MICRO FOCUS

# Dimensions CM

## Architecture and Optimization Guide

Product version: 14.5.3

Last updated: July 21, 2021

# Table of Contents


*Chapter 1* **Dimensions CM Architecture . . . . . 3**

Overview . . . . . 4
- Presentation Tier . . . . . 6
- Business Logic Tier . . . . . 8
- Database/Metadata Tier . . . . . 9
- File/Storage Tier . . . . . 10

Supported Platforms . . . . . 11

Network Protocols . . . . . 12
- Standard Dimensions Protocol . . . . . 12
- Secure Standard Dimensions Protocol . . . . . 12
- Dimensions CM Server HTTP Connector . . . . . 12

Security . . . . . 13
- Password Encryption . . . . . 13
- HTTPS . . . . . 13
- LDAP . . . . . 13

Authentication . . . . . 14
- Native Authentication . . . . . 14
- Single Sign On . . . . . 14
- Lightweight Directory Access Protocol . . . . . 15
- Pluggable Authentication Modules . . . . . 15
- Common Access Cards . . . . . 16

Access Control and Authorization . . . . . 16

Application Programming Interfaces . . . . . 17
- C/C++ . . . . . 17
- Java API . . . . . 17
- RESTful Web Services . . . . . 17
- Event Callout Interface . . . . . 17
- SOAP Web Services . . . . . 17
- Application Lifecycle Framework Events . . . . . 17
- Templating Language . . . . . 18

Server Configurations . . . . . 18
- Standalone Server . . . . . 18
- Multiple Servers . . . . . 18
- Distributed Environments . . . . . 20

Vertical Scaling . . . . . 21
- Scaling Processors . . . . . 21
- Scaling Memory . . . . . 21
- Scaling Single System Configurations . . . . . 22
- Network Hardware Performance . . . . . 22

Horizontal Scaling . . . . . 23
- Dynamic Load Balancing . . . . . 24
- Static Load Balancing . . . . . 24

Chapter 2 **Optimizing Dimensions CM . . . . . 25**
Optimizing Your Configuration . . . . . 26
Hardware Scaling Recommendations . . . . . 27
Improving WAN Performance . . . . . 28
File Transfer with PLCD and Library Cache . . . . . 28
Memory File Cache . . . . . 29
Automatic Library Cache Population . . . . . 29
Configuring PLCD on a Server . . . . . 30
Configuring PLCD on a Client . . . . . 30
Using Delta Compression . . . . . 32
Measuring Performance Benefits . . . . . 33
Replicating Repositories . . . . . 34
Versioned Repository Schema Caching . . . . . 35
Overview . . . . . 35
Configuring VRS . . . . . 35
Pre-Caching VRS Data . . . . . 36

Chapter 3 **Advanced Optimization Techniques . . . . . 39**
Using Single Privilege Checks . . . . . 40
Optimizing Performance Based on Bandwidth . . . . . 40
Calculating BDP . . . . . 40
Optimizing TCP/IP Buffer Values . . . . . 41
Optimizing the Web Client . . . . . 41
Optimizing Memory Setup for Tomcat . . . . . 41
Disabling ACKs and The Nagle Algorithm . . . . . 43
Disabling Delayed ACKs . . . . . 43
Disabling the Nagle Algorithm . . . . . 43
Optimizing the File Compression Levels . . . . . 45
File Compression Levels . . . . . 45
Setting Compression Levels . . . . . 46
Non-Encryption of Item Contents . . . . . 47
Optimizing AIX Performance . . . . . 47
Managing the Server Application Pool . . . . . 48
Modifying Server Configuration Symbols . . . . . 51
Modifying Configuration Symbols . . . . . 51
Configuration Symbols Reference . . . . . 52

Chapter 1

# Dimensions CM Architecture


| | |
|---|---|
| Overview | 4 |
| Supported Platforms | 11 |
| Network Protocols | 12 |
| Security | 13 |
| Authentication | 14 |
| Access Control and Authorization | 16 |
| Application Programming Interfaces | 17 |
| Templating Language | 18 |
| Server Configurations | 18 |
| Vertical Scaling | 21 |
| Horizontal Scaling | 23 |

# Overview

Dimensions CM has a multi-tiered cross-platform architecture with the following layers:

