storing multiple passwords, the file enables LDAP logons during the period of transition between old and new passwords.

The LdapServicePasswordFile property can be added to either of these locations:

- Service element in the LdapConfig section of the TDGSS user configuration file
- MechanismProperties element under Mechanism Name="*mechanism*" for the LDAP and KRB5 mechanisms on the server side (SQL Engine or Unity) of TDGSS

### Valid Settings

Setting	Description
""	No LDAP service password file is specified. This feature is disabled.
A file name	The fully qualified path and file name of the LDAP service password file.

### Editing Guidelines

- To set a value, you must manually add this property to the TDGSS configuration file on needed mechanisms. See [About Editing Configuration Files](#).
- Edit this property on the database and on Unity, if used. Also see [Coordinating Mechanism Property Values for Unity](#).
- If you use the LdapServicePasswordFile property, the LdapServicePassword and LdapServicePasswordProtected properties are ignored, and passwords are read exclusively from the password file.

### LDAP Service Password File

Use the LDAP service password file to enable changing the LDAP service password without the need for a database system restart. The file contains a list of encrypted passwords, one per line. When you want to change the password, add the new password to the top of the list in the file, leaving, at a minimum, the previous password listed on the second line.

After you have updated the password file on all nodes, you can change the actual service password in the directory server.

If an attempted LDAP logon fails, Vantage reads the passwords in the password file, trying each successively. This technique of preserving the old password in the file, and trying all passwords allows for slight delays in updating the LDAP password file for every system node.



## Encrypting the Passwords

Passwords listed in the LDAP service password file must be encrypted using the `tdspasswd` command-line utility:

1. At the Vantage system console command prompt, enter:

```
$ tdspasswd -m mechanism
```

where *mechanism* is the authentication mechanism, `ldap` or `krb5`.

2. The system prompts you to enter the new password.

```
Enter New password:
Confirm New password:
```

---

### Note:

The system does not display the password when you enter it.

---

3. After the system confirms the new password, it generates and displays an encrypted version of the password, for example:

```
$ tdspasswd -m ldap
Enter New password:
Confirm New password:
AV8Jeq2cvjmAjiHgcSrAUoE=
$
```

4. Copy the encrypted new password to the first line of the LDAP service password file.

## Usage Notes for the LDAP Service Password File

- If your site locks accounts after a number of failed logon attempts, the LDAP service account could become unusable due to what the directory would see as consecutive logon failures from nodes that try the old password, before they have received and processed the updated password file. If you cannot raise the limit on failed logon attempts for the service account (`LdapServiceFQDN`), one or more manual unlocks of the account may be required during the password change process.

If possible, disable any account lockout requirements on the service account before you reset the password. Then reset the password and direct LDAP logons to every node. This causes the new password to be picked up and used by every node. After that, re-establish the lockout requirements on the service account.

- Use a tool like `pcl` (a PDE tool) to propagate changes to the LDAP service password file to all nodes.

## LdapServicePasswordProtected

The value of the LdapServicePasswordProtected property indicates whether the password defined in LdapServicePassword is stored in encrypted form. You can use the -s option of the tdspasswd tool to create an encrypted version of the LdapServicePassword. See [Identity Search Implementation Process](#).

The LdapServicePassword property is usable for all mechanisms that support service binds.

### Note:

If you use this property also configure the LdapServiceFQDN and LdapServicePassword properties.

## Valid Settings

Setting	Description
yes	The TdgssUserConfigFile.xml stores the LdapServicePassword in encrypted form
no (default)	The TdgssUserConfigFile.xml stores the LdapServicePassword in plain text.

## Editing Guidelines

- To set a value, you must manually add this property to the TDGSS configuration file on needed mechanisms. See [About Editing Configuration Files](#).
- Edit this property on database nodes and the Unity server, if used. Also see [Coordinating Mechanism Property Values for Unity](#).
- This property does not encrypt the password. It is only an indicator of encryption.
- If you want to encrypt the password, use the procedure found in [Editing TdgssUserConfigFile.xml for Service Binds](#) to generate an encrypted password for the LdapServicePassword property, and insert it into the TdgssUserConfigFile.xml, then set the LdapServicePasswordProtected property to yes to indicate that the service password is encrypted.

## LDAP Policy Properties

LDAP policy properties only apply when configuring security policies. The policy properties must be used along with other LDAP properties that are not used exclusively for policy configuration.

For information, see [Configuring Security Policies in the TdgssUserConfigFile.xml](#).

## LdapNetworkBaseFQDN

The fully qualified distinguished name of the directory container object that contains ipNetwork entries.

An ipNetwork entry specifies an IP address or a range of IP addresses. When you configure an ipNetwork object as a member of an:

- Internal Network Group object, the included IP addresses are subject to any policy in which the Network Group object is a member.
- External Network Group object, the included IP addresses are exempt from any policy in which the Network Group object is a member.

### Valid Settings

- "" (default), that is, no FQDN is specified
- The FQDN of a bindable directory object that contains ipNetwork entries.

### Editing Guidelines

- The LdapNetworkBaseFQDN property is not configurable in a mechanism. Configure the LdapNetworkBaseFQDN property as part of the <LdapConfig> section in the TDGSS configuration file, immediately following the <Mechanisms> section. See [About Editing Configuration Files](#).
- Edit this property on the database and on Unity, if used. Also see [Coordinating Mechanism Property Values for Unity](#).
- You must uniquely identify the object that contains ipNetwork entries if you assign security policies by IP address.
- If you configure this property, you do not need to specify a value for the LdapBaseFQDN property.

For details on configuring LDAP properties to configure security policy, see [Configuring Security Policies in the TdgssUserConfigFile.xml](#).

## LdapPolicyFQDN

The fully qualified distinguished name of a tdatPolicy directory object.

A tdatPolicy object is the parent of a directory-based security policy structure.

### Valid Settings

- "" (default), that is, no FQDN is specified
- The FQDN of a bindable directory object that is the parent of a security policy structure.

### Editing Guidelines

- To set a value, you must manually add this property to the TDGSS configuration file for needed mechanisms. See [About Editing Configuration Files](#).
- Edit this property on the database and on Unity, if used. Also see [Coordinating Mechanism Property Values for Unity](#).
- Specify a property value that is the FQDN of a bindable directory object (tdatPolicy object) that is the parent of a security policy structure.

For details on configuring LDAP properties to configure security policy, see [Configuring Security Policies in the TdgssUserConfigFile.xml](#).

## MechanismIgnoreQOP

Determines whether a session logged on with a mechanism uses or ignores an otherwise-applicable QOP policy.

The MechanismIgnoreQOP property is necessary to automatically exempt mechanisms that do not use QOP settings from being subject to QOP policy, and are set to yes by default:

- KRB5
- SPNEGO

---

### Note:

The KRB5 and SPNEGO mechanisms ignore QOP because Kerberos encrypts sessions independent of Teradata security policy.

---

### Valid Settings

- yes, ignores applicable QOP settings
- no, uses applicable QOP settings.

### Editing Guidelines

This property is not editable.

## LDAP Protection Properties

Teradata recommends TLS protection for systems that use simple binding, including service binds. LDAP protection configured in the LDAP mechanism applies to all external authentication mechanisms.

---

### Note:

Review [Using TLS with a Directory Server](#) before configuring TLS.

---

Protection Feature	Mechanism Property
Enable TLS protection	<ul style="list-style-type: none"> <li>• For TLS Protection, see <a href="#">LdapClientUseTLS</a></li> </ul>
Verify the certificate chain	<ul style="list-style-type: none"> <li>• All certs in one file, <a href="#">LdapClientTlsCACert</a></li> <li>• All certs in one directory <a href="#">LdapClientTlsCACertDir</a></li> <li>• <a href="#">LdapClientTlsReqCert</a></li> <li>• <a href="#">LdapClientTlsCRLCheck</a></li> </ul>
Configure protection for mutual authentication	<ul style="list-style-type: none"> <li>• <a href="#">LdapClientTlsCert</a></li> <li>• <a href="#">LdapClientTlsKey</a></li> </ul>
TLS utility properties	<ul style="list-style-type: none"> <li>• <a href="#">LdapClientTlsCipherSuite</a></li> </ul>

Protection Feature	Mechanism Property
	<ul style="list-style-type: none"> <li>• <a href="#">LdapClientTlsRandFile</a></li> </ul>
Allow use of unsafe versions of directory software that do not conform to IETF RFC 5746	<a href="#">LdapAllowUnsafeServerConnect</a>

## Avoiding Conflicts with OpenLDAP Tunables

Some TDGSS protection properties perform similar functions to OpenLdap tunables. Always use the TDGSS values instead of OpenLdap tunable settings, to avoid unexpected results.

TDGSS LDAP Protection Property	OpenLdap Tunable
LdapClientTlsCACert	TLS_CACERT
LdapClientTlsCACertDir	TLS_CACERTDIR
LdapClientTlsCert	TLS_CERT
LdapClientTlsCipherSuite	TLS_CIPHER_SUITE
LdapClientTlsCRLCheck	TLS_CRLCHECK
LdapClientTlsKey	TLS_KEY
LdapClientTlsRandFile	TLS_RANDFILE
LdapClientTlsReqCert	TLS_REQCERT

## LdapAllowUnsafeServerConnect

The setting of the LdapAllowUnsafeServerConnect property is based on whether the directory supports IETF RFC 5746 “Transport Layer Security (TLS) Renegotiation Indication Extension.” Older directory software versions may not support this standard. Teradata Vantage software can operate with either a supporting or non-supporting directory, depending on the property setting.

This property is applicable only for systems that use simple binding and TLS protection.

### Note:

Teradata recommends leaving this setting at the default value to ensure connectivity to directory servers that do not support IETF RFC 5746.

### Default Property Value

The default setting is yes, that is, Vantage operates with directories that do not support IETF RFC 5746.

## Valid Settings

Setting	Description
"yes" (default)	Allows connection to the database using directories that do not support IETF RFC 5746.
"no"	Requires that the directory support IETF RFC 5746 to connect to the database.

## Editing Guidelines

- Edit this property on the database and on Unity, if used. Also see [Coordinating Mechanism Property Values for Unity](#).
- To set a value, you must manually add this property to the TDGSS configuration file for the needed mechanisms. See [About Editing Configuration Files](#).
- Although you can configure this property only in the LDAP mechanism, the setting applies to all external authentication mechanisms.
- Before changing the default property value, make sure your directory software supports IETF RFC 5746 “Transport Layer Security (TLS) Renegotiation Indication Extension.” Consult your directory software vendor for details.

## LdapClientSASLSecProps

The LdapClientSaslSecProps property specifies the security level for the token exchange.

When a directory user logs on to a Teradata Vantage system, and the SASL token exchange between the directory server and Vantage uses DIGEST-MD5 binding, an attacker could challenge the exchange and redirect it to send the token in clear text. You can set the LdapClientSaslSecProps property to provide extra protection for a DIGEST-MD5 token exchange.

### Note:

The DIGEST-MD5 authentication protocol used by LDAP is deprecated. Teradata strongly recommends you use simple binding with TLS protection, and stop using DIGEST-MD5.

## Default Property Value

The default value of the LdapClientSaslSecProps property is minssf=0, that is, the security level is compatible with all supported directory types and configurations, but it does not provide any extra protection.

## Editing Guidelines

- To set a value, you must manually add this property to the TDGSS configuration file for the LDAP mechanism. See [About Editing Configuration Files](#).
- Edit this property on the database and on Unity, if used. Also see [Coordinating Mechanism Property Values for Unity](#).

- If you set the property value to minssf=0, the setting avoids possible conflicts with directory types and configurations that cannot use a higher security level.
- You can set the property value to minssf=1, to cause the directory server to offer an authint or auth-conf QOP.
  - Auth-int adds a message digest (signing) to messages between the database and directory.
  - Auth-conf adds encryption and message digests (signing and sealing) to messages between the database and directory.

Integrity checking prevents man-in-the-middle attack, which could reset the QOP level and cause the password to be transmitted in clear text. A setting of minssf=1 is sufficient for most implementations.

- You can set the property value to encrypt the token exchange. A setting of:
  - minssf=56 uses DES or other low-level ciphers
  - minssf=112 uses triple DES and other strong ciphers
  - minssf=128 uses of the strongest ciphers, for example, RC4.

---

**Note:**

If you specify a minssf value above 1, the directory must support the corresponding encryption level, and your setting cannot exceed the directory setting for the maxssf property.

---

## LdapClientTlsCACert

The LdapClientTlsCACert property specifies the name of the file that contains all the Certificate Authorities (CAs), including the certificate for the CA that signed the directory server certificate that TDGSS and OpenLdap tools trust. If the signing CA is not a top-level (root) CA, certificates for the entire sequence of CAs from the signing CA to the top-level CA must be present. The certificate order is not significant.

---

**Note:**

You can use this property for certificate chain verification when all CAs are in one file, but LdapClientTlsCACertDir is preferred. See [LdapClientTlsCACertDir](#).

---

## Valid Settings

Setting	Description
"" (default)	No file is specified
A file name	The file must contain concatenated CA certificates in PEM format.

## Editing Guidelines

- To set a value, you must manually add this property to the TDGSS configuration file for the needed mechanisms. See [About Editing Configuration Files](#).

- Configure this property or LdapClientTlsCACertDir (preferred) when using TLS. See [Verifying the Directory Server Certificate Chain](#).
- If you decide to use TLS protection, edit this property for all mechanisms that have the AuthorizationSupported property set to yes.
- Set the value on each node and the Unity server, if used. Also see [Coordinating Mechanism Property Values for Unity](#).

**Note:**

The Linux user under which Teradata Vantage runs must own and have read access to this file. For sites that configured this property before Release 14.0, the permission is granted automatically by a script upon upgrade to Release 14.0. For sites that configure this property on Release 14.0 or later, you must grant the permission manually.

## LdapClientTlsCACertDir

The LdapClientTlsCACertDir property specifies the path of a directory that contains individual CA certificates in separate files. You can use the [LdapClientTlsCACert](#) property to support TLS certificate chain verification, but LdapClientTlsCACertDir is preferred.

To assign a value to the LdapClientTlsCACertDir property, you must generate symbolic links, using the TDGSS certlink utility, which point to the actual certificate files. See [Creating the CA Certificate Symlinks](#) for instructions on using the certlink utility.

### Valid Settings

Setting	Description
"" (default)	No cert directory is specified
A valid directory path	The path to a directory that contains individual CA certificates, in separate files, for all of the Certificate Authorities the client recognizes. The file system you use for the path must support symbolic links.

### Editing Guidelines

- The LdapClientTlsCACertDir property appears only in the library configuration file. To set a value, you must manually add it to the TDGSS configuration file for the needed mechanisms. See [About Editing Configuration Files](#).
- If you decide to use TLS protection, edit this property for all mechanisms that have the AuthorizationSupported property set to yes.
- Edit this property on the database and the Unity server. Also see [Coordinating Mechanism Property Values for Unity](#).
- Specify the path of a directory that contains individual CA certificates in separate files for all of the Certificate Authorities the client recognizes.



---

**Note:**

The Linux user under which Teradata Vantage runs must own and have read access to this file. For sites that configured this property before Release 14.0, the permission is granted automatically by a script upon upgrade to Release 14.0. For sites that configure this property on Release 14.0 or later, you must grant the permission manually.

---

## LdapClientTlsCert

The LdapClientTlsCert property specifies the file that contains the TDGSS or OpenLdap client certificate that the directory server uses to authenticate the database.

### Default Property Value

The default value of the LdapClientTlsCert property is "", meaning that no cert file is specified.

### Valid Settings

A valid file name.

### Editing Guidelines

- To set a value, you must manually add this property to the TDGSS configuration file for the needed mechanisms. See [About Editing Configuration Files](#).
- You must edit this property if you configure TLS mutual authentication of the directory and Teradata Vantage.
- Configure this property for all mechanisms that have the Authorization Supported property set to yes.
- Edit this property on the database nodes and the Unity server. Also see [Coordinating Mechanism Property Values for Unity](#).
- Specify the name of the cert file that contains the TDGSS or OpenLdap client certificate that the directory server uses to authenticate the database..

---

**Note:**

The Linux user under which Teradata Vantage runs must own and have read access to this file. For sites that configured this property before Release 14.0, the permission is granted automatically by a script upon upgrade to Release 14.0. For sites that configure this property on Release 14.0 or later, you must grant the permission manually.

---

## LdapClientTlsCipherSuite

LdapClientTlsCipherSuite specifies the ciphers and cipher preference order that TDGSS accepts from OpenSSL, for use in the token exchange during directory user authentication.

**Note:**

Do not use this property without a full understanding of the effects of specifying a particular cipher. If you are not sure about the effect of this property, contact Teradata Professional Services for assistance.

**Valid Settings**

Setting	Description
"" (default)	No ciphers are specified. Causes OpenLdap to use its default cipher suite.
A custom list of ciphers	Consult OpenSSL documentation for cipher list requirements.

**Editing Guidelines**

- To set a value, you must manually add this property to the TDGSS configuration file for the needed mechanisms. See [About Editing Configuration Files](#).
- Before you configure this property, use the command `openssl ciphers -v ALL` to obtain a list of ciphers available from OpenSSL.
- If you configure this property, use a colon-separated list of ciphers, in preference order. The list must be in accordance with OpenSSL documentation.
- You can specify HIGH, MEDIUM, LOW, EXPORT, or EXPORT40 (instead of cipher names) to indicate a strength range for acceptable ciphers.
- You can specify TLSv1, SSLv3, or SSLv2 to indicate a cipher suite.
- If you decide to configure this property, edit the value for all mechanisms that have the AuthorizationSupported property set to yes.
- Edit this property on the database and the Unity server, if used. Also see [Coordinating Mechanism Property Values for Unity](#).

**LdapClientTlsCRLCheck**

The LdapClientTlsCRLCheck property specifies how the authentication mechanism should use the Certificate Revocation List (CRL) of the CA to verify that the server certificates are not revoked. LDAP performs the checking authorized by this property on every bind.

**Default Property Value**

The default value of the LdapClientTlsCRLCheck property is none, meaning that the system performs no CRL checks.

**Valid Settings**

Setting	Description
none	Specified that LDAP performs no CRL checks

Setting	Description
peer	Specifies that LDAP checks the CRL of the server certificate
all	Specifies that LDAP checks the CRLs of the entire certificate chain

## Editing Guidelines

- You must edit this property if you want to check CRLs. However, CRL checking is optional, based on site security policy.
- To set a value, you must manually add this property to the TDGSS configuration file for the LDAP mechanism. See [About Editing Configuration Files](#).
- Edit the value of the LdapClientTlsCRLCheck property only on Teradata Vantage nodes.
- If you decide to use TLS protection, edit this property for all mechanisms that have the AuthorizationSupported property set to yes.
- Edit this property on database nodes and on the Unity. For more information, see [Coordinating Mechanism Property Values for Unity](#).

## LdapClientTlsKey

This property identifies the file that contains the private key that matches the certificate stored in the file named in the LdapClientTlsCert property. The LdapClientTlsKey value is required for mutual authentication of the directory and the database. See [Using Mutual Authentication Between the Directory Server and Teradata Vantage](#).

## Default Property Value

The default value of the LdapClientTlsKey property is "", meaning that no key file is specified.

## Valid Settings

Setting	Description
"" (default)	No key file is specified
A valid filename	The file must contain the private key that matches the certificate stored in the file named in the LdapClientTlsCert property.

## Editing Guidelines

- To set a value, you must manually add this property to the TDGSS configuration file for the LDAP mechanism. See [About Editing Configuration Files](#).
- Edit this property on nodes and on Unity. For more information, see [Coordinating Mechanism Property Values for Unity](#).
- You can edit LdapClientTlsKey in the TDGSS user configuration file to specify the file that contains the key for the certificate specified in the LdapClientTlsCert property.

**Note:**

The Linux user under which Teradata Vantage runs must own and have read access to this file. Before Release 14.0, this permission was granted automatically. For new configurations on Release 14.0 or later, you must grant the permission manually.

- If you use this property you must also configure the LdapClientTlsCert property.
- Edits to this property for the LDAP mechanism apply to all supporting mechanisms.

## LdapClientTlsRandFile

The LdapClientTlsRandFile property specifies a device, FIFO, or pipe that provides random bits when the default random number generator, for example /dev/[u]random on Linux, is not available, or if another random number generator is preferred.

**Note:**

Use with simple binding.

### Default Property Value

The default setting is "", that is, no device, FIFO, or pipe is specified.

### Valid Settings

Setting	Description
""	No random number generator is specified. The system uses the default random number generator for the operating system.
A file name	Specifies a device, FIFO, or pipe that provides random bits.

### Editing Guidelines

- To set a value, you must manually add this property to the TDGSS configuration file for the LDAP mechanism. See [About Editing Configuration Files](#).
- Edit this property on the database and on Unity, if used. Also see [Coordinating Mechanism Property Values for Unity](#).
- Specify a value for the LdapClientTlsRandFile only when the default random number generator is not available, or if you want to use another random number generator.
- If a default is not available, install a copy of EGD or PRNGD on every node and set the value of LdapClientTlsRandFile to the name of the EGD or PRNGD socket.

**Note:**

The installed copy of EGD or PRNGD must have the same name on all Teradata Vantage nodes. See the EGD or PRNGD Readme file for details.

- Although you can configure this property only in the LDAP mechanism, the effects apply to all external authentication mechanisms.

## LdapClientTlsReqCert

The LdapClientTlsReqCert property specifies what checks to perform on directory server certificates (if any), in a TLS-protected session. This property is required when Teradata Vantage authenticates the directory server.

### Valid Settings

Setting	Description
never (default)	The database does not require the directory server to provide a certificate, even if CA Certs or CRLs are configured.
allow	Vantage asks the directory server for a certificate. If it does not provide a certificate, or if it provides an invalid certificate, the connection proceeds normally.
try	Vantage asks the directory server for a certificate. If the directory server: <ul style="list-style-type: none"><li>Does not provide a certificate, the connection proceeds normally</li><li>Provides an invalid certificate, the connection terminates.</li></ul>
demand	Vantage asks the directory server for a certificate. If it does not provide a certificate, or if it provides an invalid certificate, the connection terminates.

### Editing Guidelines

- To set a value, you must manually add this property to the TDGSS configuration file for the needed mechanisms. See [About Editing Configuration Files](#).
- Edit this property on the database and on Unity, if used. Also see [Coordinating Mechanism Property Values for Unity](#).
- This property is required for optional certificate chain verification. For information, see [Verifying the Directory Server Certificate Chain](#).
- Although you can configure this property only in the LDAP mechanism, the effects apply to all external authentication mechanisms.

## LdapClientUseTLS

The LdapClientUseTls property specifies whether TLS protection is enabled. Teradata strongly recommends TLS protection when you use simple binds, including service binds.

**Note:**

This property must be set to yes to enable use of advanced TLS capabilities, such as certificate chain verification or mutual authentication.

**Valid Settings**

Setting	Description
yes	TLS protection is enabled
no (default)	TLS protection is not enabled

**Editing Guidelines**

- To set a value, you must manually add this property to the TDGSS configuration file for the needed mechanism(s). See [About Editing Configuration Files](#).
- Set the LdapClientUseTls property to yes to protect passwords on systems that use simple binds, including service binds. For information on binding, see [LDAP Binding Options](#).
- If you decide to use TLS protection, edit this property for all mechanisms that have the AuthorizationSupported property set to yes.
- Edit this property on the database and on Unity, if used. Also see [Coordinating Mechanism Property Values for Unity](#).

For detailed procedures on configuring TLS options, see [Using TLS with a Directory Server](#).

## Mechanism Status Properties

### DefaultMechanism

TDGSS uses the default mechanism when a user logon does not specify a mechanism. See [Determining the Authentication Mechanism for a Logon](#) for an explanation of how TDGSS determines the mechanism for a session.

**Default Property Value**

TDGSS initially sets the value of this property to yes for TD2, and to no for all other mechanisms, in the Teradata Vantage configuration files. TDGSS does not preset a default mechanism in the client configuration files.

**Valid Settings**

Setting	Description
yes	The mechanism is the default.
no	The mechanism is not the default.

## Editing Guidelines

- You can set the DefaultMechanism property on Teradata Vantage nodes, Vantage clients, and on the Unity server.
- You can set one only mechanism as the default on a Vantage system, but you can designate a different default mechanism for each client.
- If you designate a new default mechanism, reset this property to no for the previous default mechanism.
- If you do not designate a default mechanism, or if a user needs to use a non-default mechanism, the user must specify a mechanism.

## DefaultNegotiatingMechanism

The DefaultNegotiatingMechanism property is used to specify a default mechanism from among the mechanisms with the NegotiationSupported property set to yes. This property is used to determine the default mechanism when both the client and the server support mechanism negotiation.

---

### Note:

Any mechanism that has the mechanism property DefaultNegotiatingMechanism set to yes must also have the mechanism property NegotiationSupported set to yes; otherwise, the tdgssconfig executable reports an error.

---

If both client and server support negotiation, but no common negotiating mechanism exists between them, then the existing DefaultMechanism property is used to select a default mechanism.

## Default Property Value

TDGSS sets the value of this property to no for all mechanisms for all platforms (Teradata Vantage, Unity servers, and clients).

## Valid Settings

Setting	Description
yes	The mechanism is the default negotiating mechanism.
no	The mechanism is not the default negotiating mechanism.

## Editing Guidelines

- You can set the DefaultNegotiatingMechanism property on Teradata Vantage nodes, Unity server, and Vantage clients.
- Any mechanism that has the mechanism property DefaultNegotiatingMechanism set to yes must also have the mechanism property NegotiationSupported set to yes, otherwise the tdgssconfig executable reports an error.

- You can designate a different default mechanism and default negotiating mechanism for each client.

## MechanismEnabled

This property determines whether a mechanism is available for use.