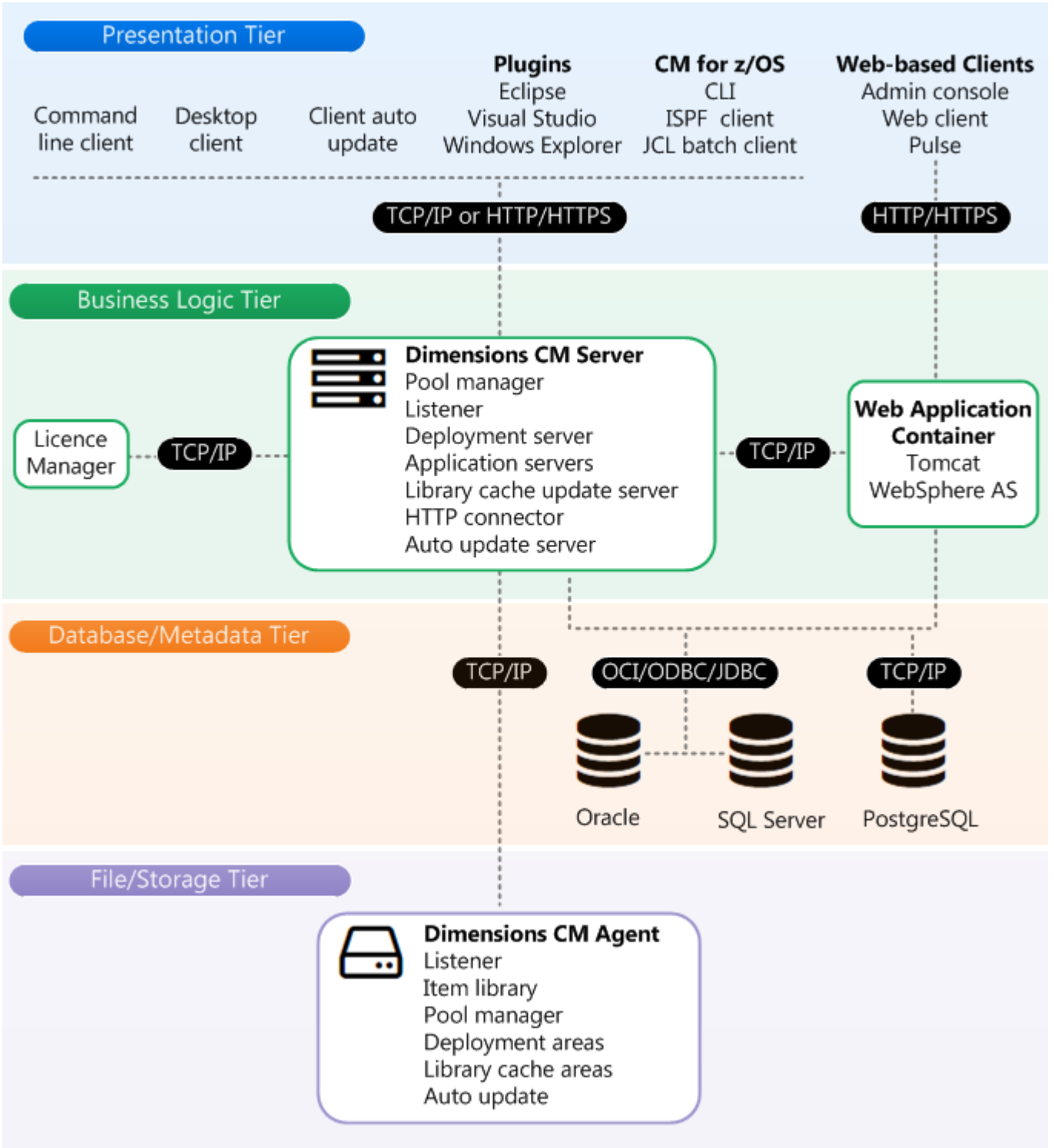
- Presentation tier
- Business logic tier
- Database/metadata tier
- File/storage tier

The components in each tier can be installed on a single machine or scaled horizontally across multiple machines.

A Dimensions CM repository is the full set of stored requests and files (or items) and includes:

- A database that stores requests and metadata.
- An item library that stores items for one or more item types in the database. The item library directory can be located on a different network node from the database.