### Valid Settings

Setting	Description
yes (the default except PROXY)	The mechanism is available for use.
no (the default for PROXY only)	The mechanism is not available for use

### Editing Guidelines

- If you disable a mechanism on the Teradata Vantage system or on the Unity server (if used), it is unavailable for use, even if the mechanism is enabled on Vantage clients.
- When using Unity, all mechanisms enabled on the Unity server must also be enabled on all connected databases.
- You can disable a mechanism on a client to make it unavailable for only that client.

---

#### Note:

If you disable a mechanism on a client, the mechanism does not appear in client application drop-down menus.

---

- To enable externally authenticated users to log on through Unity, you must copy the PROXY mechanism from the TdgssLibraryConfigFile.xml to the TdgssUserConfigFile.xml (Teradata Vantage) and to the TdgssUnityConfig.xml (Unity) and set the MechanismEnabled property to yes. Other TDGSS configuration changes are also required to support Unity. See *Teradata® Unity™ Installation, Configuration, and Upgrade Guide for Customers*, B035-2523 for information on configuring Unity.
- For TDNEGO, this property can be set independently at the client and Vantage server. To turn off TDNEGO, set MechanismEnabled to no on all Vantage servers (see [Disabling TDNEGO](#) for other properties that must be set to disable TDNEGO). If MechanismEnabled is turned off for a mechanism offered by TDNEGO, that mechanism will not be available to be offered by TDNEGO and will also not be available independently apart from TDNEGO.

## MechanismRank

The MechanismRank property defines the order of preference in which client applications offer mechanisms for user selection.

### Default Property Values

- TDNEGO = "10"



- TD2 = "20"
- JWT = "30"
- KRB5 = "40"
- SPNEGO = "65"
- LDAP = "70"
- PROXY = "80"

### Valid Settings

The value for MechanismRank must be numeric.

Default MechanismRank values are for reference only and need not be in increments of 10. The system ranks values in order from low to high, with the lowest numbers having the highest preference.

### Editing Guidelines

- TDNEGO supports MechanismRank. However, TDNEGO only looks at the mechanism rank of the mechanisms it is trying to negotiate: TD2, ldap, KRB5, and SPNEGO. For all other mechanisms, including TDNEGO itself, MechanismRank is not used.
- TDNEGO uses the MechanismRank property to determine the priority order of the mechanisms being negotiated.
- For TDNEGO, this property can be set independently on the client and Teradata Vantage server.  
Note: The decision of which mechanism to select is attempted locally at both client and server. If all mechanism ranks are different on client and server, it is the ordering at the system where the decision is made that takes effect.

## Operational Properties

You can use operational properties to set flags that request TDGSS to employ certain options when it authenticates users or transmits data.

### AnonymousAuthentication

This property indicates whether or not to reveal the initiator identity to the acceptor.

#### Valid Settings

The preset default value, no, is the only valid setting for this property.

#### Supporting Mechanisms

The AnonymousAuthentication property is reserved for future use, and is not currently supported for any mechanisms.

#### Editing Guidelines

The system ignores any edits to this property.

## ConfidentialityDesired

This property indicates whether the mechanism supports encryption of transmitted messages.

---

**Note:**

ConfidentialityDesired does not enable or disable encryption. The user can enable encryption from the client application.

---

### Valid Settings

The preset default value, yes, is the only valid setting for this property.

### Editing Guidelines

TDGSS ignores any edits to this property. The behavior of client applications may be affected by this property.

## CredentialUsage

This property indicates how the credential is used by the system.

### Default Property Value

TDGSS initially sets the value of this property to 0 for all standard mechanisms. That is, the system can use the credential to either initiate or accept a request.

### Valid Settings

The preset default value is the only valid setting for this property.

### Editing Guidelines

This property is reserved for future use, to enable a mechanism to differentiate between initiator and acceptor functions.

## DelegateCredentials

Setting the DelegateCredentials property to yes allows a remote peer to pass user credentials to the destination Vantage system.

### Valid Settings

Setting	Description
yes	Instructs the client to delegate user credentials to a remote peer.
no (default)	Instructs the client not to delegate user credentials.

## Editing Guidelines

DelegateCredentials is not used. The system ignores the value if set to yes.

## DesiredContextTime

The DesiredContextTime property indicates the time duration, in seconds, that the security context defined by the mechanism remain in effect.

## Valid Settings

The preset default value, "", which indicates that there is no time limit on the security context, is the only valid setting for this property.

## Supporting Mechanisms

The DesiredContextTime property is not currently supported for any mechanism, but is reserved for future use.

## Editing Guidelines

Vantage standard mechanisms do not support a time limit on the security context. The system ignores any edits to the value of this property.

## DesiredCredentialTime

The DesiredCredentialTime property indicates the time duration, in seconds, that the security credential submitted by the user, and authenticated by the system, remain in effect.

## Valid Settings

The preset default value "", which indicates that the credential cannot expire, is the only valid setting for this property.

## Supporting Mechanisms

The DesiredCredentialTime property is not currently supported for any mechanism, but is reserved for future use

## Editing Guidelines

Vantage mechanisms do not currently support a time limit on the user credentials. The system ignores any edits to the value of this property.

## IntegrityDesired

This property indicates whether the client requests integrity checking of messages by the receiver.

## Valid Settings

The preset default value, yes, is the only valid setting for this property.

## Editing Guidelines

TDGSS ignores any edits to this property. The behavior of client applications may be affected by this property.

## MutualAuthentication

The MutualAuthentication property tells the client whether to request that the server authenticate itself to the client. You can use mutual authentication to avoid man-in-the-middle attacks, which could allow the attacker to divert a network transmission to an unauthenticated third party.

## Valid Settings

Setting	Description
yes (default)	The mechanism supports mutual authentication.
no	The mechanism does not support mutual authentication.

## Editing Guidelines

- Edit this property on Vantage clients.
- You must set MutualAuthentication to yes for KRB5 or SPNEGO.
- If you want user logons through Unity to behave the same as user logons direct to the database, then the property setting for KRB5 and LDAP should be the same on Unity and on all connected database systems.
- If you edit the preset value of this property on a mechanism that does not currently support it, TDGSS ignores the edit.
- Do not modify the TDNEGO setting for this property.

## OutOfSequenceDetection

This property determines whether the mechanism supports tracking and identifying requests that transmit out of sequence. Out-of-sequence detection is a built-in function for the mechanisms that support it, and the value is for reference only.

If the system detects an out-of-sequence message in a session authenticated by a supporting mechanism, the system automatically rejects the out-of-sequence message and logs an error.

## Valid Settings

The preset default value, yes, is the only valid setting for this property.

**Editing Guidelines**

The system ignores any edits to this property.

**ReplayDetection**

This property determines whether the mechanism supports tracking and identifying requests that transmit more than once. The property value is for reference only. Replay detection is a built-in function for the mechanisms that support it.

If the system detects a replayed message in a session authenticated by a supporting mechanism, the system automatically rejects the replayed message and logs an error.

**Valid Settings**

The preset default value, yes, is the only valid setting for this property.

**Editing Guidelines**

The system ignores any edits to this property.

**TeradataKeyTab**

This property defines the location for the storage of Kerberos keytab files on Vantage nodes and on Unity servers.

**Default Property Value**

TDGSS initially sets the value of this property to /etc/teradata.keytab, the default storage location, for supporting mechanisms.

**Valid Settings**

Any valid Linux file location.

**Editing Guidelines****Note:**

The TeradataKeyTab value in KRB5 applies to all Kerberos authentication, including logons that use the SPNEGO mechanism.

Coordinate the value of this property with the location of the keytab file specified when setting up Kerberos authentication. Use the default file location unless site security policy requires a different location. See [Configuring Teradata Vantage and Unity Servers for Kerberos Authentication](#).

This property appears only in the library configuration file for the KRB5 mechanism. You must manually add it to the TDGSS configuration file before you can change the property value. See [About Editing Configuration Files](#).

**Note:**

The Linux user under which Teradata Vantage runs must own and have read access to the keytab file. If you specify a custom (non-default) file location, you must grant read access permission for the file to the Vantage Linux user.

## Unity Support Properties

Unity support properties only apply to the PROXY mechanism. Configuration of Unity support properties is required when users log on to Teradata Vantage through Unity.

You can modify Unity support properties without performing a TPA reset, unless `run_tdgssconfig` indicates you need to do a TPA reset.

For information about all TDGSS configuration tasks required to support Unity, see [Coordinating Mechanism Property Values for Unity](#).

## CACertDir

Specifies the directory that contains hash linked copies of certificates used in validating connections between Unity and connected Teradata Vantage systems. All certificates contained in the directory have hash links created by the certlink tool.

The directory on Unity contains certificates for connected database systems, and the database directory contains the certificates for Unity server. Also see [Coordinating Mechanism Property Values for Unity](#).

### Default Property Value

The default setting is "", that is, no directory is specified.

### Valid Settings

Setting	Description
""	No directory is specified.
A directory name	The name of the directory that contains hash links to the certificates.

### Editing Guidelines

- To set a value, copy the PROXY mechanism from the `TdgssLibraryConfigFile.xml` and add it to the TDGSS configuration file. See [About Editing Configuration Files](#).
- Edit this property on the Unity server and on connected Vantage nodes. For an example, see [Coordinating Mechanism Property Values for Unity](#).
- On a Unity server, the specified directory contains hash links to the certificates for all connected Vantage systems.

- On a Vantage system, the directory contains hash links to the certificates for all Unity servers to which the database connects.
- You can instead use the CACertFile property to identify the location of certificate files.

## CACertFile

The CACertFile property specifies the name of the certificate file, or a file made up of a concatenated set of certificates, which is used in validating the connection between Unity and a Vantage system. The Unity file contains certificates for connected database systems, and the database file contains the certificates for connected Unity servers. Also see the topics beginning with [Coordinating Mechanism Property Values for Unity](#).

### Default Property Value

The default setting is "", that is, no cert file is specified.

### Valid Settings

Setting	Description
""	No cert file is specified.
A file name	The file that contains a certificate or concatenated set of certificates.

### Editing Guidelines

- Edit this property on Unity and connected database nodes. For an example, see [Coordinating Mechanism Property Values for Unity](#).

On a Unity server, the specified file contains the certificates for all connected Vantage systems. On a Vantage system, the file contains certificates for all Unity servers to which the database connects

- To set a value, copy the PROXY mechanism from the TdgssLibraryConfigFile.xml and add it to the TDGSS configuration file. See [About Editing Configuration Files](#).
- You can instead use the CACertDir property to locate the certificate files.

## CertificateFile

The CertificateFile property specifies the name of the file that contains the certificate used by a database system or on Unity when authenticating a Unity connection to the database. Unity and each connected database system must have its own certificate.

See [Coordinating Mechanism Property Values for Unity](#).

### Default Property Value

The default setting is "", that is, no file is specified.

## Valid Settings

Setting	Description
""	No certificate file is specified.
A file name	The name of the file that contains the certificate.

## Editing Guidelines

- Edit this property on the database and on Unity. For an example, see [Coordinating Mechanism Property Values for Unity](#).
- To set a value for the CertificateFile property, you must manually add the PROXY mechanism to the TDGSS configuration file and then specify the value as part of an update to the TDGSS configuration. See [Making Changes to TdgssUserConfigFile.xml on Database Nodes](#).

## PrivateKeyFile

PrivateKeyFile identifies the file that contains the private key for the certificate specified by the CertificateFile property. See [Coordinating Mechanism Property Values for Unity](#).

## Default Property Value

The default setting is "", that is, no private key file is specified.

## Valid Settings

Setting	Description
" "	No private key file is specified. <b>Note:</b> You must specify a value for the PrivateKeyFile property if users log on to Vantage through Unity.
A file name	Specifies the name of the file that contains the private key.

## Editing Guidelines

- To set a value, copy the PROXY mechanism from the TdgssLibraryConfigFile.xml and add it to the TDGSS configuration file. See [About Editing Configuration Files](#).
- Edit this property on database nodes and on the Unity server. For an example, see [Coordinating Mechanism Property Values for Unity](#).



## PrivateKeyPassword

The PrivateKeyPassword property specifies the password for the private key defined by the PrivateKey property. See [Coordinating Mechanism Property Values for Unity](#).

### Default Property Value

The default setting is "", that is, no password is specified.

### Valid Settings

Setting	Description
"" (default)	No password is specified.
a password	The password for the private key defined by the PrivateKey property.

### Editing Guidelines

- To set a value, copy the PROXY mechanism from the TdgssLibraryConfigFile.xml and add it to the TDGSS configuration file. See [About Editing Configuration Files](#).
- Edit this property only on database nodes. Also see [Coordinating Mechanism Property Values for Unity](#).

## PrivateKeyPasswordProtected

The PrivateKeyPasswordProtected property indicates whether the password specified for the PrivateKeyPassword property is stored in encrypted form.

### Default Property Value

The default setting is "yes/no", that is, the encryption status is not yet determined.

### Valid Settings

Setting	Description
"yes/no" (default)	The encryption status of the PrivateKeyPassword is not yet determined.
"yes"	Indicates that the private key password is stored in encrypted form (recommended).
"no"	The private key password is not stored in encrypted form.

### Editing Guidelines

- To set a value, copy the PROXY mechanism from the TdgssLibraryConfigFile.xml and add it to the TDGSS configuration file. See [About Editing Configuration Files](#).

- Edit this property only on Vantage nodes. See [Coordinating Mechanism Property Values for Unity](#).

## ProxySupported

Indicates whether the mechanism can act as a proxy, transferring user credentials and authorization information from the logon user authentication mechanism, through Teradata Unity, to Vantage.

### Default Property Value

The default setting is “yes”, that is, the containing mechanism supports proxy operation.

### Valid Settings

The value for ProxySupported is an indication of built in function. The only valid setting is “yes”, the default. The system ignores all other settings.

### Editing Guidelines

- This property is not editable.

## SigningHashAlgorithm

During connection of the Unity proxy to a Vantage system, Unity and the gateway mutually authenticate. Each side of the connection (each peer) digitally signs the DH public key using their private key. Then each side verifies the digital signature of its peer using the public key embedded in the certificate it receives from the peer. Rather than signing the 2048 bit DH public key, each peer takes a hash of the key and then signs the hashed data with the private key.

The SigningHashAlgorithm property indicates what hash algorithm is applied to the DH public key before performing the signature operation.

### Default Property Value

The default setting is “SHA256”.

### Valid Settings

Setting	Description
“SHA256” (default)	Specifies the SHA256 algorithm.
“SHA512”	Specifies the SHA512 algorithm, for stronger encryption.

### Editing Guidelines

- To set a value, copy the PROXY mechanism from theTdgssLibraryConfigFile.xml and add it to the TdgssUserConfigFile.xml. See [About Editing Configuration Files](#).
- Use the default setting if possible.

- Edit this property only on a Vantage system connected to Unity. As part of the token exchanges, the gateway communicates the hash algorithm to Unity.
- You can specify SHA512 for stronger hash encryption during proxy authentication, but there is a slight degradation in logon performance if the stronger encryption is used.

Also see [Coordinating Mechanism Property Values for Unity](#).

## Coordinating Mechanism Property Values for Unity

In a Unity environment, the values of certain mechanism properties must maintain a required relationship between the TdgssUnityConfig.xml on the Unity server and the TdgssUserConfigFile.xml on connected Teradata Vantage systems. Allowable property configurations depend on whether you allow logons through both Unity and directly to connected database systems, and whether you want the same behavior for all logons.

Before configuring a property value, check the configuration requirements for the property.

### Note:

Except for MechanismEnabled and DefaultMechanism, which apply to all mechanisms, mechanism properties need only be configured for the KRB5 and LDAP external authentication mechanisms, if used.

For information on Unity configuration, see *Teradata® Unity™ Installation, Configuration, and Upgrade Guide for Customers*, B035-2523.

The following table defines property configuration rules, but does not describe how to determine specific values. Properties appear in the table in the approximate order they appear in containing mechanisms. Properties not shown in the table are not configurable

Property	Configuration on Unity and Connected Database
<ul style="list-style-type: none"> <li>• AuthorizationSupported</li> <li>• MechanismEnabled</li> </ul>	<p>The values of these properties can vary by mechanism, but for a specific mechanism the value must be the same on Unity servers and all connected databases.</p> <p><b>Note:</b></p> <p>If clients connecting through Unity use the SPNEGO mechanism, you must copy SPNEGO from the TdgssLibraryConfigFile.xml to the TdgssUnityConfig.xml and set the MechanismEnabled property to yes on the Unity server. See <i>Teradata® Unity™ Installation, Configuration, and Upgrade Guide for Customers</i>, B035-2523.</p>
DefaultMechanism	<p>The default mechanism on the Unity servers and connected database systems must match if the same authentication behavior is required for clients connecting directly to the database as for those connecting through Unity.</p>
DelegateCredentials	<p>This property is not used for Unity, and is set to 'no' in the TdgssLibraryConfigFile.xml by default.</p>

Property	Configuration on Unity and Connected Database
	<p><b>Note:</b></p> <p>On systems that previously set this property to 'yes' in the TdgssUserConfigFile.xml for use with Teradata Query Director (discontinued), you should edit the value to 'no'.</p>
MutualAuthentication	Should be set to the same value on Unity servers and on all connected databases if the same authentication and authorization behavior is required for users logging on through Unity as those logging on directly to the connected database systems.
VerifyDHKey	Editable only on the TD2 mechanism. Can be set differently on each database system and Unity server.
TeradataKeyTab	<p>Specifies a location for the keytab file generated as part of setup for Kerberos authentication.</p> <p>The location can vary among database systems and Unity servers.</p>
UseLdapConfig	<p>The UseLdapConfig property tells Teradata GSS to look in a separate &lt;LdapConfig&gt; section for certain LDAP property values. The LdapConfig section defines multiple directory services and configures a set of related mechanism properties for each service.</p> <p><b>Note:</b></p> <p>Should be set to the same value on Unity servers and on all connected databases if the same authentication and authorization behavior is required for users logging on through Unity as those logging on directly to the connected database systems.</p>
<LdapConfig> section	<p>Among configuration files for Unity servers and connected database systems, each service within the &lt;LdapConfig&gt; section should have the same:</p> <ul style="list-style-type: none"> <li>• Service ID</li> <li>• LdapServerName name value</li> <li>• TLS properties and canonicalizations</li> </ul>
LdapServerName	<p>Identifies the authenticating LDAP directory or directories. The value can be the same on Unity servers and on connected database systems if all authentication is done in the same directory.</p> <p>In some cases, the value can be different on Unity servers than on connected database systems. For example, if users can log on either through Unity or directly to each connected database system, you can set the value differently for each configuration file to authenticate users in a directory local to the tdpid for the logon.</p>
<ul style="list-style-type: none"> <li>• LdapSystemFQDN</li> <li>• LdapBaseFQDN</li> <li>• LdapGroupBaseFQDN</li> <li>• LdapUserBaseFQDN</li> </ul>	<p>The LdapSystemFQDN identifies the top level system object in the directory that is the parent of the LDAP authorization structure. If directory users can only log on through Unity, only the LdapSystemFQDN configured on Unity is in effect.</p>

Property	Configuration on Unity and Connected Database
	<p>If directory users can log on either through Unity or directly to one or more of the Teradata Vantage systems managed by Unity:</p> <ul style="list-style-type: none"> <li>• The LdapSystemFQDN must also be configured on each database system, as well as on Unity.</li> <li>• For simplicity, the LdapSystemFQDN on all database systems and on Unity is normally configured with the same system object (and authorization structure). This is required if you configure IP restrictions.</li> </ul> <p>If Unity and connected database systems all point to the same system object, then for LdapBaseFQDN, LdapGroupBaseFQDN, and LdapUserBaseFQDN, the property value on Unity and on connected database systems should be the same.</p>
<ul style="list-style-type: none"> <li>• LdapServerRealm</li> <li>• LdapClientReferrals</li> <li>• LdapClientDeref</li> <li>• LdapClientRebindAuthorization</li> <li>• LdapClientUseTls</li> </ul>	<p>The value of each property in this group should be the same on Unity servers and all connected databases that use the same LdapServerName value.</p> <p>If the LdapServerName value is different among database systems or between a system and a Unity server, the value of these properties can also be different.</p>
LdapClientDebug	Can be set differently on Unity servers than on connected database systems.
<ul style="list-style-type: none"> <li>• LdapClientTlsRandFile</li> <li>• LdapClientRandomDevice</li> </ul>	<p>Identifies a system file or device that can generate a random number for use in certain LDAP processes.</p> <p>The value of each property can be different on Unity servers and connected Vantage systems.</p>
LdapClientMechanism	The value of this property must match between Unity servers and connected database systems if the same authentication behavior is required for clients connecting directly to the database as for those connecting through Unity.
<ul style="list-style-type: none"> <li>• LdapClientTlsCaCert</li> <li>• LdapClientTlsCaCertDir</li> </ul>	<p>The location of the certificate can be different among Unity servers and connected database systems.</p> <p>The contents of the file should be the same wherever the value of LdapServerName is the same.</p>
<ul style="list-style-type: none"> <li>• LdapClientTlsCert</li> <li>• LdapClientTlsKey</li> </ul>	The value of each property can vary among Unity servers and connected database systems.
<ul style="list-style-type: none"> <li>• LdapClientTlsReqCert</li> <li>• LdapClientTlsCipherSuite</li> <li>• LdapClientTlsCRLCheck</li> </ul>	The value of each property must match among Unity servers and connected database systems if the same authentication behavior is required for clients connecting directly to the database as for those connecting through Unity.
<ul style="list-style-type: none"> <li>• LdapServiceFQDN</li> <li>• LdapServicePassword</li> <li>• LdapServicePasswordFile</li> <li>• LdapServicePasswordProtected</li> </ul>	The value can vary between the Unity servers and connected database systems.

Property	Configuration on Unity and Connected Database
<ul style="list-style-type: none"> <li>LdapServiceBindRequired</li> <li>LdapClientSASLSecProps</li> </ul>	The value of each property should match between Unity servers and connected database systems if the same authentication behavior is required for clients connecting directly to the database as for those connecting through Unity.
LdapAllowUnsafeServerConnect	The property value should be the same on Unity servers and any connected database system that uses the same LdapServerName value.
<ul style="list-style-type: none"> <li>DHKeyP</li> <li>DHKeyG</li> <li>DHkeyP2048</li> <li>DHKeyG2048</li> </ul>	The value of each property does not need to be the same on Unity servers and connected database systems Teradata recommends that you do not edit these values.
MechQOP elements (legacy, default, low, medium, and high)	The configuration of each element should match between Unity servers and connected database systems if the same authentication behavior is required for clients connecting directly to the database as for those connecting through Unity.
<LdapConfig>	
IdentityMap and IdentitySearch elements	
RequiredLibrary element (KRB5 only)	The filename does not need to match between Unity servers and connected database systems, but the Kerberos packages contained in the file must be the same version.
PROXY mechanism properties	See <i>Teradata® Unity™ Installation, Configuration, and Upgrade Guide for Customers</i> , B035-2523 and <i>Teradata® Unity™ User Guide</i> , B035-2520.
<ul style="list-style-type: none"> <li>ProxySupported</li> </ul>	Set to yes on Unity servers and all connected database systems.
<ul style="list-style-type: none"> <li>CertificateFile</li> <li>PrivateKeyFile</li> </ul>	File names do not need to match between Unity servers and connected database systems.
<ul style="list-style-type: none"> <li>PrivateKeyPassword</li> <li>PrivateKeyPasswordProtected</li> <li>SigningHashAlgorithm</li> </ul>	Only configured on database systems.
<ul style="list-style-type: none"> <li>CACertFile</li> <li>CACertDir</li> </ul>	File and directory names do not need to match between Unity servers and connected database systems, however, the file structure from which the values of these properties are taken, does use similar naming.

## Quality of Protection (QOP)

A TDGSS Global Quality of Protection (QOP) defines the set of encryption methods and attributes used by the TD2 and LDAP security mechanisms to support confidentiality and integrity.

**Note:**

For Kerberos authentication (KRB5 or SPNEGO mechanisms), the system uses the encryption standard negotiated by Kerberos. See [Message Encryption for Kerberos Authentication](#).

A QOP is based on a TDGSS algorithm, and has some of the same attributes. Algorithm definitions specify all allowable attribute values, whereas an individual QOP contains only one value for each attribute.

The following table describes the attributes present in a QOP.