The following diagram illustrates the Dimensions CM architecture:


Presentation Tier
Command line client
Desktop client
Client auto update
Plugins
Eclipse
Visual Studio
Windows Explorer
CM for z/OS
CLI
ISPF client
JCL batch client
Web-based Clients
Admin console
Web client
Pulse
TCP/IP or HTTP/HTTPS
HTTP/HTTPS
Business Logic Tier
Licence Manager
TCP/IP
Dimensions CM Server
Pool manager
Listener
Deployment server
Application servers
Library cache update server
HTTP connector
Auto update server
TCP/IP
Web Application Container
Tomcat
WebSphere AS
Database/Metadata Tier
TCP/IP
OCI/ODBC/JDBC
TCP/IP
Oracle
SQL Server
PostgreSQL
File/Storage Tier
Dimensions CM Agent
Listener
Item library
Pool manager
Deployment areas
Library cache areas
Auto update

# Presentation Tier

### *Desktop Client*

The desktop client is a Windows based application that provides full access to all CM functionality. This client uses TCP/IP to communicate over the network to the business logic tier. Other Windows application include the Project Merge Tool and the synchronize wizard (for file operations and merges). For information see the online help.

### *Web Client*

The web client is a web-based interface to CM and provides access to a limited set of CM functionality. This client uses HTTP/HTTPS to communicate to a Web Application Container in the business logic tier. HTTP communication enables deployment of these clients across a wide variety of platforms and network topologies. Applets are used to interact with the file system.

If you do not want to use applets, you can download Web Client Tools that allow native file operations. For information see the online help.

### *Micro Focus PulseUno*

Micro Focus PulseUno is a web-based user interface that enables development teams to:

- Continually build and inspect the health and quality of code using plugins such as dependency checkers and static analysis. Teams can use this information to help decide when changes are ready to be merged, deployed, and released.
- Use PulseUno vaults to secure collections of packages such as binaries, artifacts, and meta data. Vaults provide a central library for your packages and integrate with build and development pipelines.
- Visualize team activity.
- Explore the codebase.
- Use pull requests to review, evaluate, and merge a set of defined changes. Pull requests are a collaborative process that allows teams to review proposed code changes, add review comments, approve changes, and merge changes into a mainline.
- Collaborate on development projects using requests, backlogs, and iterations, to plan, track, and execute work.

For details, see the PulseUno online help.

### *Command-Line Interface*

The command line interface (*dmcli*) is a native component that runs on UNIX and Windows and enables access to a flexible command line. This client uses TCP/IP to communicate to the business logic tier. For information see the *Command-Line Reference*.

### *Developer Command Line Interface*

The Developer Command-Line provides a set of functions that enable you to manage streams from a command-line. For information see the *Command-Line Reference.*

### Administration Console

The administration console is a web-based application for administering and configuring CM. This client uses HTTP/HTTPS to communicate to a Web Application Container in the business logic tier. For information see the *Process Configuration Guide.*

### Dimensions for z/OS

Dimensions for z/OS enables mainframe hardware to participate in a CM network as a remote node and an item library server. You can use the following Dimensions for z/OS clients:

- ISPF: an interactive client running in the TSO/ISPF environment that supports most CM functions.
- A batch interface for processing CM commands.
- `dmcli`: the UNIX command-line client, available on USS.

For information see the *Dimensions for z/OS Guide.*

### Integrations

Dimensions CM supports integrations to many applications including:

- Eclipse
- Microsoft Visual Studio
- Micro Focus Connect
- Git
- Subversion
- Docker
- IBM Doors
- HP Quality Center
- SAP PowerBuilder

For a complete list of supported platforms and environments, see the Dimensions CM Support Matrix.

### Dimensions Build

Dimensions Build is a web-based application that enables you to manage build configurations and deployments. For information see the *Dimensions Build online help.*

# Business Logic Tier

Components in the business logic tier are server components responsible for carrying out CM operations on behalf of the clients, for example:

- Authenticating users
- Checking privileges and authorizations
- Enforcing process rules
- Maintaining metadata about objects under control (such as streams and items)

### *Dimensions CM Server*

The Dimensions CM server is where the main operations are performed. This component is implemented using a proprietary C/C++ application management framework that provides high scalability and performance.