Attribute	Description
Value	Global QOP Identifier Each security mechanism calls out a QOP to define how it performs encryption.
ConfidentialityAlgorithm	The algorithm designated by the QOP to perform encryption.
KeyLength	The length of the key used for message encryption.
Mode	The encryption mode type used by the encryption algorithm.
Padding	The encryption padding type used by the encryption algorithm.
IntegrityAlgorithm	The algorithm that the QOP designates to maintain message integrity during the encryption/transmission/decryption cycle.
Key Exchange Algorithm	The algorithm used to perform the encryption key exchange.
KeyLengthP	The length of the key used to perform the encryption key exchange

## Global QOPs

```
<GlobalQOPs>

  <!-- One entry for each QOP type goes here -->

  <GlobalQOP
    Value="GLOBAL_QOP_0"
    ConfidentialityAlgorithm="Blowfish"
    KeyLength="K128"
    Mode="ECB"
    Padding="NoPadding"
    IntegrityAlgorithm="NONE"/>

  <!-- Used by Teradata Method 2 -->
  <GlobalQOP
    Value="GLOBAL_QOP_1"
    ConfidentialityAlgorithm="AES"
```

```

    KeyLength="K128"
    Mode="OFB"
    Padding="PKCS5Padding"
    IntegrityAlgorithm="SHA1"/>

<GlobalQOP
    Value="AES-K128_CBC_PKCS5Padding_SHA1_DH-K2048"
    ConfidentialityAlgorithm="AES"
    KeyLength="K128"
    Mode="CBC"
    Padding="PKCS5Padding"
    IntegrityAlgorithm="SHA1"
    KeyExchangeAlgorithm="DH"
    KeyLengthP="K2048"/>

<GlobalQOP
    Value="AES-K192_CBC_PKCS5Padding_SHA1_DH-K2048"
    ConfidentialityAlgorithm="AES"
    KeyLength="K192"
    Mode="CBC"
    Padding="PKCS5Padding"
    IntegrityAlgorithm="SHA1"
    KeyExchangeAlgorithm="DH"
    KeyLengthP="K2048"/>

<GlobalQOP
    Value="AES-K256_CBC_PKCS5Padding_SHA1_DH-K2048"
    ConfidentialityAlgorithm="AES"
    KeyLength="K256"
    Mode="CBC"
    Padding="PKCS5Padding"
    IntegrityAlgorithm="SHA1"
    KeyExchangeAlgorithm="DH"
    KeyLengthP="K2048"/>
<GlobalQOP
    Value="AES-K128_GCM_PKCS5Padding_SHA2_DH-K2048"
    ConfidentialityAlgorithm="AES"
    KeyLength="K128"
    Mode="GCM"
    Padding="PKCS5Padding"
    IntegrityAlgorithm="SHA256"
    KeyExchangeAlgorithm="DH"
    KeyLengthP="K2048"/>
<GlobalQOP

```



```

Value="AES-K256_GCM_PKCS5Padding_SHA2_DH-K2048"
ConfidentialityAlgorithm="AES"
KeyLength="K256"
Mode="GCM"
Padding="PKCS5Padding"
IntegrityAlgorithm="SHA256"
KeyExchangeAlgorithm="DH"
KeyLengthP="K2048"/>
<GlobalQOP
Value="AES-K128_CCM_PKCS5Padding_SHA2_DH-K2048"
ConfidentialityAlgorithm="AES"
KeyLength="K128"
Mode="CCM"
Padding="PKCS5Padding"
IntegrityAlgorithm="SHA256"
KeyExchangeAlgorithm="DH"
KeyLengthP="K2048"/>
<GlobalQOP
Value="AES-K192_CCM_PKCS5Padding_SHA2_DH-K2048"
ConfidentialityAlgorithm="AES"
KeyLength="K192"
Mode="CCM"
Padding="PKCS5Padding"
IntegrityAlgorithm="SHA256"
KeyExchangeAlgorithm="DH"
KeyLengthP="K2048"/>
<GlobalQOP
Value="AES-K256_CCM_PKCS5Padding_SHA2_DH-K2048"
ConfidentialityAlgorithm="AES"
KeyLength="K256"
Mode="CCM"
Padding="PKCS5Padding"
IntegrityAlgorithm="SHA256"
KeyExchangeAlgorithm="DH"
KeyLengthP="K2048"/>
<GlobalQOP
Value="AES-K128_CTR_PKCS5Padding_SHA2_DH-K2048"
ConfidentialityAlgorithm="AES"
KeyLength="K128"
Mode="CTR"
Padding="PKCS5Padding"
IntegrityAlgorithm="SHA256"
KeyExchangeAlgorithm="DH"
KeyLengthP="K2048"/>

```

```

<GlobalQOP
  Value="AES-K192_CTR_PKCS5Padding_SHA2_DH-K2048"
  ConfidentialityAlgorithm="AES"
  KeyLength="K192"
  Mode="CTR"
  Padding="PKCS5Padding"
  IntegrityAlgorithm="SHA256"
  KeyExchangeAlgorithm="DH"
  KeyLengthP="K2048"/>
<GlobalQOP
  Value="AES-K256_CTR_PKCS5Padding_SHA2_DH-K2048"
  ConfidentialityAlgorithm="AES"
  KeyLength="K256"
  Mode="CTR"
  Padding="PKCS5Padding"
  IntegrityAlgorithm="SHA256"
  KeyExchangeAlgorithm="DH"
  KeyLengthP="K2048"/>
<GlobalQOP
  Value="AES-K128_AEADGCM_PKCS5Padding_SHA2_DH-K2048"
  ConfidentialityAlgorithm="AES"
  KeyLength="K128"
  Mode="AEADGCM"
  Padding="PKCS5Padding"
  IntegrityAlgorithm="SHA256"
  KeyExchangeAlgorithm="DH"
  KeyLengthP="K2048"/>
<GlobalQOP
  Value="AES-K192_AEADGCM_PKCS5Padding_SHA2_DH-K2048"
  ConfidentialityAlgorithm="AES"
  KeyLength="K192"
  Mode="AEADGCM"
  Padding="PKCS5Padding"
  KeyExchangeAlgorithm="DH"
  KeyLengthP="K2048"/>
<GlobalQOP
  Value="AES-K256_AEADGCM_PKCS5Padding_SHA2_DH-K2048"
  ConfidentialityAlgorithm="AES"
  KeyLength="K256"
  Mode="AEADGCM"
  Padding="PKCS5Padding"
  IntegrityAlgorithm="SHA256"
  KeyExchangeAlgorithm="DH"
  KeyLengthP="K2048"/>

```

```
</GlobalQOPs>
```

```
<GlobalQOPs>
  <!-- One entry for each QOP type goes here -->
  <GlobalQOP
    Value="GLOBAL_QOP_0"
    ConfidentialityAlgorithm="Blowfish"
    KeyLength="K128"
    Mode="ECB"
    Padding="NoPadding"
    IntegrityAlgorithm="NONE"/>
  <!-- Used by Teradata Method 2 -->
  <GlobalQOP
    Value="GLOBAL_QOP_1"
    ConfidentialityAlgorithm="AES"
    KeyLength="K128"
    Mode="OFB"
    Padding="PKCS5Padding"
    IntegrityAlgorithm="SHA1"/>
  <GlobalQOP
    Value="AES-K128_CBC_PKCS5Padding_SHA1_DH-K2048"
    ConfidentialityAlgorithm="AES"
    KeyLength="K128"
    Mode="CBC"
    Padding="PKCS5Padding"
    IntegrityAlgorithm="SHA1"
    KeyExchangeAlgorithm="DH"
    KeyLengthP="K2048"/>
  <GlobalQOP
    Value="AES-K192_CBC_PKCS5Padding_SHA1_DH-K2048"
    ConfidentialityAlgorithm="AES"
    KeyLength="K192"
    Mode="CBC"
    Padding="PKCS5Padding"
    IntegrityAlgorithm="SHA1"
    KeyExchangeAlgorithm="DH"
    KeyLengthP="K2048"/>
  <GlobalQOP
    Value="AES-K256_CBC_PKCS5Padding_SHA1_DH-K2048"
    ConfidentialityAlgorithm="AES"
    KeyLength="K256"
    Mode="CBC"
```

```

    Padding="PKCS5Padding"
    IntegrityAlgorithm="SHA1"
    KeyExchangeAlgorithm="DH"
    KeyLengthP="K2048"/>
<GlobalQOP
    Value="AES-K128_GCM_PKCS5Padding_SHA2_DH-K2048"
    ConfidentialityAlgorithm="AES"
    KeyLength="K128"
    Mode="GCM"
    Padding="PKCS5Padding"
    IntegrityAlgorithm="SHA256"
    KeyExchangeAlgorithm="DH"
    KeyLengthP="K2048"/>
<GlobalQOP
    Value="AES-K192_GCM_PKCS5Padding_SHA2_DH-K2048"
    ConfidentialityAlgorithm="AES"
    KeyLength="K192"
    Mode="GCM"
    Padding="PKCS5Padding"
    IntegrityAlgorithm="SHA256"
    KeyExchangeAlgorithm="DH"
    KeyLengthP="K2048"/>
<GlobalQOP
    Value="AES-K256_GCM_PKCS5Padding_SHA2_DH-K2048"
    ConfidentialityAlgorithm="AES"
    KeyLength="K256"
    Mode="GCM"
    Padding="PKCS5Padding"
    IntegrityAlgorithm="SHA256"
    KeyExchangeAlgorithm="DH"
    KeyLengthP="K2048"/>
<GlobalQOP
    Value="AES-K128_CCM_PKCS5Padding_SHA2_DH-K2048"
    ConfidentialityAlgorithm="AES"
    KeyLength="K128"
    Mode="CCM"
    Padding="PKCS5Padding"
    IntegrityAlgorithm="SHA256"
    KeyExchangeAlgorithm="DH"
    KeyLengthP="K2048"/>
<GlobalQOP
    Value="AES-K192_CCM_PKCS5Padding_SHA2_DH-K2048"
    ConfidentialityAlgorithm="AES"
    KeyLength="K192"

```

```

    Mode="CCM"
    Padding="PKCS5Padding"
    IntegrityAlgorithm="SHA256"
    KeyExchangeAlgorithm="DH"
    KeyLengthP="K2048"/>
  <GlobalQOP
    Value="AES-K256_CCM_PKCS5Padding_SHA2_DH-K2048"
    ConfidentialityAlgorithm="AES"
    KeyLength="K256"
    Mode="CCM"
    Padding="PKCS5Padding"
    IntegrityAlgorithm="SHA256"
    KeyExchangeAlgorithm="DH"
    KeyLengthP="K2048"/>
  <GlobalQOP
    Value="AES-K128_CTR_PKCS5Padding_SHA2_DH-K2048"
    ConfidentialityAlgorithm="AES"
    KeyLength="K128"
    Mode="CTR"
    Padding="PKCS5Padding"
    IntegrityAlgorithm="SHA256"
    KeyExchangeAlgorithm="DH"
    KeyLengthP="K2048"/>
  <GlobalQOP
    Value="AES-K192_CTR_PKCS5Padding_SHA2_DH-K2048"
    ConfidentialityAlgorithm="AES"
    KeyLength="K192"
    Mode="CTR"
    Padding="PKCS5Padding"
    IntegrityAlgorithm="SHA256"
    KeyExchangeAlgorithm="DH"
    KeyLengthP="K2048"/>
  <GlobalQOP
    Value="AES-K256_CTR_PKCS5Padding_SHA2_DH-K2048"
    ConfidentialityAlgorithm="AES"
    KeyLength="K256"
    Mode="CTR"
    Padding="PKCS5Padding"
    IntegrityAlgorithm="SHA256"
    KeyExchangeAlgorithm="DH"
    KeyLengthP="K2048"/>
</GlobalQOPs>

```

## Mechanism QOPs

Each applicable mechanism contains QOP options that can be set for that mechanism in the TdgssUserConfigFile.xml. The following example shows the TdgssUserConfigFile.xml for a fresh install.

For an explanation of QOP options and instructions on making QOP settings, see [Working with Quality of Protection Options](#).

```
<!-- Teradata Method 2 (uses AES) -->
<Mechanism Name="TD2">
  <!-- DHKeyP and DHKeyG are for legacy (pre-14.0) use only -->
  <MechanismProperties
    ...
  />
  <!-- To update security uncomment one or more QOPs and edit. -->
  <!-- DEFAULT QOP
  <MechQop Value="Default">
    AES-K128_GCM_PKCS5Padding_SHA2_DH-K2048
    AES-K128_CBC_PKCS5Padding_SHA1_DH-K2048
    AES-K192_GCM_PKCS5Padding_SHA2_DH-K2048
    AES-K192_CBC_PKCS5Padding_SHA1_DH-K2048
    AES-K256_GCM_PKCS5Padding_SHA2_DH-K2048
    AES-K256_CBC_PKCS5Padding_SHA1_DH-K2048
  </MechQop>
  -->
  <!-- LOW SECURITY QOP
  <MechQop Value="Low">
    AES-K128_CBC_PKCS5Padding_SHA1_DH-K2048
  </MechQop>
  -->
  <!-- MEDIUM SECURITY QOP
  <MechQop Value="Medium">
    AES-K192_CBC_PKCS5Padding_SHA1_DH-K2048
  </MechQop>
  -->
  <!-- HIGH SECURITY QOP
  <MechQop Value="High">
    AES-K256_CBC_PKCS5Padding_SHA1_DH-K2048
  </MechQop>
  -->
</Mechanism>
```

## Mechanism Configurations

Teradata Vantage currently provides a set of pre-defined security mechanisms:

- TD2
- KRB5
- SPNEGO
- TDNEGO
- LDAP
- PROXY
- JWT

## TD2 Mechanism

Teradata Method 2 (TD2) supports user authentication and authorization by Teradata Vantage, and the capability for encryption of data (confidentiality) and integrity checking. It is the preset default mechanism on the client.

### Example: TD2 TDGSS Library Configuration

```
<Mechanism Name="TD2"
  ObjectId="1.3.6.1.4.1.191.1.1012.1.1.9"
  LibraryName="gssp2td2"
  Prefix="TD2"
  InterfaceType="teradata">
<!-- Note: DHKeyP and DHKeyG are for legacy (pre-14.0) use only -->
<MechanismProperties
  AuthenticationSupported="no"
  AuthorizationSupported="no"
  SingleSignOnSupported="no"
  DefaultMechanism="no"
  MechanismEnabled="yes"
  MechanismRank="20"
  MechanismIgnoresQop="no"
  DelegateCredentials="no"
  MutualAuthentication="yes"
  ReplayDetection="yes"
  OutOfSequenceDetection="yes"
  ConfidentialityDesired="yes"
  IntegrityDesired="yes"
  AnonymousAuthentication="no"
  DesiredContextTime=""
  DesiredCredentialTime=""
  CredentialUsage="0"
  VerifyDHKey="no"

DHKeyP="E4BE0A78F54C4A0B17E7E9249A78BCC08868C17281D8463C880937853E73DDC787E41580
A8AFE2594D984C9E0814C590790354ECCD1BE8EA85961E5E0974B32EFE178335F061E80189B4BDAA
20F67B47"

DHKeyG="00000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000
00000005"

DHKeyP2048="8AB3F86E8D374B782F31DAD5F27D6AFDA30150C11A20CF6346712AE2D2C6B70A5B79
D45D4C0C232A065B207B121B2C33E147B5983C38A1087F272703B0B839CBA6F71C5D0EB51EC89093
4EACF2C7DD2A1DF6F55E89B145A0359D35EF8FB6C561E157B13FF927A35E69963648614902B1034E
F71197F545DEF3236244EADAE0689E624CF1245953630AE042BD797C4025E37C51D9F6CBDA0B2278
```

[illegible]

## KRB5 Mechanism

The KRB5 mechanism supports Kerberos user authentication and Teradata Vantage authorization. You can optionally configure the KRB5 mechanism to specify directory authorization of users. This option also requires configuration of the directory. See [Option 3: Non-LDAP External Authentication with Directory Authorization](#).

These are the types of KRB5 mechanisms:

- SSPI Kerberos appears on Windows clients
- KRB5 for UNIX appears on Linux clients, on supported TTU UNIX clients (except IBM z/OS clients), and on the database system

To use the KRB5 mechanism, you must complete the set up procedures described in the topics starting with [About External Authentication Controls](#).

**Note:**

For clients running .NET Data Provider for Teradata, you must use the SPNEGO mechanism for Kerberos authentication.



## Kerberos Multiple LAN Adapter Restriction

When you use Kerberos authentication, for example, when users employ single sign-on, Vantage nodes can have a maximum of one LAN adapter, and the machine name must correspond to the host name (*hostid*) associated with the target adapter. If a logon uses KRB5 to connect to a node with multiple LAN adapters, the logon fails.

If you decide to use multiple LAN adapters, you can disable the KRB5 mechanism to avoid logon failures. See [MechanismEnabled](#).

## Example: KRB5 for Linux Configuration in Teradata Vantage

Linux appears in the TdgssUserConfigFile.xml by default.

### Note:

If you decide to use directory authorization with Kerberos authentication, you must configure at least some of the LDAP properties. See [Option 3: Non-LDAP External Authentication with Directory Authorization](#).

```
<!-- KRB5 for TDGSS using GSS-API -->
  <Mechanism Name="KRB5"
    ObjectId="1.2.840.113554.1.2.2"
    LibraryName="gssp2gss"
    Prefix="gssp2gss"
    InterfaceType="gss">
    <RequiredLibrary Path="/usr/lib64/libgssapi_krb5.so"/>
    <MechanismProperties
      AuthenticationSupported="yes"
      AuthorizationSupported="no"
      SingleSignOnSupported="yes"
      DefaultMechanism="no"
      MechanismEnabled="yes"
      MechanismRank="40"
      GenerateCredentialFromLogon="yes"
      DelegateCredentials="no"
      MutualAuthentication="yes"
      ReplayDetection="yes"
      OutOfSequenceDetection="yes"
      ConfidentialityDesired="yes"
      IntegrityDesired="yes"
      AnonymousAuthentication="no"
      DesiredContextTime=""
      DesiredCredentialTime=""
      CredentialUsage="0"
      LdapServerName=""
```

```

LdapServerPort="389"
LdapServerRealm=""
LdapSystemFQDN=""
LdapBaseFQDN=""
LdapGroupBaseFQDN=""
LdapUserBaseFQDN=""
LdapClientReferrals="off"
LdapClientDeref="never"
LdapClientDebug="0"
LdapClientRebindAuth="yes"
LdapClientRandomDevice="/dev/urandom"
LdapClientMechanism="SASL/DIGEST-MD5"
LdapClientUseTls="no"
LdapServiceFQDN=""
LdapServicePasswordProtected="no"
LdapServicePasswordFile=""
LdapServicePassword=""
LdapClientSaslSecProps=""
UseLdapConfig="no"
TeradataKeyTab="/etc/teradata.keytab"
/>
  <MechQop Value="0"> GLOBAL_QOP_0 </MechQop>
</Mechanism>

```

## SPNEGO Mechanism

The SPNEGO mechanism supports Kerberos authentication for users that log on to Teradata Vantage from .NET clients, and functions similarly to the KRB5 mechanism.

---

### Note:

The SPNEGO mechanism is derived from the KRB5 mechanism, and is therefore subject to the multiple LAN adapter restriction. For further information, see [Kerberos Multiple LAN Adapter Restriction](#).

---

SPNEGO appears in the TdgssLibraryConfigFile.xml for all installations of Teradata Vantage and Unity; however, to make a configuration change to SPNEGO, you must manually copy the mechanism from the TdgssLibraryConfigFile.xml and add it to the TdgssUserConfigFile.xml (Teradata Vantage) and to the TdgssUnityConfig.xml (Unity). Then follow the instructions in *Teradata® Unity™ Installation, Configuration, and Upgrade Guide for Customers*, B035-2523 to update the Unity configuration file. Note, this copy only needs to be done for configuration changes; otherwise, the default configuration can remain in TdgssLibraryConfigFile.xml.

**Example: SPNEGO Configuration****Note:**

If you decide to use SPNEGO with directory authorization, you must add and configure some optional properties not shown in this example. See [Option 3: Non-LDAP External Authentication with Directory Authorization](#).

```
<!-- SPNEGO for UNIX Teradata servers -->
  <Mechanism Name="SPNEGO"
    ObjectId="1.3.6.1.5.5.2"
    LibraryName="gssp2spnego"
    Prefix="spnego"
    InterfaceType="negotiate">
    <MechanismProperties
      AuthenticationSupported="yes"
      AuthorizationSupported="no"
      SingleSignOnSupported="yes"
      DefaultMechanism="no"
      MechanismEnabled="yes"
      MechanismRank="65"
      DelegateCredentials="no"
      MutualAuthentication="yes"
      ReplayDetection="yes"
      OutOfSequenceDetection="yes"
      ConfidentialityDesired="yes"
      IntegrityDesired="yes"
      AnonymousAuthentication="no"
      DesiredContextTime=""
      DesiredCredentialTime=""
      CredentialUsage="0"
      LdapServerName=""
      LdapServerPort="389"
      LdapServerRealm=""
      LdapSystemFQDN=""
      LdapBaseFQDN=""
      LdapGroupBaseFQDN=""
      LdapUserBaseFQDN=""
      LdapClientReferrals="off"
      LdapClientDeref="never"
      LdapClientDebug="0"
      LdapClientRebindAuth="yes"
      LdapClientRandomDevice="/dev/urandom"
      LdapClientMechanism="SASL/DIGEST-MD5"
```

```

        LdapClientUseTls="no"
        LdapServiceFQDN=""
        LdapServicePasswordProtected="no"
        LdapServicePassword=""
        LdapClientSaslSecProps=""
        UseLdapConfig="no"
    />
    <MechQop Value="0"> GLOBAL_QOP_1 </MechQop>
    <NegotiatedMechanism ObjectId="1.2.840.113554.1.2.2"/>
</Mechanism>

```

## TDNEGO Mechanism

TDNEGO is enabled by default on both the client and server and can be used by the client explicitly requesting it or TDNEGO can be used by setting it as the default negotiating mechanism on the server.

To change the TDNEGO configuration, uncomment and edit the TDNEGO mechanism section in the TdgssUserConfigFile.xml. For an example of configuring TDNEGO see [Configuring TDNEGO Properties](#).

## TDNEGO Mechanism Properties

The following TDGSS mechanism properties are not available and not used with TDNEGO.

- MechanismIgnoresQop
- VerifyDHKey
- DHKeyP and DHKeyG
- DHKeyP2048 and DHKeyG2048

The following TDGSS mechanism properties are set as shown in the table for the TDNEGO mechanism and may or may not be modified.

Do Not Modify These Properties	These Properties May Be Modified
<ul style="list-style-type: none"> <li>• AuthenticationSupported="yes"</li> <li>• AuthorizationSupported="yes"</li> <li>• SingleSignOnSupported="yes"</li> <li>• GenerateCredentialsFromLogon="yes"</li> <li>• NegotiationSupported="yes"</li> <li>• MutualAuthentication="yes"</li> <li>• ReplayDetection="yes"</li> <li>• OutOfSequenceDetection="yes"</li> <li>• ConfidentialityDesired="yes"</li> <li>• IntegrityDesired="yes"</li> </ul>	<ul style="list-style-type: none"> <li>• MechanismEnabled="yes"</li> <li>• DefaultMechanism="no"</li> <li>• DefaultNegotiatingMechanism="no"</li> <li>• MechanismRank="10"</li> </ul>

TDNEGO requires a complete LdapConfig section at the server, if mechanism policy is used as a selection criterion. If you are not using mechanism policy to restrict the mechanisms available for negotiation, the LdapConfig settings are not required.

The NegotiationSupported property is boolean and signifies that a mechanism is actually a pseudo mechanism that negotiates between the client and the server to find an appropriate mechanism to use. All mechanisms that fit this description will have NegotiationSupported set to yes; TDNEGO has this property set to yes. If the mechanism has NegotiationSupported set to yes it must also have at least one NegotiatedMechanism ObjectId set, otherwise an error is reported by the tdgssconfig executable. The NegotiationSupported property should not be modified.

The DefaultNegotiatingMechanism mechanism property is boolean and is similar to the DefaultMechanism property. DefaultNegotiatingMechanism is used to specify a default negotiating mechanism from among those mechanisms whose NegotiationSupported mechanism property is yes. DefaultNegotiatingMechanism is used to determine the default mechanism when both the client and the server support mechanism negotiation. The default setting for DefaultNegotiatingMechanism is no.

If both client and server support negotiation, but no common negotiating mechanism exists between them, the existing DefaultMechanism property is used to select a default mechanism.

## Example: TDNEGO Configuration

The following shows the TDNEGO configuration in the TdgssLibraryConfigFile.xml file.

```
<!-- TDNEGO: Teradata Negotiated Method -->
<Mechanism Name="TDNEGO"
  ObjectId="1.3.6.1.4.1.28698.4.302.1.3"
  LibraryName="gssp2tdnego"
  Prefix="TDNEGO"
  InterfaceType="negotiate">
  <MechanismProperties
    AuthenticationSupported="yes"
    AuthorizationSupported="yes"
    SingleSignOnSupported="yes"
    GenerateCredentialFromLogon="yes"
    NegotiationSupported="yes"
    MechanismEnabled="yes"
    DefaultMechanism="no"
    DefaultNegotiatingMechanism="no"
    MechanismRank="10"
    MutualAuthentication="yes"
    ReplayDetection="yes"
    OutOfSequenceDetection="yes"
    ConfidentialityDesired="yes"
    IntegrityDesired="yes"
```

```

    />
    <!-- Mechanisms offered for negotiation: KRB5, SPNEGO, ldap, TD2 -->
    <NegotiatedMechanism ObjectId="1.2.840.113554.1.2.2" Enable="yes"/>
    <NegotiatedMechanism ObjectId="1.3.6.1.5.5.2" Enable="yes"/>
    <NegotiatedMechanism ObjectId="1.3.6.1.4.1.191.1.1012.1.20" Enable="yes"/>
    <NegotiatedMechanism ObjectId="1.3.6.1.4.1.191.1.1012.1.1.9" Enable="yes"/>
  </Mechanism>

```

## LDAP Mechanism

The LDAP mechanism supports directory authentication and authorization of users that are defined in an LDAP-compliant directory. To use the LDAP mechanism you must complete set up procedures described in:

- [About External Authentication Controls](#).
- [Directory Management of Database Users](#).

---

### Note:

Several of the LDAP properties shown do not appear in the TdgssUserConfigFile.xml. You must add them to the TdgssUserConfigFile.xml to configure non-default values.

---

You can modify some LDAP support properties without performing a TPA reset. run\_tdgssconfig indicates when you need to do a TPA reset. For example, AuthorizationSupported, MechanismEnabled, any property beginning with “Ldap”, and the canonicalizations are all updated without a TPA reset requirement. But as always, rely on run\_tdgssconfig to tell you when a TPA reset is required.

### Example: LDAP Configuration

```

<!-- LDAPv3 -->
<Mechanism Name="ldap"
  ObjectId="1.3.6.1.4.1.191.1.1012.1.20"
  LibraryName="gssp2ldap"
  Prefix="ldapv3"
  InterfaceType="custom">
  <!-- Note: DHKeyP and DHKeyG are for legacy (pre-14.0) use only -->
  <MechanismProperties
    AuthenticationSupported="yes"
    AuthorizationSupported="yes"
    SingleSignOnSupported="no"
    DefaultMechanism="no"
    MechanismEnabled="yes"
    MechanismRank="70"
    MechanismIgnoresQop="no"
    GenerateCredentialFromLogon="yes"
  </MechanismProperties>
</Mechanism>

```

[illegible]

```

    LdapClientUseTls="no"
    LdapClientTlsCACert=""
    LdapClientTlsCACertDir=""
    LdapClientTlsCert=""
    LdapClientTlsKey=""
    LdapClientTlsRandFile=""
    LdapClientTlsReqCert="never"
    LdapClientTlsCipherSuite=""
    LdapClientTlsCRLCheck="none"
    LdapServiceFQDN=""
    LdapServicePasswordProtected="no"
    LdapServicePassword=""
    LdapServiceBindRequired="no"
    LdapClientSaslSecProps=""
    LdapAllowUnsafeServerConnect="yes"
    UseLdapConfig="no"
  />
<!-- Low, Medium and High QOP values are all set to "Default"
      unless the Low, Medium and High values are explicitly set
      in TdgssUserConfigFile.xml -->
<!-- DEFAULT QOP -->
  <MechQop Value="Default">
    AES-K128_GCM_PKCS5Padding_SHA2_DH-K2048
    AES-K128_CBC_PKCS5Padding_SHA1_DH-K2048
    AES-K192_GCM_PKCS5Padding_SHA2_DH-K2048
    AES-K192_CBC_PKCS5Padding_SHA1_DH-K2048
    AES-K256_GCM_PKCS5Padding_SHA2_DH-K2048
    AES-K256_CBC_PKCS5Padding_SHA1_DH-K2048
  </MechQop>
</Mechanism>

```

## PROXY Mechanism

The PROXY mechanism supports user logons through Unity, acting as a proxy for the authentication mechanism in effect at logon and passing user credential information to connected Teradata Vantage systems.

PROXY appears in the TdgssLibraryConfigFile.xml for all installations of Vantage, however, to make a configuration change to PROXY, you must manually copy the mechanism from the TdgssLibraryConfigFile.xml and add it to the TDGSS configuration file.



[illegible]

```

        AES-K192_GCM_PKCS5Padding_SHA2_DH-K2048
        AES-K192_CBC_PKCS5Padding_SHA1_DH-K2048
        AES-K256_GCM_PKCS5Padding_SHA2_DH-K2048
        AES-K256_CBC_PKCS5Padding_SHA1_DH-K2048
    </MechQop>
</Mechanism>

```

## JWT Mechanism

The JSON Web Token (JWT) authentication mechanism enables single sign-on (SSO) to Teradata Vantage after the user successfully authenticates to Teradata UDA User Service. The UDA User Service authenticates users to various UDA applications and services, such as AppCenter and the Teradata® Query Service (REST services). JWT allows a user that has been authenticated to one of the applications or services to do a single sign-on to establish a session with Teradata Vantage.

You can modify some JWT support properties without performing a TPA reset. For example, you can modify MechanismEnabled and any mechanism property that begins with JWT. You can also add, remove, and modify <IdentityProvider> and <UserNameMapping> elements without a TPA reset. run\_tdgssconfig indicates when you need to do a TPA reset.

### JWT Library Configuration: TdgssLibraryConfigFile.xml

```

<!-- JWT: JSON Web Token -->

<Mechanism Name="JWT"
  ObjectId="1.3.6.1.4.1.28698.4.302.1.4"
  LibraryName="gssp2jwt"
  Prefix="JWT"
  InterfaceType="custom">

  <MechanismProperties
    AuthenticationSupported="yes"
    AuthorizationSupported="no"
    SingleSignOnSupported="yes"

    NegotiationSupported="no"
    DefaultNegotiatingMechanism="no"

    DefaultMechanism="no"
    MechanismEnabled="yes"
    MechanismRank="30"
    MechanismIgnoresQop="no"

    GenerateCredentialFromLogon="yes"

    DelegateCredentials="no"
  </MechanismProperties>
</Mechanism>

```

```
<!-- Low, Medium and High QOP values are all set to "Default"
      unless the Low, Medium and High values are explicitly set
      in TdgssUserConfigFile.xml -->
```

```

<!-- DEFAULT QOP -->
<MechQop Value="Default">
    AES-K128_AEADGCM_PKCS5Padding_SHA2_DH-K2048
    AES-K192_AEADGCM_PKCS5Padding_SHA2_DH-K2048
    AES-K256_AEADGCM_PKCS5Padding_SHA2_DH-K2048
    AES-K128_GCM_PKCS5Padding_SHA2_DH-K2048
    AES-K192_GCM_PKCS5Padding_SHA2_DH-K2048
    AES-K256_GCM_PKCS5Padding_SHA2_DH-K2048
    AES-K128_CBC_PKCS5Padding_SHA1_DH-K2048
    AES-K192_CBC_PKCS5Padding_SHA1_DH-K2048
    AES-K256_CBC_PKCS5Padding_SHA1_DH-K2048
</MechQop>

</Mechanism>

```

### Example: JWT in the User Configuration File

The following is a snippet from TdgssUserConfigFile.xml showing the JWT mechanism:

```

<!-- JWT -->
<!-- To modify JWT mechanism configuration, uncomment this section and edit --
>
<Mechanism Name="JWT">
    <MechanismProperties
        MechanismEnabled="yes"
        DefaultMechanism="no"

        JWTDynamicKey="yes"
        JWTTokenExchange="yes"
        JWTClientTlsCACertDir="/etc/ssl/certs"
    />

    <TokenExchanger
        Ref="Ping1"
        ClientId="account"
        ClientSecret="Y2I20GZkZTctM2FjMCM00WQwLWIzMGUtODJjMGIxNTY2NzAy"
        ClientSecretProtected="yes"
    />

    <IdentityProvider
        Id="Keycloak21"
        Url="https://keycloak1/auth/realms/uda"
        Type="keycloak"
    />

```

```

        ValidateByTokenExchange="yes"
    />

    <IdentityProvider
        Id="ping1"
        Url="https://auth.pingone.asia/0cea60dc-0279-4b55-98a1-
eca07904733a/as"
        Type="PingFederate"
        ValidateByTokenExchange="no"
    />

    <UserNameMapping
        Claim="given_name"
        Match="(\w+)"
        DatabaseName="{1}"
    />

    <UserNameMapping
        Claim="sub"
        Match="(\w+).*.com"
        DatabaseName="{1}"
    />

    <UserNameMapping
        Claim="preferred_username"
        Match="(\w+)@(\w+).com"
        DatabaseName="{1}"
    />
</Mechanism>
(end of commented out section)-->

```

## Related Information

For more information about JWT, see <https://tools.ietf.org/html/rfc7519>.

For more information about configuring JWT, see [Configuring Single Sign-On](#).

For more information about JWT properties, see [JWT Support Properties](#).

# Diagnostic Tools

## tdgssgetinfo

`tdgssgetinfo` is a diagnostic tool that collects and displays information useful in determining the health of the TeraGSS or TDGSS installed on the system. It displays:

- Details about the system from which the tool is run.
- Information returned in the fourth parameter from a call to `tdgss_configure()`.
- The currently active version, as well as all of the installed versions on the system.
- The available security mechanisms and the locations of the configuration files.

In addition, the default mode (no options specified upon execution) allows you to select what configuration file you would like to see the contents of and also includes an organized output of the corresponding Quality of Protection (QoP) information.

If no options are specified, the default mode (no options specified) displays the collected information listed above. Additionally, it interactively prompts you to select a configuration file that you want to see a dump of. The output, minus the interactive prompts themselves, is also simultaneously redirected into a text file named `tdgssgetinfo.outputlog.txt`, located in the same directory in which the tool is run.

## Syntax

```
tdgssgetinfo -h
tdgssgetinfo [-b] [-s]
```

## tdgssgetinfo Options

Option	Description
-b	Specifies using the <code>.bin</code> file instead of the TDGSSCONFIG GDO. Operates as if the TDGSSCONFIG GDO does not exist. The <code>tdgssconfig.bin</code> or <code>tdgssconfig.bin.prebuilt</code> file is read from instead.
-s	Summary mode. Does not dump the contents of the selected configuration file or display its corresponding QoP information. All other aforementioned information remains in the output.
-h	Displays usage information. If specified with any other option, takes precedence and only the usage message will be displayed before the tool immediately exits.

## Examples

### Example: Summary Mode

```
tdgssgetinfo -s
```

```
=====
|
| tdgssgetinfo Diagnostic Tool (TDGSS 17.10.00.100) |
|
=====

Machine name: sdtnnnnn
OS Release: SUSE Linux Enterprise Server 12 SP3
Time executed: Thu Nov 14 09:50:23 2019

-----
| Available TDGSS Versions |
-----
17E.10.00.06 /opt/teradata/tdat/tdgss/17E.10.00.06/
17H.10.00.113 /opt/teradata/tdat/tdgss/17H.10.00.113/
17H.10.00.114 /opt/teradata/tdat/tdgss/17H.10.00.114/
--> 17I.10.00.104 /opt/teradata/tdat/tdgss/17I.10.00.104/
17I.10.00.105 /opt/teradata/tdat/tdgss/17I.10.00.105/
("-->" denotes the current active version)

-----
| Configuration Information |
-----
No output returned.

-----
| Available Mechanisms |
-----
OID: TD2 - 1.3.6.1.4.1.191.1.1012.1.1.9
-----
"DefaultMechanism" = true
"DelegateCredentials" = false
"MechanismEnabled" = true
"MechanismRank" = 20
"OutOfSequenceDetection" = true
"ReplayDetection" = true
"UseLdapConfig" = false
```

OID: KRB5 - 1.2.840.113554.1.2.2

```
-----
"DefaultMechanism" = false
"DelegateCredentials" = true
"MechanismEnabled" = true
"MechanismRank" = 40
"OutOfSequenceDetection" = true
"ReplayDetection" = true
"TeradataKeyTab" = /etc/teradata.keytab
"UseLdapConfig" = false
```

OID: SPNEGO - 1.3.6.1.5.5.2

```
-----
"DefaultMechanism" = false
"DelegateCredentials" = true
"MechanismEnabled" = true
"MechanismRank" = 65
"OutOfSequenceDetection" = true
"ReplayDetection" = true
```

OID: ldap - 1.3.6.1.4.1.191.1.1012.1.20

```
-----
"DefaultMechanism" = false
"DelegateCredentials" = false
"MechanismEnabled" = true
"MechanismRank" = 70
"OutOfSequenceDetection" = true
"ReplayDetection" = true
"UseLdapConfig" = false
```

OID: PROXY - 1.3.6.1.4.1.28698.4.302.1.2

```
-----
"DefaultMechanism" = false
"DelegateCredentials" = false
"MechanismEnabled" = true
"MechanismRank" = 70
"OutOfSequenceDetection" = true
"ReplayDetection" = true
```

OID: TDNEGO - 1.3.6.1.4.1.28698.4.302.1.3



```

-----
"DefaultMechanism" = false
"DelegateCredentials" = true
"MechanismEnabled" = true
"MechanismRank" = 10
"OutOfSequenceDetection" = true
"ReplayDetection" = true

```

```

* Default mechanism is TD2.

```

```

-----
| Available TDGSS Configuration Files |
-----

```

1. /opt/teradata/tdat/tdgss/17H.10.00.113/bin/tdgssconfig.bin
2. /opt/teradata/tdat/tdgss/17H.10.00.113/etc/tdgssconfig.bin
3. /opt/teradata/tdat/tdgss/17H.10.00.114/etc/tdgssconfig.bin
4. /opt/teradata/tdat/tdgss/17E.10.00.06/etc/tdgssconfig.bin.prebuilt
5. /opt/teradata/tdat/tdgss/17H.10.00.113/etc/tdgssconfig.bin.prebuilt
6. /opt/teradata/tdat/tdgss/17H.10.00.114/etc/tdgssconfig.bin.prebuilt
7. /opt/teradata/tdat/tdgss/17I.10.00.105/etc/tdgssconfig.bin.prebuilt
8. TDGSSCONFIG GDO

### Example: No Options Specified

When no options are specified, `tdgssgetinfo` displays all collected information and prompts you to select a configuration file that you would like to see a dump of.

```
tdgssgetinfo
```

```

=====
|                                     |
| tdgssgetinfo Diagnostic Tool (TDGSS 17.10.00.100) |
|                                     |
=====

```

```

Machine name: sdtntnnnn
OS Release: SUSE Linux Enterprise Server 12 SP3
Time executed: Thu Nov 14 09:50:23 2019

```

```

-----
| Available TDGSS Versions |
-----

```

```

17E.10.00.06 /opt/teradata/tdat/tdgss/17E.10.00.06/
17H.10.00.113 /opt/teradata/tdat/tdgss/17H.10.00.113/
17H.10.00.114 /opt/teradata/tdat/tdgss/17H.10.00.114/
--> 17I.10.00.104 /opt/teradata/tdat/tdgss/17I.10.00.104/
17I.10.00.105 /opt/teradata/tdat/tdgss/17I.10.00.105/

```

("-->" denotes the current active version)

```

-----
| Configuration Information |
-----

```

No output returned.

```

-----
| Available Mechanisms |
-----

```

OID: TD2 - 1.3.6.1.4.1.191.1.1012.1.1.9

```

-----
"DefaultMechanism" = true
"DelegateCredentials" = false
"MechanismEnabled" = true
"MechanismRank" = 20
"OutOfSequenceDetection" = true
"ReplayDetection" = true
"UseLdapConfig" = false

```

OID: KRB5 - 1.2.840.113554.1.2.2

```

-----
"DefaultMechanism" = false
"DelegateCredentials" = true
"MechanismEnabled" = true
"MechanismRank" = 40
"OutOfSequenceDetection" = true
"ReplayDetection" = true
"TeradataKeyTab" = /etc/teradata.keytab
"UseLdapConfig" = false

```

OID: SPNEGO - 1.3.6.1.5.5.2

```

-----
"DefaultMechanism" = false

```

```
"DelegateCredentials" = true
"MechanismEnabled" = true
"MechanismRank" = 65
"OutOfSequenceDetection" = true
"ReplayDetection" = true
```

```
OID: ldap - 1.3.6.1.4.1.191.1.1012.1.20
-----
```

```
"DefaultMechanism" = false
"DelegateCredentials" = false
"MechanismEnabled" = true
"MechanismRank" = 70
"OutOfSequenceDetection" = true
"ReplayDetection" = true
"UseLdapConfig" = false
```

```
OID: PROXY - 1.3.6.1.4.1.28698.4.302.1.2
-----
```

```
"DefaultMechanism" = false
"DelegateCredentials" = false
"MechanismEnabled" = true
"MechanismRank" = 70
"OutOfSequenceDetection" = true
"ReplayDetection" = true
```

```
OID: TDNEGO - 1.3.6.1.4.1.28698.4.302.1.3
-----
```

```
"DefaultMechanism" = false
"DelegateCredentials" = true
"MechanismEnabled" = true
"MechanismRank" = 10
"OutOfSequenceDetection" = true
"ReplayDetection" = true
```

```
* Default mechanism is TD2.
```

```
-----
| Available TDGSS Configuration Files |
-----
```

1. /opt/teradata/tdat/tdgss/17H.10.00.113/bin/tdgssconfig.bin
2. /opt/teradata/tdat/tdgss/17H.10.00.113/etc/tdgssconfig.bin
3. /opt/teradata/tdat/tdgss/17H.10.00.114/etc/tdgssconfig.bin
4. /opt/teradata/tdat/tdgss/17E.10.00.06/etc/tdgssconfig.bin.prebuilt
5. /opt/teradata/tdat/tdgss/17H.10.00.113/etc/tdgssconfig.bin.prebuilt
6. /opt/teradata/tdat/tdgss/17H.10.00.114/etc/tdgssconfig.bin.prebuilt
7. /opt/teradata/tdat/tdgss/17I.10.00.105/etc/tdgssconfig.bin.prebuilt
8. TDGSSCONFIG GDO

Select the configuration file (1-8): 8

```
-----
| Available QOPs |
-----
```

QOP: GLOBAL\_QOP\_0

```
-----
ConfidentialityAlgorithm = Blowfish
IntegrityAlgorithm = NONE
KeyExchangeAlgorithm = DH
KeyLength = K128
KeyLengthP = K2048
Mode = ECB
Padding = NoPadding
```

QOP: GLOBAL\_QOP\_1

```
-----
ConfidentialityAlgorithm = AES
IntegrityAlgorithm = SHA1
KeyExchangeAlgorithm = DH
KeyLength = K128
KeyLengthP = K2048
Mode = OFB
Padding = PKCS5Padding
```

QOP: AES-K128\_CBC\_PKCS5Padding\_SHA1\_DH-K2048

```
-----
ConfidentialityAlgorithm = AES
IntegrityAlgorithm = SHA1
KeyExchangeAlgorithm = DH
KeyLength = K128
KeyLengthP = K2048
```

```

Mode = CBC
Padding = PKCS5Padding

QOP: AES-K192_CBC_PKCS5Padding_SHA1_DH-K2048
-----
ConfidentialityAlgorithm = AES
IntegrityAlgorithm = SHA1
KeyExchangeAlgorithm = DH
KeyLength = K192
KeyLengthP = K2048
Mode = CBC
Padding = PKCS5Padding

QOP: AES-K256_CBC_PKCS5Padding_SHA1_DH-K2048
-----
ConfidentialityAlgorithm = AES
IntegrityAlgorithm = SHA1
KeyExchangeAlgorithm = DH
KeyLength = K256
KeyLengthP = K2048
Mode = CBC
Padding = PKCS5Padding

QOP: AES-K128_GCM_PKCS5Padding_SHA2_DH-K2048
-----
ConfidentialityAlgorithm = AES
IntegrityAlgorithm = SHA256
KeyExchangeAlgorithm = DH
KeyLength = K128
KeyLengthP = K2048
Mode = GCM
Padding = PKCS5Padding

QOP: AES-K192_GCM_PKCS5Padding_SHA2_DH-K2048
-----
ConfidentialityAlgorithm = AES
IntegrityAlgorithm = SHA256
KeyExchangeAlgorithm = DH
KeyLength = K192
KeyLengthP = K2048

```

```

Mode = GCM
Padding = PKCS5Padding

QOP: AES-K256_GCM_PKCS5Padding_SHA2_DH-K2048
-----
ConfidentialityAlgorithm = AES
IntegrityAlgorithm = SHA256
KeyExchangeAlgorithm = DH
KeyLength = K256
KeyLengthP = K2048
Mode = GCM
Padding = PKCS5Padding

QOP: AES-K128_CCM_PKCS5Padding_SHA2_DH-K2048
-----
ConfidentialityAlgorithm = AES
IntegrityAlgorithm = SHA256
KeyExchangeAlgorithm = DH
KeyLength = K128
KeyLengthP = K2048
Mode = CCM
Padding = PKCS5Padding

QOP: AES-K192_CCM_PKCS5Padding_SHA2_DH-K2048
-----
ConfidentialityAlgorithm = AES
IntegrityAlgorithm = SHA256
KeyExchangeAlgorithm = DH
KeyLength = K192
KeyLengthP = K2048
Mode = CCM
Padding = PKCS5Padding

QOP: AES-K256_CCM_PKCS5Padding_SHA2_DH-K2048
-----
ConfidentialityAlgorithm = AES
IntegrityAlgorithm = SHA256
KeyExchangeAlgorithm = DH
KeyLength = K256
KeyLengthP = K2048

```

```

Mode = CCM
Padding = PKCS5Padding

QOP: AES-K128_CTR_PKCS5Padding_SHA2_DH-K2048
-----
ConfidentialityAlgorithm = AES
IntegrityAlgorithm = SHA256
KeyExchangeAlgorithm = DH
KeyLength = K128
KeyLengthP = K2048
Mode = CTR
Padding = PKCS5Padding

QOP: AES-K192_CTR_PKCS5Padding_SHA2_DH-K2048
-----
ConfidentialityAlgorithm = AES
IntegrityAlgorithm = SHA256
KeyExchangeAlgorithm = DH
KeyLength = K192
KeyLengthP = K2048
Mode = CTR
Padding = PKCS5Padding

QOP: AES-K256_CTR_PKCS5Padding_SHA2_DH-K2048
-----
ConfidentialityAlgorithm = AES
IntegrityAlgorithm = SHA256
KeyExchangeAlgorithm = DH
KeyLength = K256
KeyLengthP = K2048
Mode = CTR
Padding = PKCS5Padding

-----
| Configuration File Dump |
-----

Dumping TDGSSCONFIG GDO...
Header: Version 2
79 Elements at offset 2c (44)

```

671 Attributes at offset b6c (2924)

19 Data items at offset 206c (8300)

Level 00: < TdgssConfigFile > (Element 0)

Level 01: < Header > (Element 1)

ATTR: "ConfigFileType" = "User"

ATTR: "Version" = "1"

DATA: ""

Level 01: < Mechanisms > (Element 2)

Level 02: < Mechanism > (Element 6)

ATTR: "InterfaceType" = "teradata"

ATTR: "LibraryName" = "gssp2td2"

ATTR: "Name" = "TD2"

ATTR: "ObjectId" = "1.3.6.1.4.1.191.1.1012.1.1.9"

ATTR: "Prefix" = "TD2"

Level 03: < MechanismProperties > (Element 14)

ATTR: "AnonymousAuthentication" = "no"

ATTR: "AuthenticationSupported" = "no"

ATTR: "AuthorizationSupported" = "no"

ATTR: "ConfidentialityDesired" = "yes"

ATTR: "CredentialIsUPN" = "yes"

ATTR: "CredentialUsage" = "0"

ATTR: "DHKeyG" =

[illegible][illegible][illegible]

ATTR: "DHKeyG2048" =

[illegible][illegible][illegible][illegible][illegible][illegible][illegible][illegible]

ATTR: "DHKeyP" =



```

E4BE0A78F54C4A0B17E7E9249A78BCC08868C17281D8463C880937853E73DDC7
87E41580A8AFE2594D984C9E0814C590790354ECCD1BE8EA85961E5E0974B32E
FE178335F061E80189B4BDAA20F67B47
ATTR: "DHKeyP2048" =
8AB3F86E8D374B782F31DAD5F27D6AFDA30150C11A20CF6346712AE2D2C6B70A
5B79D45D4C0C232A065B207B121B2C33E147B5983C38A1087F272703B0B839CB
A6F71C5D0EB51EC890934EACF2C7DD2A1DF6F55E89B145A0359D35EF8FB6C561
E157B13FF927A35E69963648614902B1034EF71197F545DEF3236244EADAE068
9E624CF1245953630AE042BD797C4025E37C51D9F6CBDA0B2278FA7D5CA2D9CA
930BE2968330C811A4BA4D0845333C0D62E3EE742154F6B62F2951CD8C73C43B
5AA1C7819DEF1D7C9314411E465F8E4796666594AADE0AEB3F1256E5719E7AE5
4DD34FFDA949634E4A293C5BC60AF258BB9FE558086E83B3DD3D7491966DEE93
ATTR: "DefaultMechanism" = "yes"
ATTR: "DefaultNegotiatingMechanism" = "no"
ATTR: "DelegateCredentials" = "no"
ATTR: "DesiredContextTime" = ""
ATTR: "DesiredCredentialTime" = ""
ATTR: "IntegrityDesired" = "yes"
ATTR: "MechanismEnabled" = "yes"
ATTR: "MechanismIgnoresQop" = "no"
ATTR: "MechanismRank" = "20"
ATTR: "MutualAuthentication" = "yes"
ATTR: "NegotiationSupported" = "no"
ATTR: "OutOfSequenceDetection" = "yes"
ATTR: "ReplayDetection" = "yes"
ATTR: "SingleSignOnSupported" = "no"
ATTR: "UseLdapConfig" = "no"
ATTR: "VerifyDHKey" = "no"

```

Level 03: < MechQop > (Element 15)

```

ATTR: "Value" = "0"
DATA: "GLOBAL_QOP_1"

```

Level 03: < MechQop > (Element 16)

```

ATTR: "Value" = "Default"
DATA: "AES-K128_GCM_PKCS5Padding_SHA2_DH-K2048 AES-K128_CBC_PKCS5Paddin
g_SHA1_DH-K2048 AES-K192_GCM_PKCS5Padding_SHA2_DH-K2048 AES-K192
_CBC_PKCS5Padding_SHA1_DH-K2048 AES-K256_GCM_PKCS5Padding_SHA2_D
H-K2048 AES-K256_CBC_PKCS5Padding_SHA1_DH-K2048"

```

Level 02: < Mechanism > (Element 7)

```

ATTR: "InterfaceType" = "gss"
ATTR: "LibraryName" = "gsssp2gss"
ATTR: "Name" = "KRB5"
ATTR: "ObjectId" = "1.2.840.113554.1.2.2"
ATTR: "Prefix" = "gsssp2gss"

```

Level 03: < MechanismProperties > (Element 17)

```

ATTR: "AnonymousAuthentication" = "no"
ATTR: "AuthenticationSupported" = "yes"
ATTR: "AuthorizationSupported" = "yes"
ATTR: "ConfidentialityDesired" = "yes"
ATTR: "CredentialIsUPN" = "yes"
ATTR: "CredentialUsage" = "0"
ATTR: "DefaultMechanism" = "no"
ATTR: "DefaultNegotiatingMechanism" = "no"
ATTR: "DelegateCredentials" = "yes"
ATTR: "DesiredContextTime" = ""
ATTR: "DesiredCredentialTime" = ""
ATTR: "GenerateCredentialFromLogon" = "yes"
ATTR: "IntegrityDesired" = "yes"
ATTR: "LdapBaseFQDN" = "dc=esrootdom,dc=esdev,dc=tdat"
ATTR: "LdapClientDebug" = "0"
ATTR: "LdapClientDeref" = "never"
ATTR: "LdapClientMechanism" = "SASL/DIGEST-MD5"
ATTR: "LdapClientRandomDevice" = "/dev/urandom"
ATTR: "LdapClientRebindAuth" = "yes"
ATTR: "LdapClientReferrals" = "off"
ATTR: "LdapClientSaslSecProps" = ""
ATTR: "LdapClientUseTls" = "no"
ATTR: "LdapGroupBaseFQDN" = ""
ATTR: "LdapServerName" = "ldaps://esroot/"
ATTR: "LdapServerPort" = "0"
ATTR: "LdapServerRealm" = "esroot.esrootdom.esdev.tdat"
ATTR: "LdapServiceFQDN"
= "cn=drct01,ou=people,ou=testing,dc=esrootdom,dc=esdev,dc=tdat"
ATTR: "LdapServicePassword" = "Ad0+xzytkxtG5tByCfGT+qg="
ATTR: "LdapServicePasswordProtected" = "yes"
ATTR: "LdapSystemFQDN"
= "cn=end2end,cn=tdat,ou=testing,dc=esrootdom,dc=esdev,dc=tdat"
ATTR: "LdapUserBaseFQDN" = ""
ATTR: "MechanismEnabled" = "yes"
ATTR: "MechanismIgnoresQop" = "yes"
ATTR: "MechanismRank" = "40"