#### Listener

The *listener* component (`dmlsnr.exe`) monitors the health of the server and re-starts the server processes in the event of a failure. The listener starts a *pool manager* process (`dmpool.exe`) that manages a pool of *application servers* (see below). When a user is actively using a CM client they are allocated a process from the pool. When the user is idle the process returns to the pool and is available for other users. You can configure:

- The initial, minimum, and maximum size of the pool.
- The period of time that a client must be inactive for their application server to return to the pool.

This pooling architecture allows high scalability as many users can be connected to CM but only active users consume system resources.

The pool manager process usually runs as the following privileged user account:

- **UNIX:** `root`
- **Windows:** `SYSTEM`

The pool manager runs as a privileged user account so that it can:

- Authenticate local OS user credentials (via native OS authentication).
- Run processes as the local OS users (to access the file system on behalf of these users).

You can change the user who runs the pool manager process to a non-privileged user account if you are only using LDAP or SSO authentication and do not require access to the file system on behalf of other local OS users.

#### Application Server

An *application server* process (`dmappsrv.exe`) is where the main business logic of CM is performed. Each process connects to the database to perform metadata queries and updates and communicates with the client. All application server processes are pooled. When a user retrieves file content for an item revision, the application server communicate via TCP/IP to the file/storage tier to retrieve the content from the file system. Application server processes are started by the pool manager and run using a local non-privileged user account.

### *Web Application Container*

The Web Application Container is a J2EE servlet container (such as Apache Tomcat) that handles requests from the administration console, web client, PulseUno, and Web Services clients.

The administration console and web client use Java, HTML, JavaScript, and JSP (Java Server Pages) and their implementation follows a model 2 MVC (Model-View-Controller) pattern to separate presentation, logic, and content.

Most of the business logic is implemented by the Dimensions CM server. The servlet code does not access the database directly, rather it communicates over TCP/IP to the server to perform operations, query data, and store/retrieve file content.

When CM is installed it is deployed and configured to run under Apache Tomcat but it can also run under other supported Web Application Containers and the roles of the web server and servlet container can be separated. For example, Microsoft IIS (Internet Information Services) can be used as the web server and Tomcat for the servlet container. By default, the web applications are hosted by Tomcat on TCP/IP port number 8080 but this is configurable.

### *License Manager (SLM)*

The license manager is an extension of Flexera Software FlexLM. It supports both concurrent and named user licensing models. Micro Focus also provides a Java Swing based GUI to perform license administration activities.

When a connection is made from a client to a CM server, the pool manager process calls to the license manager to obtain a license for the client. When the user exits a client or a session time out occurs, the license is returned. Communication between the license manager and the CM server is via TCP/IP using FlexLM APIs and protocols.

## Database/Metadata Tier

Dimensions CM is highly configurable with a rich set of relationships and properties. To enable high performance and scalability, CM uses relational database technology to store and query metadata.

A single server can connect to multiple database instances, and an instance can contain multiple Dimensions CM base databases. For example, one base database is used for production work and another for testing. Each base database has its own schema and there is a single common schema (`PCMS_SYS`) that stores some metadata.

# File/Storage Tier

Dimensions CM securely manages the content of assets. The file storage and retrieval tasks are performed using the agent technology described below.

### *Item Library*

When assets are checked into CM, their content (the versions of source code, documents etc) is stored in a secure location called an item library. The machine hosting the item library must have a Dimensions CM server or agent installed on it. Different item types can be placed in different item library locations, for example:

- Separate directories on the same machine.
- Separate physical machines that host item libraries for different item types.

When an item library is accessed, the *library server* process (`dmlibsrv.exe`) runs as the following user:

- **Windows:** `SYSTEM`
- **UNIX** (when the item library is on a different machine to the server): `root`

You can also configure CM to specify a non-privileged user account to run the library server process and to access item libraries that are local to the server directly from application server processes. See the *System Administration Guide* for information about using the following configuration symbols:

- `DM_LIBRARY_USER_USER_<product id>_<item type> <username>`
- `DM_DIRECT_LOCAL_LIBRARY_ACCESS`

You can use the following methods in an item library to store the content of files:

- Flat files

  One file for every revision of an item held in CM. This is referred to as a normal item library. This is the recommended method for storing file content as it offers the highest performance and lowest overhead.

- Delta storage

  One file per item that contains all revisions of the item. Only the differences between each revision are stored. This method is useful if disk space is limited.

- Compressed

  Content is stored in flat, compressed files. This type of item library is referred to as a normal, compressed item library. Some CPU overhead may occur when accessing and uncompressing data.