```

```

ATTR: "MutualAuthentication" = "yes"
ATTR: "NegotiationSupported" = "no"
ATTR: "OutOfSequenceDetection" = "yes"
ATTR: "ReplayDetection" = "yes"
ATTR: "SingleSignOnSupported" = "yes"
ATTR: "TeradataKeyTab" = "/etc/teradata.keytab"
ATTR: "UseLdapConfig" = "no"
ATTR: "LdapServiceBindRequired" = "yes"

```

Level 03: < IdentitySearch > (Element 18)

```

ATTR: "Base" = "ou=testing,dc=esrootdom,dc=esdev,dc=tdat"
ATTR: "BindName" = "${result}"
ATTR: "DatabaseName" = "${0}"
ATTR: "Filter" = "(userPrincipalName=${1}@esrootdom.esdev.tdat)"
ATTR: "Match" = "(.*)"
ATTR: "Scope" = "subtree"

```

Level 03: < RequiredLibrary > (Element 19)

```
ATTR: "Path" = "/usr/lib64/libgssapi_krb5.so"
```

Level 03: < RequiredLibrary > (Element 20)

```
ATTR: "Path" = "/usr/lib64/libgssapi_krb5.so.2"
```

Level 03: < RequiredLibrary > (Element 21)

```
ATTR: "Path" = "/lib64/libgssapi_krb5.so.2"
```

Level 03: < RequiredLibrary > (Element 22)

```
ATTR: "Path" = "/usr/lib/libgssapi_krb5.so"
```

Level 03: < RequiredLibrary > (Element 23)

```
ATTR: "Path" = "/usr/lib/libgssapi_krb5.so.2"
```

Level 03: < RequiredLibrary > (Element 24)

```
ATTR: "Path" = "/lib/libgssapi_krb5.so.2"
```

Level 03: < MechQop > (Element 25)

```
ATTR: "Value" = "0"  
DATA: "GLOBAL_QOP_0"
```

Level 02: < Mechanism > (Element 8)

```
ATTR: "InterfaceType" = "custom"
ATTR: "LibraryName" = "gssp2ldap"
ATTR: "Name" = "ldap"
ATTR: "ObjectId" = "1.3.6.1.4.1.191.1.1012.1.20"
ATTR: "Prefix" = "ldapv3"
```

Level 03: < MechanismProperties > (Element 26)

```
ATTR: "AnonymousAuthentication" = "no"
ATTR: "AuthenticationSupported" = "yes"
ATTR: "AuthorizationSupported" = "yes"
ATTR: "ConfidentialityDesired" = "yes"
ATTR: "CredentialIsUPN" = "yes"
ATTR: "CredentialUsage" = "0"
```

ATTR: "DHKeyG" =

[illegible]

ATTR: "DHKeyG2048" =

[illegible]

ATTR: "DHKeyP" =

E4BE0A78F54C4A0B17E7E9249A78BCC08868C17281D8463C880937853E73DDC7  
87E41580A8AFE2594D984C9E0814C590790354ECCD1BE8EA85961E5E0974B32E  
FE178335F061E80189B4BDAA20F67B47

ATTR: "DHKeyP2048" =

8AB3F86E8D374B782F31DAD5F27D6AFDA30150C11A20CF6346712AE2D2C6B70A  
5B79D45D4C0C232A065B207B121B2C33E147B5983C38A1087F272703B0B839CB  
A6F71C5D0EB51EC890934EACF2C7DD2A1DF6F55E89B145A0359D35EF8FB6C561  
E157B13FF927A35E69963648614902B1034EF71197F545DEF3236244EADAE068  
9E624CF1245953630AE042BD797C4025E37C51D9F6CBDA0B2278FA7D5CA2D9CA  
930BE2968330C811A4BA4D0845333C0D62E3EE742154F6B62F2951CD8C73C43B  
5AA1C7819DEF1D7C9314411E465F8E4796666594AADE0AEB3F1256E5719E7AE5  
4DD34FFDA949634E4A293C5BC60AF258BB9FE558086E83B3DD3D7491966DEF93

```

ATTR: "DefaultMechanism" = "no"
ATTR: "DefaultNegotiatingMechanism" = "no"
ATTR: "DelegateCredentials" = "no"
ATTR: "DesiredContextTime" = ""
ATTR: "DesiredCredentialTime" = ""
ATTR: "GenerateCredentialFromLogon" = "yes"
ATTR: "IntegrityDesired" = "yes"
ATTR: "LdapAllowUnsafeServerConnect" = "yes"
ATTR: "LdapBaseFQDN" = "dc=esrootdom,dc=esdev,dc=tdat"
ATTR: "LdapClientDebug" = "0"
ATTR: "LdapClientDeref" = "never"
ATTR: "LdapClientMechanism" = "simple"
ATTR: "LdapClientRandomDevice" = "/dev/urandom"
ATTR: "LdapClientRebindAuth" = "yes"
ATTR: "LdapClientReferrals" = "off"
ATTR: "LdapClientSaslSecProps" = ""
ATTR: "LdapClientTlsCACert" = ""
ATTR: "LdapClientTlsCACertDir" = ""
ATTR: "LdapClientTlsCRLCheck" = "none"
ATTR: "LdapClientTlsCert" = ""
ATTR: "LdapClientTlsCipherSuite" = ""
ATTR: "LdapClientTlsKey" = ""
ATTR: "LdapClientTlsRandFile" = ""
ATTR: "LdapClientTlsReqCert" = "never"
ATTR: "LdapClientUseTls" = "no"
ATTR: "LdapGroupBaseFQDN" = ""
ATTR: "LdapServerName" = "ldaps://esroot/"
ATTR: "LdapServerPort" = "0"
ATTR: "LdapServerRealm" = "esroot.esrootdom.esdev.tdat"
ATTR: "LdapServiceBindRequired" = "yes"
ATTR: "LdapServiceFQDN"
= "cn=drct01,ou=people,ou=testing,dc=esrootdom,dc=esdev,dc=tdat"
ATTR: "LdapServicePassword" = "Ad0+xzytkxtG5tByCfgT+qg="
ATTR: "LdapServicePasswordProtected" = "yes"
ATTR: "LdapSystemFQDN"
= "cn=end2end,cn=tdat,ou=testing,dc=esrootdom,dc=esdev,dc=tdat"
ATTR: "LdapUserBaseFQDN" = ""
ATTR: "MechanismEnabled" = "yes"
ATTR: "MechanismIgnoresQop" = "no"
ATTR: "MechanismRank" = "70"
ATTR: "MutualAuthentication" = "yes"
ATTR: "NegotiationSupported" = "no"
ATTR: "OutOfSequenceDetection" = "yes"
ATTR: "ReplayDetection" = "yes"

```

[illegible]

[illegible]

ATTR: "DHKeyP2048" =  
8AB3F86E8D374B782F31DAD5F27D6AFDA30150C11A20CF6346712AE2D2C6B70A  
5B79D45D4C0C232A065B207B121B2C33E147B5983C38A1087F272703B0B839CB  
A6F71C5D0EB51EC890934EACF2C7DD2A1DF6F55E89B145A0359D35EF8FB6C561  
E157B13FF927A35E69963648614902B1034EF71197F545DEF3236244EADAE068  
9E624CF1245953630AE042BD797C4025E37C51D9F6CBDA0B2278FA7D5CA2D9CA  
930BE2968330C811A4BA4D0845333C0D62E3EE742154F6B62F2951CD8C73C43B  
5AA1C7819DEF1D7C9314411E465F8E479666594AADE0AEB3F1256E5719E7AE5  
4DD34FFDA949634E4A293C5BC60AF258BB9FE558086E83B3DD3D7491966DEE93

```
ATTR: "DefaultMechanism" = "no"
ATTR: "DefaultNegotiatingMechanism" = "no"
ATTR: "DelegateCredentials" = "no"
ATTR: "DesiredContextTime" = ""
ATTR: "DesiredCredentialTime" = ""
ATTR: "GenerateCredentialFromLogon" = "yes"
ATTR: "IntegrityDesired" = "yes"
ATTR: "JWTDecryptionKeyFile" = ""
ATTR: "JWTSkewTime" = "300"
ATTR: "JWTVerificationKeyFile" = ""
ATTR: "MechanismEnabled" = "no"
ATTR: "MechanismIgnoresQop" = "no"
ATTR: "MechanismRank" = "30"
ATTR: "MutualAuthentication" = "no"
ATTR: "NegotiationSupported" = "yes"
ATTR: "OutOfSequenceDetection" = "yes"
ATTR: "ReplayDetection" = "yes"
ATTR: "SingleSignOnSupported" = "yes"
ATTR: "UseLdapConfig" = "no"
```

Level 03: &lt; MechOp &gt; (Element 31)

```
ATTR: "Value" = "Default"
DATA: "AES-K128_GCM_PKCS5Padding_SHA2_DH-K2048 AES-K192_GCM_PKCS5Padding_SHA2_DH-K2048 AES-K256_GCM_PKCS5Padding_SHA2_DH-K2048 AES-K128_CBC_PKCS5Padding_SHA1_DH-K2048 AES-K192_CBC_PKCS5Padding_SHA1_DH-K2048 AES-K256_CBC_PKCS5Padding_SHA1_DH-K2048"
```

Level 02: < Mechanism > (Element 10)

ATTR: "InterfaceType" = "teradata"  
 ATTR: "LibraryName" = "gssp2td1"  
 ATTR: "Name" = "TD1"  
 ATTR: "ObjectId" = "1.3.6.1.4.1.191.1.1012.1.1.8"  
 ATTR: "Prefix" = "TD1"

Level 03: < MechanismProperties > (Element 32)

ATTR: "AnonymousAuthentication" = "no"  
 ATTR: "AuthenticationSupported" = "no"  
 ATTR: "AuthorizationSupported" = "no"  
 ATTR: "ConfidentialityDesired" = "yes"  
 ATTR: "CredentialIsUPN" = "yes"  
 ATTR: "CredentialUsage" = "0"  
 ATTR: "DefaultMechanism" = "no"  
 ATTR: "DefaultNegotiatingMechanism" = "no"  
 ATTR: "DelegateCredentials" = "no"  
 ATTR: "DesiredContextTime" = ""  
 ATTR: "DesiredCredentialTime" = ""  
 ATTR: "IntegrityDesired" = "yes"  
 ATTR: "MechanismEnabled" = "no"  
 ATTR: "MechanismIgnoresQop" = "yes"  
 ATTR: "MechanismRank" = "10"  
 ATTR: "MutualAuthentication" = "yes"  
 ATTR: "NegotiationSupported" = "no"  
 ATTR: "OutOfSequenceDetection" = "yes"  
 ATTR: "ReplayDetection" = "yes"  
 ATTR: "SingleSignOnSupported" = "no"  
 ATTR: "UseLdapConfig" = "no"

Level 03: < MechQop > (Element 33)

ATTR: "Value" = "0"  
 DATA: "GLOBAL\_QOP\_0"

Level 02: < Mechanism > (Element 11)

ATTR: "InterfaceType" = "negotiate"  
 ATTR: "LibraryName" = "gssp2spnego"  
 ATTR: "Name" = "SPNEGO"  
 ATTR: "ObjectId" = "1.3.6.1.5.5.2"



ATTR: "Prefix" = "spnego"

Level 03: < MechanismProperties > (Element 34)

ATTR: "AnonymousAuthentication" = "no"  
 ATTR: "AuthenticationSupported" = "yes"  
 ATTR: "AuthorizationSupported" = "no"  
 ATTR: "ConfidentialityDesired" = "yes"  
 ATTR: "CredentialIsUPN" = "yes"  
 ATTR: "CredentialUsage" = "0"  
 ATTR: "DefaultMechanism" = "no"  
 ATTR: "DefaultNegotiatingMechanism" = "no"  
 ATTR: "DelegateCredentials" = "yes"  
 ATTR: "DesiredContextTime" = ""  
 ATTR: "DesiredCredentialTime" = ""  
 ATTR: "IntegrityDesired" = "yes"  
 ATTR: "LdapBaseFQDN" = ""  
 ATTR: "LdapClientDebug" = "0"  
 ATTR: "LdapClientDeref" = "never"  
 ATTR: "LdapClientMechanism" = "SASL/DIGEST-MD5"  
 ATTR: "LdapClientRandomDevice" = "/dev/urandom"  
 ATTR: "LdapClientRebindAuth" = "yes"  
 ATTR: "LdapClientReferrals" = "off"  
 ATTR: "LdapClientSaslSecProps" = ""  
 ATTR: "LdapClientUseTls" = "no"  
 ATTR: "LdapGroupBaseFQDN" = ""  
 ATTR: "LdapServerName" = ""  
 ATTR: "LdapServerPort" = "389"  
 ATTR: "LdapServerRealm" = ""  
 ATTR: "LdapServiceFQDN" = ""  
 ATTR: "LdapServicePassword" = ""  
 ATTR: "LdapServicePasswordProtected" = "no"  
 ATTR: "LdapSystemFQDN" = ""  
 ATTR: "LdapUserBaseFQDN" = ""  
 ATTR: "MechanismEnabled" = "yes"  
 ATTR: "MechanismRank" = "65"  
 ATTR: "MutualAuthentication" = "yes"  
 ATTR: "NegotiationSupported" = "no"  
 ATTR: "OutOfSequenceDetection" = "yes"  
 ATTR: "ReplayDetection" = "yes"  
 ATTR: "SingleSignOnSupported" = "yes"  
 ATTR: "UseLdapConfig" = "no"

Level 03: < MechQop > (Element 35)

```
ATTR: "Value" = "0"
DATA: "GLOBAL_QOP_1"
```

Level 03: < NegotiatedMechanism > (Element 36)

```
ATTR: "Enable" = "yes"
ATTR: "ObjectId" = "1.2.840.113554.1.2.2"
```

Level 02: < Mechanism > (Element 12)

```
ATTR: "InterfaceType" = "custom"
ATTR: "LibraryName" = "gssp2proxy"
ATTR: "Name" = "PROXY"
ATTR: "ObjectId" = "1.3.6.1.4.1.28698.4.302.1.2"
ATTR: "Prefix" = "Proxy"
```

Level 03: < MechanismProperties > (Element 37)

```
ATTR: "AnonymousAuthentication" = "no"  
ATTR: "AuthenticationSupported" = "yes"  
ATTR: "AuthorizationSupported" = "no"  
ATTR: "CACertDir" = ""  
ATTR: "CACertFile" = ""  
ATTR: "CertificateFile" = ""  
ATTR: "ConfidentialityDesired" = "yes"  
ATTR: "CredentialIsUPN" = "yes"  
ATTR: "CredentialUsage" = "0"  
ATTR: "DHKeyG" =  
00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000  
00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000  
00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000  
ATTR: "DHKeyG2048" =  
00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000  
00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000  
00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000  
00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000  
00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000  
00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000  
00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000  
00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000  
ATTR: "DHKeyP" =  
E4BE0A78F54C4A0B17E7E9249A78BCC08868C17281D8463C880937853E73DDC7
```

```
87E41580A8AFE2594D984C9E0814C590790354ECCD1BE8EA85961E5E0974B32E
FE178335F061E80189B4BDAA20F67B47
```

```
ATTR: "DHKeyP2048" =
```

```
8AB3F86E8D374B782F31DAD5F27D6AFDA30150C11A20CF6346712AE2D2C6B70A
5B79D45D4C0C232A065B207B121B2C33E147B5983C38A1087F272703B0B839CB
A6F71C5D0EB51EC890934EACF2C7DD2A1DF6F55E89B145A0359D35EF8FB6C561
E157B13FF927A35E69963648614902B1034EF71197F545DEF3236244EADAE068
9E624CF1245953630AE042BD797C4025E37C51D9F6CBDA0B2278FA7D5CA2D9CA
930BE2968330C811A4BA4D0845333C0D62E3EE742154F6B62F2951CD8C73C43B
5AA1C7819DEF1D7C9314411E465F8E4796666594AADE0AEB3F1256E5719E7AE5
4DD34FFDA949634E4A293C5BC60AF258BB9FE558086E83B3DD3D7491966DEE93
```

```
ATTR: "DefaultMechanism" = "no"
```

```
ATTR: "DefaultNegotiatingMechanism" = "no"
```

```
ATTR: "DelegateCredentials" = "no"
```

```
ATTR: "DesiredContextTime" = ""
```

```
ATTR: "DesiredCredentialTime" = ""
```

```
ATTR: "GenerateCredentialFromLogon" = "yes"
```

```
ATTR: "IntegrityDesired" = "yes"
```

```
ATTR: "MechanismEnabled" = "yes"
```

```
ATTR: "MechanismIgnoresQop" = "no"
```

```
ATTR: "MechanismRank" = "70"
```

```
ATTR: "MutualAuthentication" = "yes"
```

```
ATTR: "NegotiationSupported" = "no"
```

```
ATTR: "OutOfSequenceDetection" = "yes"
```

```
ATTR: "PrivateKeyFile" = ""
```

```
ATTR: "PrivateKeyPassword" = ""
```

```
ATTR: "PrivateKeyPasswordProtected" = "no"
```

```
ATTR: "ProxySupported" = "yes"
```

```
ATTR: "ReplayDetection" = "yes"
```

```
ATTR: "SigningHashAlgorithm" = "SHA256"
```

```
ATTR: "SingleSignOnSupported" = "no"
```

```
ATTR: "UseLdapConfig" = "no"
```

```
Level 03: < MechQop > (Element 38)
```

```
ATTR: "Value" = "0"
```

```
DATA: "GLOBAL_QOP_1"
```

```
Level 03: < MechQop > (Element 39)
```

```
ATTR: "Value" = "Default"
```

```
DATA: "AES-K128_GCM_PKCS5Padding_SHA2_DH-K2048 AES-K128_CBC_PKCS5Paddin
g_SHA1_DH-K2048 AES-K192_GCM_PKCS5Padding_SHA2_DH-K2048 AES-K192
_CBC_PKCS5Padding_SHA1_DH-K2048 AES-K256_GCM_PKCS5Padding_SHA2_D
```

H-K2048 AES-K256\_CBC\_PKCS5Padding\_SHA1\_DH-K2048"

Level 02: < Mechanism > (Element 13)

ATTR: "InterfaceType" = "negotiate"  
 ATTR: "LibraryName" = "gsssp2tdnego"  
 ATTR: "Name" = "TDNEGO"  
 ATTR: "ObjectId" = "1.3.6.1.4.1.28698.4.302.1.3"  
 ATTR: "Prefix" = "TDNEGO"

Level 03: < MechanismProperties > (Element 40)

ATTR: "AuthenticationSupported" = "yes"  
 ATTR: "AuthorizationSupported" = "yes"  
 ATTR: "ConfidentialityDesired" = "yes"  
 ATTR: "CredentialIsUPN" = "yes"  
 ATTR: "DefaultMechanism" = "no"  
 ATTR: "DefaultNegotiatingMechanism" = "no"  
 ATTR: "DelegateCredentials" = "yes"  
 ATTR: "GenerateCredentialFromLogon" = "yes"  
 ATTR: "IntegrityDesired" = "yes"  
 ATTR: "MechanismEnabled" = "yes"  
 ATTR: "MechanismRank" = "10"  
 ATTR: "MutualAuthentication" = "yes"  
 ATTR: "NegotiationSupported" = "yes"  
 ATTR: "OutOfSequenceDetection" = "yes"  
 ATTR: "ReplayDetection" = "yes"  
 ATTR: "SingleSignOnSupported" = "yes"  
 ATTR: "UseLdapConfig" = "no"

Level 03: < NegotiatedMechanism > (Element 41)

ATTR: "Enable" = "yes"  
 ATTR: "ObjectId" = "1.2.840.113554.1.2.2"

Level 03: < NegotiatedMechanism > (Element 42)

ATTR: "Enable" = "yes"  
 ATTR: "ObjectId" = "1.3.6.1.5.5.2"

Level 03: < NegotiatedMechanism > (Element 43)

ATTR: "Enable" = "yes"  
 ATTR: "ObjectId" = "1.3.6.1.4.1.28698.4.302.1.4"

Level 03: < NegotiatedMechanism > (Element 44)

ATTR: "Enable" = "yes"

ATTR: "ObjectId" = "1.3.6.1.4.1.191.1.1012.1.20"

Level 03: < NegotiatedMechanism > (Element 45)

ATTR: "Enable" = "yes"

ATTR: "ObjectId" = "1.3.6.1.4.1.191.1.1012.1.1.9"

Level 01: < LdapConfig > (Element 3)

Level 02: < Tls > (Element 46)

ATTR: "LdapClientTlsCACert" = ""

ATTR: "LdapClientTlsCACertDir" = ""

ATTR: "LdapClientTlsCRLCheck" = "none"

ATTR: "LdapClientTlsCert" = ""

ATTR: "LdapClientTlsCipherSuite" = ""

ATTR: "LdapClientTlsKey" = ""

ATTR: "LdapClientTlsRandFile" = ""

ATTR: "LdapClientTlsReqCert" = "never"

Level 02: < Services > (Element 47)

Level 03: < Service > (Element 49)

ATTR: "Id" = "esrootdom"

ATTR: "LdapAllowUnsafeServerConnect" = "yes"

ATTR: "LdapBaseFQDN" = "dc=esrootdom,dc=esdev,dc=tdat"

ATTR: "LdapClientDebug" = "0"

ATTR: "LdapClientDeref" = "never"

ATTR: "LdapClientMechanism" = "simple"

ATTR: "LdapClientRandomDevice" = "/dev/urandom"

ATTR: "LdapClientRebindAuth" = "yes"

ATTR: "LdapClientReferrals" = "off"

ATTR: "LdapClientSaslSecProps" = ""

ATTR: "LdapClientUseTls" = "no"

ATTR: "LdapGroupBaseFQDN" = ""

ATTR: "LdapServerName" = "ldap://esroot.esrootdom.esdev.tdat:389/"

ATTR: "LdapServerPort" = "389"

ATTR: "LdapServerRealm" = ""

ATTR: "LdapServiceBindRequired" = "no"

```

ATTR: "LdapServiceFQDN"
= "cn=drct01,ou=people,ou=testing,dc=esrootdom,dc=esdev,dc=tdat"
ATTR: "LdapServicePassword" = "secret"
ATTR: "LdapServicePasswordProtected" = "no"
ATTR: "LdapSystemFQDN"
= "cn=end2end,cn=tdat,ou=testing,dc=esrootdom,dc=esdev,dc=tdat"
ATTR: "LdapUserBaseFQDN" = ""

```

Level 03: < Service > (Element 50)

```

ATTR: "Id" = "tdgs23"
ATTR: "LdapAllowUnsafeServerConnect" = "yes"
ATTR: "LdapBaseFQDN" = "dc=esrootdom,dc=esdev,dc=tdat"
ATTR: "LdapClientDebug" = "0"
ATTR: "LdapClientDeref" = "never"
ATTR: "LdapClientMechanism" = "simple"
ATTR: "LdapClientRandomDevice" = "/dev/urandom"
ATTR: "LdapClientRebindAuth" = "yes"
ATTR: "LdapClientReferrals" = "off"
ATTR: "LdapClientSaslSecProps" = ""
ATTR: "LdapClientUseTls" = "no"
ATTR: "LdapGroupBaseFQDN" = "dc=esrootdom,dc=esdev,dc=tdat"
ATTR: "LdapServerName" = "ldap://tdgss.esrootdom.esdev.tdat:389/"
ATTR: "LdapServerPort" = "389"
ATTR: "LdapServerRealm" = ""
ATTR: "LdapServiceBindRequired" = "no"
ATTR: "LdapServiceFQDN"
= "cn=drct01,ou=people,ou=testing,dc=esrootdom,dc=esdev,dc=tdat"
ATTR: "LdapServicePassword" = "secret"
ATTR: "LdapServicePasswordProtected" = "no"
ATTR: "LdapSystemFQDN"
= "cn=end2end,cn=tdat,ou=testing,dc=esrootdom,dc=esdev,dc=tdat"
ATTR: "LdapUserBaseFQDN" = "dc=esrootdom,dc=esdev,dc=tdat"

```

Level 03: < Service > (Element 51)

```

ATTR: "Id" = "s3"
ATTR: "LdapAllowUnsafeServerConnect" = "yes"
ATTR: "LdapBaseFQDN" = "dc=esrootdom,dc=esdev,dc=tdat"
ATTR: "LdapClientDebug" = "0"
ATTR: "LdapClientDeref" = "never"
ATTR: "LdapClientMechanism" = "simple"
ATTR: "LdapClientRandomDevice" = "/dev/urandom"
ATTR: "LdapClientRebindAuth" = "yes"

```

```

ATTR: "LdapClientReferrals" = "off"
ATTR: "LdapClientSaslSecProps" = ""
ATTR: "LdapClientUseTls" = "no"
ATTR: "LdapGroupBaseFQDN" = "dc=esrootdom,dc=esdev,dc=tdat"
ATTR: "LdapServerName" = "ldap://esroot.esrootdom.esdev.tdat:389/"
ATTR: "LdapServerPort" = "389"
ATTR: "LdapServerRealm" = ""
ATTR: "LdapServiceBindRequired" = "no"
ATTR: "LdapServiceFQDN"
= "cn=drct01,ou=people,ou=testing,dc=esrootdom,dc=esdev,dc=tdat"
ATTR: "LdapServicePassword" = "secret"
ATTR: "LdapServicePasswordProtected" = "no"
ATTR: "LdapSystemFQDN"
= "cn=end2end,cn=tdat,ou=testing,dc=esrootdom,dc=esdev,dc=tdat"
ATTR: "LdapUserBaseFQDN" = "dc=esrootdom,dc=esdev,dc=tdat"

```

Level 03: < Service > (Element 52)

```

ATTR: "Id" = "s4"
ATTR: "LdapAllowUnsafeServerConnect" = "yes"
ATTR: "LdapBaseFQDN" = "dc=esrootdom,dc=esdev,dc=tdat"
ATTR: "LdapClientDebug" = "0"
ATTR: "LdapClientDeref" = "never"
ATTR: "LdapClientMechanism" = "simple"
ATTR: "LdapClientRandomDevice" = "/dev/urandom"
ATTR: "LdapClientRebindAuth" = "yes"
ATTR: "LdapClientReferrals" = "off"
ATTR: "LdapClientSaslSecProps" = ""
ATTR: "LdapClientUseTls" = "no"
ATTR: "LdapGroupBaseFQDN" = "dc=esrootdom,dc=esdev,dc=tdat"
ATTR: "LdapServerName" = "ldap://tdgss.esrootdom.esdev.tdat:389/"
ATTR: "LdapServerPort" = "389"
ATTR: "LdapServerRealm" = ""
ATTR: "LdapServiceBindRequired" = "no"
ATTR: "LdapServiceFQDN"
= "cn=drct01,ou=people,ou=testing,dc=esrootdom,dc=esdev,dc=tdat"
ATTR: "LdapServicePassword" = "secret"
ATTR: "LdapServicePasswordProtected" = "no"
ATTR: "LdapSystemFQDN"
= "cn=end2end,cn=tdat,ou=testing,dc=esrootdom,dc=esdev,dc=tdat"
ATTR: "LdapUserBaseFQDN" = "dc=esrootdom,dc=esdev,dc=tdat"

```

Level 02: < Canonicalizations > (Element 48)

Level 03: < IdentitySearch > (Element 53)

ATTR: "Base" = "ou=testing,dc=esrootdom,dc=esdev,dc=tdat"  
 ATTR: "BindName" = "\${result}"  
 ATTR: "DatabaseName" = "\${0}"  
 ATTR: "Filter" = "(userPrincipalName=\${1}@esrootdom.esdev.tdat)"  
 ATTR: "Match" = "td-(.+) "  
 ATTR: "Ref" = "tdgs23"  
 ATTR: "Scope" = "subtree"

Level 03: < IdentitySearch > (Element 54)

ATTR: "Base" = "ou=testing,dc=esrootdom,dc=esdev,dc=tdat"  
 ATTR: "BindName" = "\${result}"  
 ATTR: "DatabaseName" = "\${0}"  
 ATTR: "Filter" = "(userPrincipalName=\${1}@esrootdom.esdev.tdat)"  
 ATTR: "Match" = "sd-(.+) "  
 ATTR: "Ref" = "s3"  
 ATTR: "Scope" = "subtree"

Level 03: < IdentitySearch > (Element 55)

ATTR: "Base" = "ou=testing,dc=esrootdom,dc=esdev,dc=tdat"  
 ATTR: "BindName" = "\${result}"  
 ATTR: "DatabaseName" = "\${0}"  
 ATTR: "Filter" = "(userPrincipalName=\${1}@esrootdom.esdev.tdat)"  
 ATTR: "Match" = "es-(.+) "  
 ATTR: "Ref" = "s4"  
 ATTR: "Scope" = "subtree"

Level 03: < IdentitySearch > (Element 56)

ATTR: "Base" = "ou=testing,dc=esrootdom,dc=esdev,dc=tdat"  
 ATTR: "BindName" = "\${result}"  
 ATTR: "DatabaseName" = "\${0}"  
 ATTR: "Filter" = "(userPrincipalName=\${1}@esrootdom.esdev.tdat)"  
 ATTR: "Match" = "(.+) "  
 ATTR: "Ref" = "esrootdom"  
 ATTR: "Scope" = "subtree"

Level 01: < LegalValues > (Element 4)

Level 02: < AlgorithmName > (Element 57)



```

ATTR: "AES" = "2"
ATTR: "Blowfish" = "1"
ATTR: "DH" = "5"
ATTR: "DIFFIE_HELLMAN" = "5"
ATTR: "MD5" = "3"
ATTR: "NONE" = "0"
ATTR: "SHA1" = "4"
ATTR: "SHA256" = "6"
ATTR: "SHA512" = "7"
DATA: ""

```

Level 02: < KeyLength > (Element 58)

```

ATTR: "K0" = "0"
ATTR: "K1024" = "1024"
ATTR: "K128" = "128"
ATTR: "K192" = "192"
ATTR: "K2048" = "2048"
ATTR: "K256" = "256"
ATTR: "K416" = "416"
ATTR: "K448" = "448"
ATTR: "K512" = "512"
DATA: ""

```

Level 02: < KeyLengthP > (Element 59)

```

ATTR: "K2048" = "2048"
DATA: ""

```

Level 02: < Mode > (Element 60)

```

ATTR: "CBC" = "1"
ATTR: "CCM" = "6"
ATTR: "CFB" = "2"
ATTR: "CTR" = "7"
ATTR: "ECB" = "3"
ATTR: "GCM" = "5"
ATTR: "NONE" = "0"
ATTR: "OFB" = "4"
DATA: ""

```

Level 02: < Padding > (Element 61)

```

ATTR: "NoPadding" = "0"

```

```

ATTR: "OAEPWithDIGESTAndMGFPadding" = "1"
ATTR: "PKCS1Padding" = "3"
ATTR: "PKCS5Padding" = "4"
ATTR: "SSL3Padding" = "5"
DATA: ""

Level 02: < InterfaceType > (Element 62)

DATA: "gss sspi teradata custom negotiate"

Level 02: < AlgorithmType > (Element 63)

DATA: "Confidentiality Integrity KeyExchange"

Level 02: < MechanismProperties > (Element 64)

ATTR: "AnonymousAuthentication" = "23"
ATTR: "AuthenticationSupported" = "2"
ATTR: "AuthorizationSupported" = "3"
ATTR: "CACertDir" = "65"
ATTR: "CACertFile" = "64"
ATTR: "CertificateFile" = "60"
ATTR: "ConfidentialityDesired" = "21"
ATTR: "CredentialIsUPN" = "67"
ATTR: "CredentialUsage" = "26"
ATTR: "DHKeyG" = "101"
ATTR: "DHKeyG2048" = "103"
ATTR: "DHKeyP" = "100"
ATTR: "DHKeyP2048" = "102"
ATTR: "DefaultMechanism" = "16"
ATTR: "DefaultNegotiatingMechanism" = "112"
ATTR: "DelegateCredentials" = "18"
ATTR: "DesiredContextTime" = "24"
ATTR: "DesiredCredentialTime" = "25"
ATTR: "GenerateCredentialFromLogon" = "41"
ATTR: "IntegrityDesired" = "22"
ATTR: "JWTDecryptionKeyFile" = "114"
ATTR: "JWTSkewTime" = "116"
ATTR: "JWTVerificationKeyFile" = "115"
ATTR: "LdapAllowUnsafeServerConnect" = "57"
ATTR: "LdapBaseFQDN" = "32"
ATTR: "LdapClientDebug" = "38"
ATTR: "LdapClientDeref" = "37"
ATTR: "LdapClientMechanism" = "43"

```

```

ATTR: "LdapClientRandomDevice" = "40"
ATTR: "LdapClientRebindAuth" = "39"
ATTR: "LdapClientReferrals" = "36"
ATTR: "LdapClientSaslSecProps" = "56"
ATTR: "LdapClientTlsCACert" = "47"
ATTR: "LdapClientTlsCACertDir" = "48"
ATTR: "LdapClientTlsCRLCheck" = "54"
ATTR: "LdapClientTlsCert" = "49"
ATTR: "LdapClientTlsCipherSuite" = "53"
ATTR: "LdapClientTlsKey" = "50"
ATTR: "LdapClientTlsRandFile" = "51"
ATTR: "LdapClientTlsReqCert" = "52"
ATTR: "LdapClientUseTls" = "45"
ATTR: "LdapGroupBaseFQDN" = "34"
ATTR: "LdapServerName" = "28"
ATTR: "LdapServerPort" = "29"
ATTR: "LdapServerRealm" = "30"
ATTR: "LdapServiceBindRequired" = "44"
ATTR: "LdapServiceFQDN" = "46"
ATTR: "LdapServicePassword" = "42"
ATTR: "LdapServicePasswordFile" = "117"
ATTR: "LdapServicePasswordProtected" = "55"
ATTR: "LdapSystemFQDN" = "31"
ATTR: "LdapUserBaseFQDN" = "35"
ATTR: "MechanismEnabled" = "1"
ATTR: "MechanismIgnoresQop" = "111"
ATTR: "MechanismRank" = "17"
ATTR: "MutualAuthentication" = "19"
ATTR: "NegotiationSupported" = "113"
ATTR: "OutOfSequenceDetection" = "33"
ATTR: "PrivateKeyFile" = "61"
ATTR: "PrivateKeyPassword" = "62"
ATTR: "PrivateKeyPasswordProtected" = "63"
ATTR: "ProxySupported" = "59"
ATTR: "ReplayDetection" = "20"
ATTR: "SigningHashAlgorithm" = "66"
ATTR: "SingleSignOnSupported" = "8"
ATTR: "TeradataKeyTab" = "110"
ATTR: "UseLdapConfig" = "58"
ATTR: "VerifyDHKey" = "27"
DATA: ""

```

Level 02: < ConfigFileType > (Element 65)

DATA: "Library User"

Level 01: < GlobalQOPs > (Element 5)

Level 02: < GlobalQOP > (Element 66)

ATTR: "ConfidentialityAlgorithm" = "Blowfish"

ATTR: "IntegrityAlgorithm" = "NONE"

ATTR: "KeyExchangeAlgorithm" = "DH"

ATTR: "KeyLength" = "K128"

ATTR: "KeyLengthP" = "K2048"

ATTR: "Mode" = "ECB"

ATTR: "Padding" = "NoPadding"

ATTR: "Value" = "GLOBAL\_QOP\_0"

Level 02: < GlobalQOP > (Element 67)

ATTR: "ConfidentialityAlgorithm" = "AES"

ATTR: "IntegrityAlgorithm" = "SHA1"

ATTR: "KeyExchangeAlgorithm" = "DH"

ATTR: "KeyLength" = "K128"

ATTR: "KeyLengthP" = "K2048"

ATTR: "Mode" = "OFB"

ATTR: "Padding" = "PKCS5Padding"

ATTR: "Value" = "GLOBAL\_QOP\_1"

Level 02: < GlobalQOP > (Element 68)

ATTR: "ConfidentialityAlgorithm" = "AES"

ATTR: "IntegrityAlgorithm" = "SHA1"

ATTR: "KeyExchangeAlgorithm" = "DH"

ATTR: "KeyLength" = "K128"

ATTR: "KeyLengthP" = "K2048"

ATTR: "Mode" = "CBC"

ATTR: "Padding" = "PKCS5Padding"

ATTR: "Value" = "AES-K128\_CBC\_PKCS5Padding\_SHA1\_DH-K2048"

Level 02: < GlobalQOP > (Element 69)

ATTR: "ConfidentialityAlgorithm" = "AES"

ATTR: "IntegrityAlgorithm" = "SHA1"

ATTR: "KeyExchangeAlgorithm" = "DH"

ATTR: "KeyLength" = "K192"

ATTR: "KeyLengthP" = "K2048"

```
ATTR: "Mode" = "CBC"
ATTR: "Padding" = "PKCS5Padding"
ATTR: "Value" = "AES-K192_CBC_PKCS5Padding_SHA1_DH-K2048"
```

Level 02: < GlobalQOP > (Element 70)

```
ATTR: "ConfidentialityAlgorithm" = "AES"
ATTR: "IntegrityAlgorithm" = "SHA1"
ATTR: "KeyExchangeAlgorithm" = "DH"
ATTR: "KeyLength" = "K256"
ATTR: "KeyLengthP" = "K2048"
ATTR: "Mode" = "CBC"
ATTR: "Padding" = "PKCS5Padding"
ATTR: "Value" = "AES-K256_CBC_PKCS5Padding_SHA1_DH-K2048"
```

Level 02: < GlobalQOP > (Element 71)

```
ATTR: "ConfidentialityAlgorithm" = "AES"
ATTR: "IntegrityAlgorithm" = "SHA256"
ATTR: "KeyExchangeAlgorithm" = "DH"
ATTR: "KeyLength" = "K128"
ATTR: "KeyLengthP" = "K2048"
ATTR: "Mode" = "GCM"
ATTR: "Padding" = "PKCS5Padding"
ATTR: "Value" = "AES-K128_GCM_PKCS5Padding_SHA2_DH-K2048"
```

Level 02: < GlobalQOP > (Element 72)

```
ATTR: "ConfidentialityAlgorithm" = "AES"
ATTR: "IntegrityAlgorithm" = "SHA256"
ATTR: "KeyExchangeAlgorithm" = "DH"
ATTR: "KeyLength" = "K192"
ATTR: "KeyLengthP" = "K2048"
ATTR: "Mode" = "GCM"
ATTR: "Padding" = "PKCS5Padding"
ATTR: "Value" = "AES-K192_GCM_PKCS5Padding_SHA2_DH-K2048"
```

Level 02: < GlobalQOP > (Element 73)

```
ATTR: "ConfidentialityAlgorithm" = "AES"
ATTR: "IntegrityAlgorithm" = "SHA256"
ATTR: "KeyExchangeAlgorithm" = "DH"
ATTR: "KeyLength" = "K256"
ATTR: "KeyLengthP" = "K2048"
```

```
ATTR: "Mode" = "GCM"
ATTR: "Padding" = "PKCS5Padding"
ATTR: "Value" = "AES-K256_GCM_PKCS5Padding_SHA2_DH-K2048"
```

Level 02: < GlobalQOP > (Element 74)

```
ATTR: "ConfidentialityAlgorithm" = "AES"
ATTR: "IntegrityAlgorithm" = "SHA256"
ATTR: "KeyExchangeAlgorithm" = "DH"
ATTR: "KeyLength" = "K128"
ATTR: "KeyLengthP" = "K2048"
ATTR: "Mode" = "CCM"
ATTR: "Padding" = "PKCS5Padding"
ATTR: "Value" = "AES-K128_CCM_PKCS5Padding_SHA2_DH-K2048"
```

Level 02: < GlobalQOP > (Element 75)

```
ATTR: "ConfidentialityAlgorithm" = "AES"
ATTR: "IntegrityAlgorithm" = "SHA256"
ATTR: "KeyExchangeAlgorithm" = "DH"
ATTR: "KeyLength" = "K192"
ATTR: "KeyLengthP" = "K2048"
ATTR: "Mode" = "CCM"
ATTR: "Padding" = "PKCS5Padding"
ATTR: "Value" = "AES-K192_CCM_PKCS5Padding_SHA2_DH-K2048"
```

Level 02: < GlobalQOP > (Element 76)

```
ATTR: "ConfidentialityAlgorithm" = "AES"
ATTR: "IntegrityAlgorithm" = "SHA256"
ATTR: "KeyExchangeAlgorithm" = "DH"
ATTR: "KeyLength" = "K256"
ATTR: "KeyLengthP" = "K2048"
ATTR: "Mode" = "CCM"
ATTR: "Padding" = "PKCS5Padding"
ATTR: "Value" = "AES-K256_CCM_PKCS5Padding_SHA2_DH-K2048"
```

Level 02: < GlobalQOP > (Element 77)

```
ATTR: "ConfidentialityAlgorithm" = "AES"
ATTR: "IntegrityAlgorithm" = "SHA256"
ATTR: "KeyExchangeAlgorithm" = "DH"
ATTR: "KeyLength" = "K128"
ATTR: "KeyLengthP" = "K2048"
```

```
ATTR: "Mode" = "CTR"
ATTR: "Padding" = "PKCS5Padding"
ATTR: "Value" = "AES-K128_CTR_PKCS5Padding_SHA2_DH-K2048"
```

Level 02: < GlobalQOP > (Element 78)

```
ATTR: "ConfidentialityAlgorithm" = "AES"
ATTR: "IntegrityAlgorithm" = "SHA256"
ATTR: "KeyExchangeAlgorithm" = "DH"
ATTR: "KeyLength" = "K192"
ATTR: "KeyLengthP" = "K2048"
ATTR: "Mode" = "CTR"
ATTR: "Padding" = "PKCS5Padding"
ATTR: "Value" = "AES-K192_CTR_PKCS5Padding_SHA2_DH-K2048"
```

Level 02: < GlobalQOP > (Element 79)

```
ATTR: "ConfidentialityAlgorithm" = "AES"
ATTR: "IntegrityAlgorithm" = "SHA256"
ATTR: "KeyExchangeAlgorithm" = "DH"
ATTR: "KeyLength" = "K256"
ATTR: "KeyLengthP" = "K2048"
ATTR: "Mode" = "CTR"
ATTR: "Padding" = "PKCS5Padding"
ATTR: "Value" = "AES-K256_CTR_PKCS5Padding_SHA2_DH-K2048"
```

Do you want to select another configuration file? (y/n): n

# Privilege Dictionary

## Privileges

Below is a description of each of the columns in the privileges table that follows:

Column	Description
Privilege	Name of privilege.
Abbreviation in System Views	Two-letter code used in system views to represent each privilege granted on a particular object. The AccessRight column in the following views lists privileges for users and roles: <ul style="list-style-type: none"> <li>• DBC.AllRightsV</li> <li>• DBC.UserGrantedRightsV</li> <li>• DBC.UserRightsV</li> <li>• DBC.AllRoleRightsV</li> <li>• DBC.UserRoleRightsV</li> </ul>
Automatically Granted: Creators	When you create a user or database, the system automatically grants you privileges on the created database or user.
Automatically Granted: Created User or Database	When you create a user or database, it automatically gets certain privileges on itself.
Explicitly Granted: Creators	When you create a user, you do not automatically receive certain privileges on the user.
Explicitly Granted: Created User or Database	Although the system automatically grants many database privileges, some privileges are only available to users if you explicitly grant them. You must have the WITH GRANT OPTION privilege on these privileges before you can grant them. When you create a user, it does not automatically get certain privileges on itself.
Group	Functional area of privilege.

The table below lists information about each of the privileges.

Privilege	Abbreviation in System Views	Automatically Granted		Explicitly Granted		Privilege Category
		Creators	Created User or Database	Creators	Created User or Database	
ABORT SESSION or ABORTSESSION	AS	No	No	Yes	Yes	Monitor



Privilege	Abbreviation in System Views	Automatically Granted		Explicitly Granted		Privilege Category
		Creators	Created User or Database	Creators	Created User or Database	
ALTER EXTERNAL PROCEDURE	AE	No	No	Yes	Yes	
ALTER FUNCTION	AF	No	No	Yes	Yes	
ALTER PROCEDURE	AP	No	No	Yes	Yes	
ANY	AN	Yes	Yes	No	No	Indicates a HELP or SHOW statement for which at least one privilege, but no specific privilege, is required.
CHECKPOINT	CP	Yes	Yes	No	No	
CONNECT THROUGH		No	No	Yes	Yes	Trusted session- related
CONSTRAINT ASSIGNMENT	SA	No	No	Yes	Yes	System- level
CONSTRAINT DEFINITION	SD	No	No	Yes	Yes	System- level
CREATE AUTHORIZATION	CA	Yes	Yes	No	No	
CREATE DATABASE	CD	Yes	No	No	Yes	
CREATE DATASET SCHEMA	C1	No	No	Yes	Yes	Dataset Schema
CREATE EXTERNAL PROCEDURE	CE	No	No	Yes	Yes	
CREATE FUNCTION	CF	No	No	Yes	Yes	
CREATE GLOP	GC	No	No	Yes	Yes	Global and Persistent

Privilege	Abbreviation in System Views	Automatically Granted		Explicitly Granted		Privilege Category
		Creators	Created User or Database	Creators	Created User or Database	
						(GLOP) Data
CREATE MACRO	CM	Yes	Yes	No	No	
CREATE MAP	MC	No	No	Yes	Yes	System- level
CREATE OWNER PROCEDURE	OP	No	No	Yes	Yes	
CREATE PROCEDURE	PC	No	No	Yes	Yes	
CREATE PROFILE	CO	No	No	Yes	Yes	System- level
CREATE ROLE	CR	No	No	Yes	Yes	System- level
CREATE SERVER	CS	No	No	Yes	Yes	
CREATE TABLE	CT	Yes	Yes	No	No	
CREATE TRIGGER	CG	Yes	Yes	No	No	
CREATE USER	CU	Yes	No	No	Yes	
CREATE VIEW	CV	Yes	Yes	No	No	
CREATE ZONE	CZ	No	No	Yes	Yes	System- level
CTCONTROL	TH	No	No	Yes	Yes	Trusted session- related
DELETE	D	Yes	Yes	No	No	
DROP AUTHORIZATION	DA	Yes	Yes	No	No	
DROP DATABASE	DD	Yes	No	No	Yes	
DROP DATASET SCHEMA	D1	Yes	No	Yes	Yes	Dataset Schema
DROP FUNCTION	DF	Yes	Yes	No	No	
DROP GLOP	GD	No	No	Yes	Yes	Global and Persistent

Privilege	Abbreviation in System Views	Automatically Granted		Explicitly Granted		Privilege Category
		Creators	Created User or Database	Creators	Created User or Database	
						(GLOP) Data
DROP MACRO	DM	Yes	Yes	No	No	
DROP MAP	MD	No	No	Yes	Yes	System- level
DROP PROCEDURE	PD	Yes	Yes	No	No	
DROP PROFILE	DO	No	No	Yes	Yes	System- level
DROP ROLE	DR	No	No	Yes	Yes	System- level
DROP SERVER	DS	No	No	Yes	Yes	
DROP TABLE	DT	Yes	Yes	No	No	
DROP TRIGGER	DG	Yes	Yes	No	No	
DROP USER	DU	Yes	No	No	Yes	
DROP VIEW	DV	Yes	Yes	No	No	
DROP ZONE	DZ	No	No	Yes	Yes	System- level
DUMP	DP	Yes	Yes	No	No	
EXECUTE	E	Yes	Yes	No	No	
EXECUTE FUNCTION	EF	No	No	Yes	Yes	
EXECUTE PROCEDURE	PE	No	No	Yes	Yes	
GLOP MEMBER	GM	No	No	Yes	Yes	Global and Persistent (GLOP) Data
INDEX	IX	No	No	Yes	Yes	Table-level
INSERT	I	Yes	Yes	No	No	
MONITOR RESOURCE or MONRESOURCE	MR	No	No	Yes	Yes	Monitor

Privilege	Abbreviation in System Views	Automatically Granted		Explicitly Granted		Privilege Category
		Creators	Created User or Database	Creators	Created User or Database	
MONITOR SESSION or MONSESSION	MS	No	No	Yes	Yes	Monitor
NONTEMPORAL	NT	No	No	Yes	Yes	Temporal
OVERRIDE DELETE	OD	No	No	Yes	Yes	Security Constraint Override
OVERRIDE DUMP	OA	No	No	Yes	Yes	Security Constraint Override
OVERRIDE INSERT	OI	No	No	Yes	Yes	Security Constraint Override
OVERRIDE RESTORE	OR	No	No	Yes	Yes	Security Constraint Override
OVERRIDE SELECT	OS	No	No	Yes	Yes	Security Constraint Override
OVERRIDE UPDATE	OU	No	No	Yes	Yes	Security Constraint Override
REFERENCES	RF	No	No	Yes	Yes	Table-level
RESTORE	RS	Yes	Yes	No	No	
RETRIEVE /SELECT	R	Yes	Yes	No	No	
SET RESOURCE RATE or SETRESRATE	SR	No	No	Yes	Yes	Monitor
SET SESSION RATE or SETSESSRATE	SS	No	No	Yes	Yes	Monitor
SHOW	SH	No	No	Yes	Yes	
STATISTICS	ST	Yes	Yes	No	No	Also for creators of tables

Privilege	Abbreviation in System Views	Automatically Granted		Explicitly Granted		Privilege Category
		Creators	Created User or Database	Creators	Created User or Database	
UDT METHOD or UDTMETHOD	UM	No	No	Yes	Yes	UDT
UDT TYPE or UDTTYPE	UT	No	No	Yes	Yes	UDT
UDT USAGE or UDTUSAGE	UU	No	No	Yes	Yes	UDT
UPDATE	U	Yes	Yes	No	No	
WITH DATASET SCHEMA	W1	Yes	No	Yes	Yes	Dataset Schema
ZONE OVERRIDE	ZO	No	No	No	No	System- level

## Multiple Privileges with a Single Keyword

Teradata Vantage provides a special category of keywords that you can use to grant multiple privileges. For example, the following request uses the keyword DATABASE to grant both the CREATE DATABASE and DROP DATABASE privileges on *user\_name1* to *user\_name2*:

```
GRANT DATABASE ON user_name1 TO user_name2;
```

The following keyword grants enter multiple privileges for the grantee in the DBC.AccessRights table:

Keyword	Included Privileges
ALL	All implicit and explicit object privileges that the grantor owns, and on which the grantor has WITH GRANT OPTION, on the object specified in the ON clause.
CHECKPOINT	The privilege to execute the: <ul style="list-style-type: none"> <li>CHECKPOINT SQL statement.</li> <li>HUT CHECKPOINT command.</li> </ul>
DATABASE	<ul style="list-style-type: none"> <li>CREATE DATABASE</li> <li>DROP DATABASE</li> </ul>
FUNCTION	<ul style="list-style-type: none"> <li>CREATE FUNCTION</li> <li>DROP FUNCTION</li> </ul>
GLOP	<ul style="list-style-type: none"> <li>CREATE GLOP</li> <li>DROP GLOP</li> </ul>

Keyword	Included Privileges
INDEX	<ul style="list-style-type: none"> <li>• CREATE INDEX</li> <li>• DROP INDEX</li> </ul>
MACRO	<ul style="list-style-type: none"> <li>• CREATE MACRO</li> <li>• DROP MACRO</li> </ul>
MAP	<ul style="list-style-type: none"> <li>• CREATE MAP</li> <li>• DROP MAP</li> </ul>
OVERRIDE	<ul style="list-style-type: none"> <li>• OVERRIDE INSERT</li> <li>• OVERRIDE SELECT</li> <li>• OVERRIDE UPDATE</li> <li>• OVERRIDE DELETE</li> <li>• OVERRIDE DUMP</li> <li>• OVERRIDE RESTORE</li> </ul>
PROCEDURE	<ul style="list-style-type: none"> <li>• CREATE PROCEDURE</li> <li>• DROP PROCEDURE</li> </ul>
PROFILE	<ul style="list-style-type: none"> <li>• CREATE PROFILE</li> <li>• DROP PROFILE</li> </ul>
RESTORE	<p>The privilege to execute the following HUT commands:</p> <ul style="list-style-type: none"> <li>• DELETE JOURNAL</li> <li>• ROLLBACK</li> <li>• ROLLFORWARD</li> </ul>
ROLE	<ul style="list-style-type: none"> <li>• CREATE ROLE</li> <li>• DROP ROLE</li> </ul>
SHOW	<p>Only the following variations of the HELP and SHOW commands:</p> <ul style="list-style-type: none"> <li>• HELP <i>database_object</i></li> <li>• SHOW <i>database_object</i></li> </ul>
TABLE	<ul style="list-style-type: none"> <li>• CREATE TABLE</li> <li>• DROP TABLE</li> </ul>
TRIGGER	<ul style="list-style-type: none"> <li>• CREATE TRIGGER</li> <li>• DROP TRIGGER</li> </ul>
USER	<ul style="list-style-type: none"> <li>• CREATE USER</li> <li>• DROP USER</li> </ul> <p><b>Note:</b> If the target of a GRANT USER is a user, then USER confers CREATE and DROP privileges. If the target of a GRANT USER is a database, then USER confers the privilege to CREATE users within the database.</p>

Keyword	Included Privileges
VIEW	<ul style="list-style-type: none"> <li>• CREATE VIEW</li> <li>• DROP VIEW</li> </ul>
ZONE	<ul style="list-style-type: none"> <li>• CREATE ZONE</li> <li>• DROP ZONE</li> </ul>

## Required DBC Privileges

DBC needs privileges, for example, DELETE, UPDATE, and CREATE, on certain Data Dictionary objects, to enable the system to maintain the Data Dictionary. The system setup process grants these privileges by default.