### Remote Work and Deployment Areas

You can use remote work areas in addition to local ones.For example, a user is running the desktop client but is developing software on a mainframe therefore assets need to accessed from partitioned datasets (PDS) on a mainframe MVS file system. A Dimensions CM agent must be installed on the mainframe so that CM can interact with it.

When a remote work area is accessed, the pool manager on the CM agent node attempts to login using the credentials you used to login to the server. If the log in fails, the server prompts the user for credentials to access that work area. After a user has entered their credentials, they are not prompted again unless they logout of their client or their session expires.

When a deployment area is accessed, if the CM administrator chose to store credentials for that area then those credentials are used. If they were not stored the user is prompted.

### Item Library Cache

Software development teams are often geographically dispersed, with remote users connecting to a main site across a Virtual Private Network (VPN) utilizing a Wide Area Network (WAN). However, VPN connections over a WAN can have low bandwidth and suffer from a high latency. A single application message may take a long time to travel over a slow connection. If the protocol utilized by the users' applications at a remote location is 'chatty' the remote users typically experience more delays and slower response times. To address the problem of WAN latency, CM uses a library cache that improves performance by caching file contents on a network node that is geographically close to the remote users. When users update files, they are transferred from the library cache not the CM repository, reducing transfer times by eliminating the geographical distance and resulting network latency. If a project has many large files it is more likely to benefit from using library cache areas.

Personal library cache directories (PLCD) and delta compression on file transfers can also provide significant performance improvements for geographically distributed development teams.

For information see page 28 and the *System Administration Guide*.

# Supported Platforms

For a complete list of supported platforms and environments, see the Dimensions CM Support Matrix.

The Support Matrix includes details about:

- Operating systems
- Databases
- Web browsers and servers

For details about integrations with third-party tools and Micro Focus products, see the Dimensions CM Integrations Matrix.

# Network Protocols

## Standard Dimensions Protocol

Dimensions CM uses the Standard Dimensions Protocol (SDP) to communicate to the different components of the system. SDP works as follows:

- Most server components communicate using TCP/IP.
- Messages sent over such connections are RPC (Remote Procedure Call) style.
- The caller requests a particular function and sends a number of input parameters.
- The other end responds with output parameters.

By default, server and agent listen on TCP/IP port 671 though this is configurable. The TCP/IP connection is initiated from the client to the server. If you are configuring a firewall on your network you must allow incoming connections on that port to the server, and also from the server to agents (if required).

When an RPC request or response is large enough, it is automatically compressed by the server to provide low-bandwidth delivery of data. When transferring files, content is also automatically compressed. You can configure the level of compression in the administration console. When transferring many files, CM reads the file content from the file/storage tier to a memory buffer in the server, compresses the data, and delivers the compressed data buffer to the client. Files are only transferred if they have changed since the client last obtained them. When updating local files to newer versions, delta compression is utilized to further reduce the amount of data to be transferred. This optimization and compression ensures that CM performs well over a slow WAN.

## Secure Standard Dimensions Protocol

Secure Standard Dimensions Protocol (SSDP) and is identical to SDP described above. All data sent over the wire is Secure Socket Layer (SSL) encrypted. If your network traffic travels over a public network link, Micro Focus recommends you configure CM to use SSDP. Encryption is provided using TLS version 3.1 with the following cipher suite:

```
TLS_DHE_RSA_WITH_3DES_EDE_CBC_SHA
```

The encryption is performed using the OpenSSL API (http://www.openssl.org). Diffie-Hellman key agreement using ephemeral keying (key exchange via a temporary key) is employed to encrypt the traffic using a customer provided certificate. Micro Focus recommends a 1024 bit key, which can be configured by your CM administrator.

## Dimensions CM Server HTTP Connector

HTTP may be required for security reasons or for compatibility with network infrastructure such as proxies and firewalls. This is particularly true where the connection between a Dimensions CM server and clients goes over a public or virtual private network.

The Dimensions CM Server HTTP Connector allows clients to connect to a server using the HTTP or HTTPS network protocol instead of the default Standard Dimensions Protocol. The connector accepts network connections using HTTP/S and forwards the traffic to the server.