### NOTICE

Never revoke DBC privileges. Re-granting DBC privileges requires assistance from the Teradata Support Center.

## Default PUBLIC Privileges

By default, the system grants several default privileges to PUBLIC on specific Data Dictionary objects, and all valid database users automatically have PUBLIC privileges. You can grant and revoke PUBLIC privileges according to your site security policy, but you should keep a list of the default privileges so you can re-grant them to PUBLIC if necessary.

Until you make changes, you can query the system to view the default PUBLIC privileges:

```
SELECT DatabaseName, TableName, AccessRight FROM DBC.AllRights WHERE
UserName='PUBLIC' ORDER BY 2, 1;
```

### Default PUBLIC Privileges Required by Client Applications

The following privilege...	Granted to PUBLIC on...	Affects the following tools and utilities...
SELECT	DBC.Databases	NetBackup™ Teradata Extension
SELECT	DBC.DBCInfo	Teradata System Emulation Tool (Teradata SET)
SELECT	DBC.TDStats	Teradata Viewpoint Stats Manager. See <i>Teradata® Unity™ Installation, Configuration, and Upgrade Guide for Customers</i> , B035-2523.
SELECT	DBC.Tables	NetBackup™ Teradata Extension, Teradata SET

The following privilege...	Granted to PUBLIC on...	Affects the following tools and utilities...
INSERT, SELECT, DELETE, UPDATE	SysAdmin.FastLog <sup>a</sup>	FastExport, FastLoad, MultiLoad
EXECUTE	SysAdmin.FastLogIns <sup>a</sup>	FastExport, FastLoad, MultiLoad
SELECT	SysAdmin.FastLogRestartV <sup>a</sup>	FastExport, FastLoad, MultiLoad
EXECUTE	SysAdmin.FastLogUpd <sup>a</sup>	FastExport, FastLoad, MultiLoad
SELECT	SysAdmin.FastLogV <sup>a</sup>	FastExport, FastLoad, MultiLoad
<sup>a</sup> Denotes an internal table containing FastLoad information.		

## Privileges Needed for Database Administration

The following topics describe the privileges required by administrators to manage common database objects and processes.

This is a privilege quick reference. For details, see the information about required privileges for each SQL statement in:

- *Teradata Vantage™ - SQL Data Control Language*, B035-1149
- *Teradata Vantage™ - SQL Data Definition Language Syntax and Examples*, B035-1144

## Databases

To...	the user must have the...
create a database	CREATE DATABASE privilege on the immediate owner database.
modify or drop a database	DROP DATABASE privilege on the database. <b>Note:</b> A database must be empty before it can be dropped.

## DATASET SCHEMA

Action	User Privilege Required
Create a DATASET schema	CREATE DATASET SCHEMA privilege on the containing database or user.
Drop a DATASET schema	DROP DATASET SCHEMA privilege on the database or user or on the individual schema-level. <b>Note:</b> To be dropped, the schema must also not be in use.



Action	User Privilege Required
WITH DATASET SCHEMA Provide users with permission to associate a created schema with a column of a table	WITH DATASET SCHEMA privilege on the database or user or on the individual schema-level. The creator of a schema is automatically granted this privilege with grant option on the created schema.

## Geospatial Data Types

Action	Privilege Required
Grant the use of geospatial data types with their associated functions, metadata tables, and procedures	<ul style="list-style-type: none"> <li>• GRANT ALL ON SYSSPATIAL TO SYSSPATIAL WITH GRANT OPTION;</li> <li>• GRANT SELECT ON DBC TO SYSSPATIAL WITH GRANT OPTION;</li> <li>• GRANT EXECUTE PROCEDURE ON SYSSPATIAL TO DBC WITH GRANT OPTION;</li> <li>• GRANT UDTMETHOD ON SYSUDTLIB TO SYSSPATIAL;</li> <li>• GRANT UDTTYPE ON SYSUDTLIB TO SYSSPATIAL;</li> <li>• GRANT CREATE FUNCTION ON SYSUDTLIB TO SYSSPATIAL;</li> </ul>
Grant the use of the GeoSequence date type	<ul style="list-style-type: none"> <li>• GRANT UDTUSAGE ON SYSUDTLIB TO <i>geouser</i>;</li> <li>• GRANT EXECUTE FUNCTION ON SYSSPATIAL TO <i>geouser</i>;</li> <li>• GRANT SELECT ON SYSPATIAL TO <i>geouser</i>;</li> <li>• GRANT EXECUTE PROCEDURE ON SYSPATIAL TO <i>geouser</i>;</li> </ul>

## GLOP

To...	the user must have the...
create a GLOP set	CREATE GLOP privilege on the containing database or user.
drop a GLOP set	DROP GLOP privilege on the database or user containing GLOP_set_name.  <b>Note:</b> The DROP GLOP privilege is automatically granted to the creator and owner of a GLOP set.
allow a user/database to specify access to a GLOP set that is from a database other than the one that contains the external routine referencing the GLOP set	GLOP MEMBER privilege If the GLOP is in the same database as the external routine, access to the GLOP is automatically granted. The database where the external routine resides must have either CREATE GLOP, DROP GLOP or GLOP MEMBER privileges on the GLOP set

## Hash Index and Join Index

To...	the user must have the...
create a hash or join index	CREATE TABLE privilege on the database in which the hash index or join index is created.
drop a hash index	DROP TABLE or INDEX privilege on the indexed base table or its containing database.
drop a join index	DROP TABLE or INDEX privilege on each of the covered tables or their containing databases.

## Macros

To...	the user must have the...
create a macro	<p>CREATE MACRO privilege on the database or user in which the macro is to be created. The creator automatically gets the DROP MACRO and EXECUTE privileges WITH GRANT OPTION. The immediate owner of the macro:</p> <ul style="list-style-type: none"> <li>• is the database in which it exists, not the user who created it</li> <li>• determines the macro access privileges, not the macro</li> <li>• must have all the appropriate privileges for executing the macro, including WITH GRANT OPTION</li> </ul>
drop a macro	DROP MACRO privilege on the macro.
rename a macro	<p>DROP MACRO privilege on the macro.</p> <p>In addition, the user renaming a macro must have the privileges for all statements the macro performs.</p>
execute a macro	<p>EXECUTE privilege on the macro.</p> <p>In addition, the immediate owner of the macro (the database in which the macro resides) must have the necessary privileges on objects named in the request set for statements that are contained in the macro.</p>
replace a macro	<ul style="list-style-type: none"> <li>• privileges for all statements the macro performs</li> <li>• privileges that depend on whether the macro being replaced already exists. <ul style="list-style-type: none"> <li>◦ If the macro already exists, the DROP MACRO privilege on the macro or its containing database or user.</li> <li>◦ If the macro does not already exist, the CREATE MACRO privilege on the macro or its containing database or user</li> </ul> </li> </ul> <p><b>Note:</b></p> <p>Once a macro has been replaced, its immediate owner is the database in which it exists, not the user who replaced it. The immediately owning database must have all the appropriate privileges for executing the macro, including WITH GRANT OPTION.</p>
delete a macro	DROP MACRO privilege on the specified macro

## Teradata Vantage MAPS architecture (MAPS)

To...	the user must have the...
create a map	<p>CREATE MAP or MAP privilege must be granted to the user or role.</p> <ul style="list-style-type: none"> <li>Only sparse maps can be created. A sparse map is specific to the secure zone of the creator.</li> <li>The user must be granted the parent contiguous map to create a sparse map with CREATE MAP.</li> </ul>
drop a map	<p>DROP MAP or MAP privilege.</p> <ul style="list-style-type: none"> <li>Users and roles within a secure zone cannot drop a sparse map created in another secure zone and cannot drop a contiguous map.</li> </ul>
grant/revoke map privileges from users or roles	<p>The user must have the map privilege WITH GRANT OPTION to grant or revoke a map privilege.</p> <ul style="list-style-type: none"> <li>The CREATE MAP and DROP MAP privileges are system-level privileges (they cannot be granted/revoked on an object).</li> </ul>
grant/revoke a map	<ul style="list-style-type: none"> <li>A user must be granted the map WITH GRANT OPTION to grant or revoke that map to other users, roles, or PUBLIC. Roles cannot be granted a map WITH GRANT OPTION.</li> <li>The creator of a map is automatically granted the specified map WITH GRANT OPTION, which allows the creator to grant or revoke the created sparse map to users, PUBLIC, and roles.</li> <li>You must be in the same secure zone as the sparse map to grant or revoke it.</li> <li>The GRANT OPTION FOR clause cannot be specified when revoking a map from a role or PUBLIC.</li> <li>If GRANT OPTION FOR is specified, only the WITH GRANT OPTION is revoked.</li> </ul>

## Profiles

To...	the user must have the...
create a profile	CREATE PROFILE privilege (system level).
modify or drop a profile	DROP PROFILE privilege (system level).

## Proxy Users and Trusted Users

To...	the user must have the...
<p>execute a GRANT CONNECT THROUGH statement to:</p> <ul style="list-style-type: none"> <li>Define a trusted user (application logon user)</li> <li>Identified associated proxy users and roles</li> <li>Specify the WITH TRUST ONLY option to prevent the use of SET QUERY BAND statements that can change proxy user role assignments</li> </ul>	CTCONTROL privilege (system level).

To...	the user must have the...
use the SET QUERY BAND statement to set a proxy user role	GRANT CONNECT THROUGH privilege that specifies the user, role, and trusted user application

## Roles and External Roles

To...	the user must have...
create a role or external role	the CREATE ROLE privilege (system level).
drop a role or external role	one of the following privileges: <ul style="list-style-type: none"> <li>• DROP ROLE privilege (system level)</li> <li>• role membership WITH ADMIN OPTION</li> </ul>
grant privileges to a role, or grant role membership	The role granted WITH ADMIN OPTION. <b>Note:</b> The creator of a role is automatically granted the specified role WITH ADMIN OPTION.

## Security Constraints

To...	the user must have the...
create, alter, or drop CONSTRAINT objects	CONSTRAINT DEFINITION privilege (system level).
<ul style="list-style-type: none"> <li>• assign constraints to, and remove them from, users and profiles</li> <li>• define and remove security constraint table columns in tables, views and indexes</li> <li>• grant and revoke security constraint OVERRIDES</li> </ul>	CONSTRAINT ASSIGNMENT privilege (system level). For managing security constraints for users, tables, views and indexes the user must also have the necessary CREATE/REPLACE/ MODIFY privileges on the object.
execute the SHOW CONSTRAINT statement	CONSTRAINT DEFINITION or CONSTRAINT ASSIGNMENT privilege.
bypass enforcement of security constraint UDFs	OVERRIDE INSERT, OVERRIDE SELECT, OVERRIDE UPDATE, OVERRIDE DELETE for the security constraint, on the object being accessed.

## Statistics

To...	the user must have...
collect/drop statistics on a permanent or global temporary table, a hash index, or a join index	the STATISTICS privilege on the table, global temporary table, hash index or join index.

To...	the user must have...
collect/drop statistics on a permanent row level security table	the STATISTICS and OVERRIDE SELECT privileges on the table.
collect/drop statistics on a volatile table or a materialized global temporary table	No privileges required
use HELP STATISTICS to display summary or detail information for collected statistics for the specified data table or hash or join index	<ul style="list-style-type: none"> <li>Any privilege on the table_name, join_index_name, or hash_index_name</li> <li>Ownership of the table_name or join_index_name</li> </ul>
use SHOW STATISTICS to display summary or detail information for collected statistics for the specified data table or hash or join index	if the <i>values</i> option is not specified: <ul style="list-style-type: none"> <li>Any privilege on the table_name, join_index_name, or hash_index_name</li> <li>Ownership of the table_name or join_index_name</li> </ul> if the <i>values</i> option is specified: <ul style="list-style-type: none"> <li>SELECT on the object being reported</li> </ul>

## SQL and External Procedures

For information about privilege requirements and options of SQL and external procedures, see:

- Teradata Vantage™ - SQL Data Definition Language Syntax and Examples, B035-1144
- Teradata Vantage™ - SQL External Routine Programming, B035-1147

## Tables

To...	the user must have the...
create a table (including error table)	CREATE TABLE privilege on the database or user in which the table is created.
alter or drop a table	DROP TABLE on the table or the database that contains the table. Additional privileges are required for altering a table to add or change a PI, security constraint column, or UDT column.
drop or rename a table	DROP TABLE on the table or on the database containing the table
archive a table	DUMP privilege on the database or table or you must be the owner of the table. If the table has security constraint columns, you must also have OVERRIDE DUMP privilege.
copy a table	the following privileges: <ul style="list-style-type: none"> <li>CREATE MACRO</li> <li>CREATE TABLE</li> <li>CREATE VIEW</li> <li>RESTORE</li> </ul>

To...	the user must have the...
	<b>Note:</b> Copying of security constraint tables is not supported.
restore a table	CREATE TABLE privilege on the table being restored, or be an owner of the table. If the table has security constraint columns, you must also have OVERRIDE RESTORE privilege.

## Triggers

Creating or replacing a trigger does not grant trigger-related privileges to either the creator or the immediate owner of that trigger.

### Note:

You cannot grant privileges on a trigger, only on the database or table to which the trigger applies.

To...	the user must have the...
create a trigger	<ul style="list-style-type: none"> <li>• CREATE TRIGGER on both of the following:               <ul style="list-style-type: none"> <li>◦ The database in which the trigger is created</li> <li>◦ Either the subject table or its containing database</li> </ul> </li> <li>• SELECT on any column referenced in a WHEN clause or a triggered SQL statement subquery</li> <li>• INSERT, UPDATE, or DELETE on the triggered SQL statement target table (depending on the triggered action).</li> <li>• privileges that would normally be required to execute the individual triggered SQL statements</li> </ul>
replace a trigger	<ul style="list-style-type: none"> <li>• DROP TRIGGER on the subject table or the database                The exception is when you use REPLACE TRIGGER when no target trigger exists and you instead create a new trigger, in which case you need CREATE TRIGGER privilege on both of the following:               <ul style="list-style-type: none"> <li>◦ The database in which the trigger is created</li> <li>◦ Either the subject table or its containing database</li> </ul> </li> <li>• SELECT on any column referenced in a WHEN clause or a triggered SQL statement subquery</li> <li>• Depending on the triggered SQL statement, INSERT, UPDATE, or DELETE on the triggered SQL statement target table</li> <li>• privileges that would normally be required to execute the individual triggered SQL statements</li> </ul>
drop a trigger	DROP TRIGGER privilege on the subject table or the database containing the table
execute a trigger	privileges required for executing triggering statements. In addition, the immediate owner of the trigger must have:

To...	the user must have the...
	<ul style="list-style-type: none"> <li>• CREATE TRIGGER on the subject table or the database containing the table.</li> <li>• SELECT on any column referenced in the WHEN clause of the CREATE TRIGGER statement, or any column in a triggered action statement that requires read access for the execution of the statement.</li> </ul>

## Users

To...	the user must have the...
create a user	CREATE USER privilege.
modify or drop a user	DROP USER privilege.

## User-Defined Functions (UDFs)

To...	the user must have...
create a function that calls a UDF from the SYSUDTLIB database	<p>the CREATE FUNCTION privilege, on the containing database and on the SYSUDTLIB database.</p> <p>If you grant a user the CREATE FUNCTION privilege on a database, any function a user creates automatically has DROP FUNCTION, EXECUTE FUNCTION, and WITH GRANT OPTION on the function.</p>
<ul style="list-style-type: none"> <li>• change the execution mode of a function</li> <li>• recompile or re-link a function</li> </ul>	<p>the ALTER FUNCTION privilege.</p> <p>A UDF running in protected mode runs in a process created to run as the user called “tdatuser.” This user has no special operating system privileges. It is up to the site to decide what resources “tdatuser” is to have access to by setting access privileges to system resources as required by the site.</p> <p><b>Note:</b></p> <p>If you specify EXECUTE NOT PROTECTED, and the function fails during execution, the database could restart.</p>
use the SHOW FUNCTION statement to display CREATE/REPLACE FUNCTION text and the source code	<p>at least one privilege on the function or the database containing the function to display the CREATE/REPLACE FUNCTION text.</p> <p>If the user has the DROP FUNCTION privilege, the SHOW FUNCTION statement also displays the C source.</p>
drop a UDF	the DROP FUNCTION on the function or the containing database or user.
execute a function	<p>the EXECUTE FUNCTION privilege on the database or specific function.</p> <p>To grant the user privileges to execute a specific user-defined functions, submit:</p>

To...	the user must have...
	<pre>GRANT EXECUTE ON SPECIFIC FUNCTION <i>function_name</i> TO <i>username</i>;</pre> <p>To grant execute privileges on all UDFs in a database:</p> <pre>GRANT EXECUTE FUNCTION ON <i>dbname</i> TO <i>username</i>;</pre>
use the HELP FUNCTION to get: <ul style="list-style-type: none"> <li>the function name</li> <li>a list of parameters</li> <li>the data types of the parameters</li> <li>whether the function is used to compress or decompress character or graphic data</li> <li>any comments associated with the parameters for SQL, scalar, aggregate, and table functions</li> </ul>	at least one privilege on the function or the database containing the function.
rename a function	the DROP FUNCTION privilege on the function or the containing database, and the CREATE FUNCTION privilege on the database containing the function.
replace an existing function	the DROP FUNCTION privilege on the function or on the database containing the function.

**Note:**

If you specify a UDT as an input parameter or the function result, the current user must have one of the following privileges:

- UDTUSAGE on the SYSUDTLIB database.
- UDTUSAGE on the specified UDT.

For additional information on creating UDFs, see *Teradata Vantage™ - SQL External Routine Programming*, B035-1147.

## UDTs and UDMs

To...	the user must have the...
create, alter, or drop a UDT	UDTTYPE is only granted at the database level. This privilege includes all the abilities of UDTUSAGE plus the ability to create, alter, and drop UDTs. Users can also create or drop cast, ordering, or transform properties. Users with this privilege cannot, however, create new methods or drop and replace existing methods.
create, alter, or drop a UDM	UDTMETHOD. This privilege includes all the abilities of UDTTYPE plus ability to use, create, drop or alter any UDT and its methods without any restrictions.



To...	the user must have the...
use a UDT or UDM in a table or view, and execute all associated methods	UDTUSAGE. You can grant this privilege at both the database and object level. It is not an automatic privilege and a user must be granted this privilege or acquire it through a role. A user granted UDTUSAGE WITH GRANT OPTION can grant it (optionally also with the WITH GRANT OPTION) to others. UDTUSAGE allows users to execute all SQL statements that reference existing UDTs and their existing methods. It does not permit creation of new UDTs, altering the ordering, casting, or transform behavior of existing UDTs, or creating new methods.
create or alter a cast operation for a UDT	CREATE CAST and REPLACE CAST privileges.

## Views

The system automatically grants the following privileges on a newly created view to its creator:

- DELETE
- DROP VIEW
- INSERT
- SELECT
- UPDATE

To...	the user must have the...
create a view	CREATE VIEW privilege.
drop or replace a view	DROP VIEW privilege.

## Teradata Secure Zones

The DBC user or a user who already has the CREATE ZONE, DROP ZONE, and WITH GRANT OPTION privileges must explicitly grant these privileges to a zone creator.

To ...	the user must have the ...
create a zone and create a zone root	CREATE ZONE privilege
drop a zone	DROP ZONE privilege
add or drop a zone root	ALTER ZONE privilege
assign a primary DBA to a zone	CREATE USER AS DBA or MODIFY USER AS DBA
grant zone access to zone guests	GRANT ZONE
revoke the zone access of zone guests	REVOKE ZONE

## Determining Privileges for a User

You can create an AllUserRights macro to list all the privileges a user has on a specific database. The macro gets information from the DBC.AllRightsV and DBC.AllRolesRightsV views, and spells out the two character privilege code it finds in the AccessRightDesc field of the views.

### Sample Macro for Determining User Privileges

```
create macro database_name.AllUserRights (UserName char(128)) as (
locking row for access select
    UserName      (varchar(128))
  ,AccessType     (varchar(128))
  ,RoleName       (varchar(128))
  ,DatabaseName   (varchar(128))
  ,TableName      (varchar(128))
  ,ColumnName     (varchar(128))
  ,AccessRight
  ,case
    when accessright='AE' then 'ALTER EXTERNALPROCEDURE'
    when accessright='AF' then 'ALTER FUNCTION'
    when accessright='AP' then 'ALTER PROCEDURE'
    when accessright='AS' then 'ABORT SESSION'
    when accessright='CA' then 'CREATE AUTHORIZATION'
    when accessright='CD' then 'CREATE DATABASE'
    when accessright='CE' then 'CREATE EXTERNAL PROCEDURE'
    when accessright='CF' then 'CREATE FUNCTION'
    when accessright='CG' then 'CREATE TRIGGER'
    when accessright='CM' then 'CREATE MACRO'
    when accessright='CO' then 'CREATE PROFILE'
    when accessright='CP' then 'CHECKPOINT'
    when accessright='CR' then 'CREATE ROLE'
    when accessright='CS' then 'CREATE SERVER'
    when accessright='CT' then 'CREATE TABLE'
    when accessright='CU' then 'CREATE USER'
    when accessright='CV' then 'CREATE VIEW'
    when accessright='CZ' then 'CREATE ZONE'
    when accessright='C1' then 'CREATE DATASET SCHEMA'
    when accessright='D'  then 'DELETE'
    when accessright='DA' then 'DROP AUTHORIZATION'
    when accessright='DD' then 'DROP DATABASE'
    when accessright='DF' then 'DROP FUNCTION'
    when accessright='DG' then 'DROP TRIGGER'
    when accessright='DM' then 'DROP MACRO'
```

```

when accessright='DO' then 'DROP PROFILE'
when accessright='DP' then 'DUMP'
when accessright='DR' then 'DROP ROLE'
when accessright='DS' then 'DROP SERVER'
when accessright='DT' then 'DROP TABLE'
when accessright='DU' then 'DROP USER'
when accessright='DV' then 'DROP VIEW'
when accessright='DZ' then 'DROP ZONE'
when accessright='D1' then 'DROP DATASET SCHEMA'
when accessright='E' then 'EXECUTE'
when accessright='EF' then 'EXECUTE FUNCTION'
when accessright='GC' then 'CREATE GLOP'
when accessright='GD' then 'DROP GLOP'
when accessright='GM' then 'GLOP MEMBER'
when accessright='I' then 'INSERT'
when accessright='IX' then 'INDEX'
when accessright='MC' then 'CREATE MAP'
when accessright='MD' then 'DROP MAP'
when accessright='MR' then 'MONITOR RESOURCE'
when accessright='MS' then 'MONITOR SESSION'
when accessright='NT' then 'NONTEMPORAL'
when accessright='OD' then 'OVERRIDE DELETE POLICY'
when accessright='OI' then 'OVERRIDE INSERT POLICY'
when accessright='OP' then 'CREATE OWNER PROCEDURE'
when accessright='OS' then 'OVERRIDE SELECT POLICY'
when accessright='OU' then 'OVERRIDE UPDATE POLICY'
when accessright='PC' then 'CREATE PROCEDURE'
when accessright='PD' then 'DROP PROCEDURE'
when accessright='PE' then 'EXECUTE PROCEDURE'
when accessright='R' then 'RETRIEVE/SELECT'
when accessright='RF' then 'REFERENCES'
when accessright='RS' then 'RESTORE'
when accessright='SA' then 'SECURITY CONSTRAINT ASSIGNMENT'
when accessright='SD' then 'SECURITY CONSTRAINT DEFINITION'
when accessright='ST' then 'STATISTICS'
when accessright='SS' then 'SET SESSION RATE'
when accessright='SR' then 'SET RESOURCE RATE'
when accessright='TH' then 'CTCONTROL'
when accessright='U' then 'UPDATE'
when accessright='UU' then 'UDT Usage'
when accessright='UT' then 'UDT Type'
when accessright='UM' then 'UDT Method'
when accessright='W1' then 'WITH DATASET SCHEMA'
when accessright='ZO' then 'ZONE OVERRIDE'

```

```

else''
    end (varchar(26)) as AccessRightDesc
    ,GrantAuthority
    ,GrantorName (varchar(128))
    ,AllnessFlag
    ,CreatorName (varchar(128))
    ,CreateTimeStamp
from
(
select
    UserName
    ,'User' (varchar(128)) as AccessType
    ,' ' (varchar(128)) as RoleName
    ,DatabaseName
    ,TableName
    ,ColumnName
    ,AccessRight
    ,GrantAuthority
    ,GrantorName
    ,AllnessFlag
    ,CreatorName
    ,CreateTimeStamp
    from dbc.allrights
where UserName = :username
    and CreatorName not = :username
union all
select
    Grantee as UserName
    ,'Member' as UR
    ,r.RoleName
    ,DatabaseName
    ,TableName
    ,ColumnName
    ,AccessRight
    ,null (char(1)) as GrantAuthority
    ,GrantorName
    ,null (char(1)) as AllnessFlag
    ,null (char(1)) as CreatorName
    ,CreateTimeStamp
    from dbc.allrolerights r
    join dbc.rolemembers m
        on m.RoleName = r.RoleName
where UserName = :username
union all

```

```

select
    User as UserName
  ,m.Grantee as UR
  ,r.RoleName
  ,DatabaseName
  ,TableName
  ,ColumnName
  ,AccessRight
  ,null (char(1)) as GrantAuthority
  ,GrantorName
  ,null (char(1)) as AllnessFlag
  ,null (char(1)) as CreatorName
  ,CreateTimeStamp
from dbc.allrolerights r
join dbc.rolemembers m
    on m.RoleName = r.RoleName
where m.grantee in (select rolename from dbc.rolemembers where
grantee = :username)
) AllRights
order by 4,5,6,7; );

```

where *database\_name* is the name of a database in your system, for which the macro checks user privileges. For example, if you create the macro in the DBAdmin database, you identify the macro as DBAdmin.AllUserRights.

---

**Note:**

This macro returns all privileges granted to a user either directly or through a role. It does not return implicit (ownership) privileges.

---

## Executing the Privilege Check Macro

After you create the macro, you can execute it to check access privileges for a particular user with the command:

```
execute database_name.AllUserRights ('username');
```

***database\_name***

The name of the macro, and also identifies database for which the macro checks user privileges.

***username***

The name of a permanent database user for which the macro checks privileges. The user that executes the macro must have EXECUTE privileges on the database that contains the macro.

# Teradata GSS Administrative Package

TTU 16.10 and later no longer supports an independent TeraGSS package for client machines. Instead, the client interfaces embed TeraGSS functionality and the previous method of configuring TeraGSS is not possible. If you need to configure TeraGSS, you must install the Teradata Generic Security Service Administrative package (teragssAdmin) included in TTU.

---

**Note:**

Although teragssAdmin is provided for configuring TeraGSS, Teradata recommends not configuring TeraGSS on the clients.

---

## TeraGSS Limitations

The following lists the limitations in TeraGSS:

- There is no LDAP server-side support in TTU 16.10 and later. Configuring LDAP properties in the LDAP mechanism or configuring an <LdapConfig> section can no longer be done.
- There is no client side support for the SPNEGO mechanism in TTU. This mechanism is present only in TDGSS to support the .NET Data Provider for Teradata. So, you cannot configure SPNEGO using teragssAdmin on the client.
- There is no support for the PROXY mechanism in TTU. This mechanism is present only in TDGSS to support Unity. Unity now uses TDGSS rather than TeraGSS. See the Unity documentation for more information on configuring and managing PROXY: *Teradata® Unity™ Installation, Configuration, and Upgrade Guide for Customers*, B035-2523 and *Teradata® Unity™ User Guide*, B035-2520.
- TTU now provides configuration to set the preferred default TeraGSS authentication mechanism on a per-client machine basis. While setting the default mechanism is still possible with TeraGSS, the preferred method is to configure TTU rather than TeraGSS. See the Teradata Tools and Utilities documentation for more information.
- The TD1, NTLM, NTLMC, and KRB5C compatibility mechanisms are deprecated and have been removed.

## Guidelines for Configuring TeraGSS

The process for configuring TeraGSS and TDGSS is similar, so information about configuring TDGSS from other parts of this document can be used as a guide.

Ensure that the teragssAdmin package has been installed on the client machine. If it is not present, install it from the appropriate TTU package. This package is installed in the 16.10 TTU directory structure. The tools required to manage and debug a TeraGSS configuration are in the TTU bin directory. The supporting XML files and XSD schema are in the TTU etc directory.

If you already have a copy of the TdgssUserConfigFile.xml file in your TTU site directory, you may continue to use that file. If TdgssUserConfigFile.xml is configured for the deprecated mechanisms (see the list of limitations above) or contains configuration for PROXY and SPNEGO, those configurations must be removed. The only configurations that should remain are configurations for the TD2, LDAP, KRB5, JWT, and TDNEGO mechanisms.

If you do not have a copy of the TdgssUserConfigFile.xml file in your TTU site directory, you may copy the one in the TTU etc directory to the site directory and make edits to the copy.

Once your edits are complete, execute the run\_tdgssconfig script found in the TTU bin directory. This script will compile the changes you made to the TdgssUserConfigFile.xml file into the tdgssconfig.bin file located in the TTU etc directory.

## Configurable Items in TeraGSS

The following mechanism attributes and elements are configurable in TeraGSS, following the editing instructions found in other sections of this document:

- DefaultMechanism – Note that the preferred method of picking a per-client machine specific default mechanism is to configure it through TTU installation and configuration. We recommend that this attribute is set to no for all mechanisms in TeraGSS. See [DefaultMechanism](#) for editing guidelines.
- DefaultNegotiatingMechanism – See [DefaultNegotiatingMechanism](#) for editing guidelines.
- MechanismEnabled – See [MechanismEnabled](#) for editing guidelines.
- <MechQop> elements may be adjusted and managed if defaults are not good enough. See [QOP Configuration Options](#) for more information.
- MechanismRank – See [MechanismRank](#) for editing guidelines.
- <NegotiatedMechanism> - See [Configuring TDNEGO Properties](#) for more information.
- One or more <RequiredLibraryPath> elements may be added to the KRB5 mechanism to specify the location of a libgssapi\_krb5.so library when this library resides in a non-standard location. Alternatively, the TDGSS\_KRB5\_KRB5LIB environment variable may be set to the location of the library if you wish to avoid modifying the TeraGSS configuration. Absolute paths are required in both the <RequiredLibraryPath> element and the environment variable. See [Reconfiguring TDGSS for a Non-Standard Installation of Kerberos](#) for a non-standard installation of Kerberos.



# Password Restricted Words

The following lists restricted words that you can use to control the content of user passwords. Changes to the password restricted words list can occur when:

- Teradata Vantage adds new words to the list as part of a new software release.
- An administrator adds custom restricted words, or deletes existing restricted words.

Passwords are not subject to these restrictions unless you enable the PasswordRestrictedWords option. For additional information about use of password restricted words, see [PasswordRestrictWords](#).

## Default Restricted Words

The system provides approximately 2400 default restricted words, in two categories:

- Frequently used words (approximately 2000 words)
- Common names (approximately 400 words)

---

### Note:

Password restricted words are not case-sensitive

---

## Frequently Used Words

A				
ability	advantage	allowed	any	aspects
able	affairs	almost	anyone	assignment
about	afraid	alone	anything	assistance
academic	after	along	apart	associated
accept	afternoon	already	apartment	association
accepted	addition	also	apparent	assume
according	additional	although	apparently	assumed
account	address	always	appeal	atmosphere
achieved	adequate	america	appear	attack
achievement	again	american	appears	attempt
across	against	americans	application	attention
act	age	among	appearance	attitude

acting	agencies	amount	appeared	attorney
action	agency	analysis	applied	audience
actions	ago	ancient	arrived	authority
active	agreed	and	art	available
activities	agreement	animal	article	average
activity	ahead	animals	artist	avoid
actual	aid	announced	artists	aware
actually	air	annual	arts	away
add	aircraft	anode	aside	
added	alive	another	ask	
administration	all	answer	asked	
advance	allow	answered	asking	
<b>B</b>				
baby	became	benefit	boat	british
back	because	berlin	bodies	broad
background	become	beside	body	broke
bad	becomes	besides	book	broken
balance	becoming	best	books	brother
ball	bed	better	born	brought
bank	been	between	boston	brown
bar	before	beyond	both	budget
base	began	bible	bottle	build
baseball	begin	big	bottom	building
based	beginning	bill	bought	buildings
basic	begins	billion	box	built
basis	behavior	birds	boy	business
battle	behind	birth	boys	busy
bay	being	bit	break	but
beach	belief	black	bridge	buy
bear	believe	block	brief	

beat	believed	blood	bright	
beautiful	below	blue	bring	
beauty	beneath	board	britain	
<b>C</b>				
california	central	city	communism	contemporary
call	century	civil	communist	continue
called	certain	claim	community	continued
calls	certainly	claims	companies	continuing
came	chair	class	company	contract
camp	chairman	classes	compared	contrast
campaign	chance	clay	competition	control
can	change	clean	complete	cool
cannot	changed	clear	completed	corner
capable	changes	clearly	completely	corporation
capacity	chapter	close	completion	corps
capital	character	closed	complex	cost
captain	characteristic	closely	components	costs
car	charge	closer	concept	could
care	charged	clothes	concern	council
career	charles	club	concerned	countries
careful	check	coast	concerning	country
carefully	chemical	coffee	conclusion	county
carried	chicago	cold	condition	couple
carry	chief	collection	conditions	course
carrying	child	college	conduct	courses
cars	children	color	conducted	court
case	china	column	conference	cover
cases	chinese	combination	confidence	covered
catholic	choice	come	congo	created
cattle	chosen	comes	congress	credit

caught	christ	coming	connection	crisis
cause	christian	command	consider	critical
causes	churches	commercial	considered	cultural
caused	church	commerce	considerable	cross
cell	circle	commission	constant	culture
cells	circumstances	committee	construction	current
cent	cities	common	contact	cut
center	citizens	communication	contained	cutting
<b>D</b>				
daily	degree	device	distance	dramatic
dallas	degree	dictionary	distribution	draw
dance	demand	did	district	drawn
danger	demands	die	divided	dream
dark	democratic	died	division	dress
data	department	difference	doctor	drew
date	described	differences	does	drink
daughter	design	different	dog	drive
day	designed	difficult	dogs	drop
days	desire	difficulty	doing	dropped
dead	desk	dinner	dollars	drove
deal	despite	direct	domestic	dry
death	detail	directed	dominant	due
december	details	direction	done	during
decided	determine	directly	door	dust
decision	determined	director	double	duty
declared	develop	discovered	doubt	
deep	developed	discussed	down	
defence	development	discussion	dr	
<b>E</b>				
each	eight	enough	even	exist

earlier	either	enter	evening	existence
early	election	entered	event	existing
earth	electric	entire	events	expect
easily	electronic	entirely	ever	expected
east	elements	entitled	every	experience
easy	else	entrance	everybody	experiment
eat	emotional	equal	everyone	experiments
economic	emphasis	equally	everything	explain
economy	employees	equipment	evidence	explained
edge	empty	escape	evident	expressed
editor	end	especially	evil	expression
education	ended	essential	exactly	extended
educational	ends	establish	example	extent
effect	enemy	established	excellent	extreme
effective	energy	estimated	except	eye
effects	england	etc	exchange	eyes
effort	english	europe	executive	
efforts	enjoyed	european	exercise	
<b>F</b>				
face	father	file	follow	frame
faces	favor	filled	followed	france
facilities	fear	film	following	frank
fact	features	final	follows	free
factor	federal	finally	food	freedom
factors	feed	financial	foot	french
facts	feel	find	for	frequently
faculty	feeling	finds	force	fresh
failed	feelings	fine	forced	friday
failure	feet	fingers	forces	friend
fair	fell	finished	foreign	friendly

fairly	fellow	fire	forest	friends
faith	felt	firm	form	from
fail	few	firms	formed	front
familiar	field	first	former	full
families	fields	fiscal	forms	fully
family	fifteen	fit	formula	function
famous	fifty	five	fort	fund
far	fig	fixed	forth	funds
farm	fight	flat	forward	further
fashion	fighting	floor	found	future
fast	figure	flow	four	
fat	figures	flowers	fourth	
<b>G</b>				
gain	germany	goal	granted	grounds
game	get	god	gray	group
games	gets	goes	great	groups
garden	getting	going	greater	grow
gas	girl	gone	greatest	growing
gave	girls	good	greatly	growth
general	give	goods	greek	guest
generally	given	got	green	guests
generation	gives	government	grew	gun
george	giving	governments	gross	
german	glass	governor	ground	
<b>H</b>				
had	have	hell	himself	hospital
hair	having	help	his	hot
half	head	helped	historical	hotel
hall	headed	hence	history	hour
hand	headquarters	henry	hit	hours

hands	health	her	hold	house
hanover	hear	here	holding	houses
happen	heard	herself	hole	housing
happened	hearing	high	home	how
happy	heart	higher	homes	however
hard	heat	highest	honor	human
hardly	heavily	highly	hope	hundred
has	heavy	hill	horse	hung
hat	held	him	horses	husband
<b>I</b>				
idea	improved	independence	informed	interior
ideal	inch	independent	initial	internal
ideas	inches	index	inside	international
image	include	india	instance	into
imagination	included	indicate	instead	involved
imagine	including	indicated	institutions	island
immediate	income	individual	intellectual	issue
immediately	increase	individuals	intensity	issues
impact	increased	industrial	interest	items
importance	increases	industry	interested	its
important	increasing	influence	interesting	itself
impossible	indeed	information	interests	
<b>J</b>				
jack	job	joined	judgement	just
james	jobs	jones	july	justice
jazz	joe	joseph	june	
jesus	john	jr	junior	
jewish	join	judge	jury	
<b>K</b>				
keep	key	killed	knew	known

keeping	kruschev	kind	knife	knows
kennedy	kid	king	know	
kept	kill	kitchen	knowledge	
<b>L</b>				
labor	lead	length	line	look
lack	leader	less	lines	looked
lady	leaders	let	lips	looking
laid	leadership	letter	list	looks
land	leading	letters	literary	lord
language	league	level	literature	lose
laos	learn	levels	little	loss
large	learned	lewis	live	lost
largely	learning	liberal	lived	lot
larger	least	library	lives	louis
last	leave	lie	living	love
late	leaving	life	local	loved
later	led	light	located	low
latter	left	like	location	lower
law	leg	liked	london	
laws	legal	likely	long	
lay	legs	limited	longer	
<b>M</b>				
machine	married	meeting	miles	moon
machinery	martin	member	military	moral
made	mary	members	million	more
main	mass	membership	mind	mother
maintain	master	memory	minds	motion
maintenance	material	men	minor	motor
major	materials	mentioned	minute	mouth
majority	matter	mercier	minutes	move



make	matters	merely	miss	moved
makes	maximum	message	mission	movement
making	may	met	model	moving
man	maybe	metal	modern	much
management	mean	method	moment	murder
manager	meaning	methods	monday	music
manner	means	middle	money	musical
many	meant	might	month	must
march	measure	mike	months	my
mark	measures	mile	moreover	myself
marked	medical	mine	morgan	
market	meeting	minimum	morning	
marriage	meet	minister	most	
<b>N</b>				
name	near	neither	no	notes
named	nearly	never	nobody	nothing
names	necessary	nevertheless	none	notice
narrow	neck	new	nor	novel
nation	need	news	normal	november
national	needed	newspaper	north	now
nations	needs	next	nose	nuclear
natural	negro	nice	not	number
naturally	negroes	night	note	numbers
nature	neighborhood	nine	noted	
<b>O</b>				
object	offered	one	opportunity	other
objective	office	ones	opposite	others
objects	officer	only	orchestra	otherwise
observed	officers	onto	order	ought
obtained	official	open	ordered	our

obvious	officials	opened	orders	ourselves
obviously	often	opening	ordinary	out
occasion	oil	operating	organization	outside
occurred	old	operation	organizations	over
off	older	operations	organized	own
offer	once	opinion	original	
<b>P</b>				
page	period	plays	practical	production
paid	permit	please	practice	products
pain	permitted	pleasure	prepared	professional
painting	person	plenty	presence	professor
pale	personal	plus	present	program
palmer	personnel	poems	presented	programs
paper	persons	poet	president	progress
parents	phase	poetry	press	project
paris	phil	point	pressure	projects
park	philosophy	pointed	pretty	proper
parker	physical	points	prevent	properties
part	pick	police	previous	property
particular	picked	policies	previously	propose
particularly	picture	policy	price	protection
parties	pictures	political	prices	proved
parts	piece	politics	primarily	provide
party	pieces	pool	primary	provided
pass	place	poor	principal	providence
passed	placed	popular	principle	provides
passing	places	population	principles	providing
past	plan	portion	private	public
patient	plane	position	probably	published
pattern	planned	positive	problem	pulled

pay	planning	possibility	problems	pure
peace	plans	possible	procedure	purpose
people	plant	possibly	procedures	purposes
per	plants	post	process	put
percent	platform	potential	processes	
perfect	play	power	produce	
performance	played	powerful	produced	
perhaps	playing	powers	product	
<b>Q</b>				
quality	questions	quickly	quite	
question	quick	quiet		
<b>R</b>				
race	really	relation	requirements	rise
radiation	reason	relations	requires	rising
radio	reasonable	relationship	research	river
railroad	reasons	relatively	resolution	road
rain	receive	relief	resources	roads
raised	received	religion	respect	robert
ran	recent	religious	response	rock
range	recently	remain	responsibility	role
rapidly	recognize	remained	responsible	roman
rate	recognized	remains	rest	rome
rates	record	remember	result	roof
rather	records	remembered	results	room
reach	red	remove	return	rooms
reached	reduce	removed	returned	rose
reaction	reduced	repeated	review	round
read	reference	replied	revolution	rule
reading	refused	report	rhode	rules
ready	regard	reported	rich	run

real	regarded	reports	richard	running
reality	region	represented	rifle	runs
realize	regular	require	right	russia
realized	related	required	rights	russian
<b>S</b>				
safe	simply	son	stands	structure
said	since	song	stared	struggle
sales	single	songs	start	student
sam	sir	soon	started	students
same	sit	sort	starting	studied
sample	site	sought	state	studies
san	sitting	sound	stated	study
sat	situation	source	statement	style
saturday	six	sources	statements	subject
save	size	south	states	subjects
shelter	sky	southern	station	substantial
ship	sleep	soviet	stations	success
shook	slightly	space	status	successful
shop	slow	speak	stay	such
shore	slowly	speaking	stayed	suddenly
short	small	special	step	sufficient
shot	smaller	specific	steps	suggested
should	smile	speech	still	summer
shoulder	smiled	speed	stock	sun
show	snow	spent	stone	sunday
showed	social	spirit	stood	supply
showing	society	spiritual	stop	support
shown	soft	spite	stopped	suppose
shows	soldiers	spoke	store	supposed
side	solid	spot	stories	sure

sides	solution	spread	story	surface
sight	some	spring	straight	surprised
sign	somebody	square	strange	sweet
signal	somehow	staff	street	system
significance	someone	stage	streets	systems
significant	something	stand	strength	
signs	sometimes	standard	stress	
similar	somewhat	standards	strong	
simple	somewhere	standing	struck	
<b>T</b>				
table	ten	thin	told	trees
take	tension	thing	tom	trial
taken	term	things	tomorrow	tried
takes	terms	think	tone	trip
taking	test	thinking	too	trouble
talk	tests	third	took	truck
talked	texas	thirty	top	true
talking	text	this	total	truly
task	than	thomas	touch	truth
taste	that	those	toward	try
tax	the	though	towards	trying
teacher	their	thought	town	tuesday
teachers	then	thousand	trade	turn
teaching	theme	three	tradition	turned
team	themselves	through	traditional	turning
technical	then	throughout	traffic	twenty
technique	theory	thus	train	twice
techniques	there	time	training	two
teeth	therefore	times	travel	type
telephone	these	title	treated	types

tell	they	today	treatment	typical
temperature	thick	together	tree	
<b>U</b>				
ultimate	union	universe	upper	usual
uncle	unique	university	use	usually
under	unit	unless	used	
understand	united	until	useful	
understanding	units	unusual	uses	
understood	unity	upon	using	
<b>V</b>				
valley	various	view	visit	volume
value	vast	village	vital	vote
values	very	virginia	vocation	
variety	victory	vision	voice	
<b>W</b>				
wage	watching	when	wilson	wore
wait	water	where	win	work
waited	way	whether	wind	worked
waiting	ways	which	window	workers
walk	weapons	while	wine	working
walked	weather	white	winter	works
wall	week	who	wish	world
walls	weeks	whole	with	worry
want	weight	whom	within	worth
wanted	well	whose	without	would
wants	went	why	woman	write
war	were	wide	women	writer
warm	west	wife	won	writers
was	western	wild	wonder	writing
washington	what	will	wondered	written

watch	whatever	william	word	wrong
watched	wheel	willing	words	wrote
<b>Y</b>				
your	yards	yes	york	
yourself	year	yesterday	you	
youth	years	yet	young	

## Frequently Used Names

<b>A</b>				
adam	alexander	alyssa	angela	arthur
adrian	alexandra	amanda	anita	ashley
alan	alexandria	amber	ann	austin
albert	alexis	amy	anthony	autumn
alejandro	alicia	andrea	antonio	
<b>B</b>				
bailey	bianca	bradley	brian	bryan
barb	billy	brandon	brittany	bryce
barry	blake	breanna	brittney	
ben	bob	brenda	brooke	
beth	bonnie	brett	bruce	
<b>C</b>				
caitlin	catherine	cheyenne	colleen	crystal
caleb	cathy	chloe	connie	curtis
cameron	chad	chris	connor	cynthia
carl	charles	cindy	corey	
carol	chase	claire	cory	
casey	chelsea	cody	courtney	
cassandra	cheryl	cole	craig	
<b>D</b>				
dakota	darrell	dean	derek	dustin

dale	darren	debbie	destiny	dylan
dalton	daryl	deborah	devin	
dana	dave	debra	diana	
dan	david	denise	don	
darlene	dawn	dennis	doug	
<b>E</b>				
edward	elizabeth	emma	erik	ethan
elijah	emily	eric	erin	evan
<b>F</b>				
faith	frank			
<b>G</b>				
gabriel	garrett	george	gina	grace
gabriella	gary	gerald	glen	gregory
gabrielle	gavin			
<b>H</b>				
hailey	hannah	heather	holly	hunter
haley	harold	henry		
<b>I</b>				
ian	isaac	isabel	isabella	isiah
<b>J</b>				
jack	janet	jeremy	john	julia
jacob	janice	jerry	jon	julian
jacqueline	jared	jesse	jordan	julie
jada	jasmine	jessica	jose	justin
jake	jason	jesus	joshua	
james	jay	jill	joyce	
jamie	jeff	jimmy	juan	
jane	jenn	joe	judy	
<b>K</b>				
kaitlyn	katheryn	keith	kimberly	



karen	kathy	kelly	kristen	
kate	katie	kelsey	kristin	
katherine	kayla	ken	kristina	
kathleen	kaylee	kevin	kyle	
<b>L</b>				
larry	lawrence	lindsay	lori	lynn
laura	leah	lindsey	lucas	
lauren	leslie	lisa	luis	
laurie	linda	logan	luke	
<b>M</b>				
mackenzie	maria	mary	melinda	miguel
madeline	mariah	mason	melissa	mike
madison	marissa	matt	mia	mittchell
makalya	mark	megan	michael	molly
marcus	martha	meghan	michele	monica
margaret	martin	melanie	michelle	morgan
<b>N</b>				
nancy	nathan	nicholas	nicole	noah
natalie	nathaniel			
<b>O</b>				
Olivia				
<b>P</b>				
paige	patricia	paul	penny	philip
pam	patrick	paula	peter	phillip
<b>R</b>				
rachel	rebecca	rich	rodney	russell
randall	regina	ricky	roger	ryan
randy	renee	robert	ron	
ray	rhonda	robin	roy	
<b>S</b>				

sabrina	scott	sharon	sherry	stephen
sam	sean	shawn	sierra	steve
sandra	sebastian	sheila	sophia	susan
sara	seth	shelby	spencer	suzanne
sarah	shane	shelly	stacy	sydney
savannah	shannon	sherri	stephanie	
<b>T</b>				
tammy	terri	tim	tracy	tyler
tanner	terry	tina	travis	
tara	theresa	todd	trevor	
taylor	thomas	tony	trinity	
teresa	tiffany	tracey	troy	
<b>V</b>				
valerie	veronica	victor	victoria	vincent
vanessa				
<b>W</b>				
walter	wayne	wendy	whitney	will
wanda				
<b>Z</b>				
zachary	zoe			

# Additional Information

## Teradata Links

Link	Description
<a href="https://docs.teradata.com/">https://docs.teradata.com/</a>	Search Teradata Documentation, customize content to your needs, and download PDFs. Customers: Log in to access Orange Books.
<a href="https://support.teradata.com">https://support.teradata.com</a>	One-stop source for Teradata community support, software downloads, and product information. Log in for customer access to: <ul style="list-style-type: none"><li>• Community support</li><li>• Software updates</li><li>• Knowledge articles</li></ul>
<a href="https://www.teradata.com/University/Overview">https://www.teradata.com/University/Overview</a>	Teradata education network
<a href="https://support.teradata.com/community">https://support.teradata.com/community</a>	Link to Teradata community