# Security

## Password Encryption

During authentication, the client sends the user's login credentials to the CM server and encrypts it using Blowfish encryption with a randomly generated 64-bit key. This encryption protects sensitive data from being discovered using a network packet sniffer. If further security is required, Micro Focus recommends enabling SSDP to encrypt all traffic.

Passwords for key user accounts are stored on the CM server on the file system (in a file located in `dfs/registry.dat` under your server installation root folder). To ensure that these passwords are secure on UNIX, they are owned and only readable by *root*. On Windows you can secure these files after the installation of CM. This information is also protected by encryption technology. The user names and passwords are encrypted using the AES128 encryption algorithm but you can use AES256 if required. Use the utility `dmpasswd` to administer the accounts stored in this file. The stored accounts include:

- **Dimensions pool owner**

  The *application server* processes running on the server are all owned by default by a single user account (the *pool owner*). The pool manager process (`dmpool.exe`) needs to know the password for this user to start these application server processes.

- **Database schema**

  Each CM base database is a database schema. The schema username and password are stored to allow CM to connect to the schema.

- **LDAP bind user**

  If you have configured CM to use Lightweight Directory Access Protocol (LDAP) and are using a specific user account for performing searches on the LDAP directory, their DN and password must be stored in this file.

- **Deployment area credentials**

  If the CM administrator has entered user credentials for a deployment area in the administration console, these are also stored using the same encryption scheme.

## HTTPS

The web client and administration console can be used over the industry standard HTTPS protocol to encrypt all communication between the browser and the Web Application Container using SSL/TLS.

## LDAP

When using LDAP to authenticate user credentials, CM supports *LDAP over an SSL tunnel* and *LDAP with StartTLS* extensions. Both of these security mechanisms encrypt the LDAP communication using strong SSL/TLS encryption. Not all LDAP servers support both secure protocols.

# Authentication

When a user logs in to a CM client they must provide a user name and password or use a smart card. The server can be configured to validate these credentials using different authentication systems.

## Native Authentication

This is the default configuration for authentication and uses the server's native operating system authentication mechanisms to validate the users credentials.

- UNIX

  Uses the POSIX standard `getpwnam`, `getspnam`, and `crypt` functions to encrypt the password provided by the client and compare it to the value stored in the system's password file. If the values match the user is successfully authenticated.

- Windows

  Queries the user account information using the `NetGetUserInfo` function provided by the Microsoft Platform SDK. The domain controller that is queried depends on how the system is configured.

  - If a domain name was specified as part of the user name (`mydomain\myuser`), CM queries the domain controller for that domain and calls `NetGetUserInfo` against that domain controller.
  - If the server is configured to authenticate against a given domain name (using the `DM_LOGON_DOMAIN` parameter), CM looks up the domain controller for that specific domain and calls `NetGetUserInfo` against that domain controller.
  - If no domain was specified in the user name and no domain name has been configured, CM queries the primary domain controller for the server.

  After obtaining the account information, CM verifies if the user's password has expired and if necessary allows them to change it. CM then calls the Microsoft Win32 API function `LogonUser` and passes it the default log on provider and a log on type of `LOGON32_LOGON_NETWORK`. If the `LogonUser` function succeeds, the user's credentials are validated and they can access CM.

- z/OS

  On a Dimensions z/OS agent machine the POSIX `getpwnam` and `crypt` functions are used the same as on UNIX. These system calls then authenticate users against the chosen security system (RACF, TopSecret etc).

## Single Sign On

Dimensions CM supports Single Sign On. For details see the UNIX or Windows *Installation Guide,* the *System Administration Guide*, and the online help.

## Lightweight Directory Access Protocol

A CM server can be configured to authenticate users with a Lightweight Directory Access Protocol (LDAP) server, for example, Microsoft Active Directory Server or OpenLDAP directory server. Enterprise customers use these directory servers to centralize their user account information. Dimensions uses the OpenLDAP API to communicate between the pool manager and the LDAP server. To authenticate using LDAP, CM identifies the DN (Distinguished Name) for the user and "bind" to the LDAP directory server using that name and the password provided.

LDAP is a highly configurable directory server and CM provides the following ways to identify a user's DN:

- Search for a matching DN

  This is typically the most common way of authenticating using LDAP. The CM server searches the LDAP directory for a DN that has a particular attribute value and then binds using that DN.

  You need to configure a "bind user" which is the DN of a user who has authority to search the directory and register their password using the *dmpasswd* utility. CM "binds" initially as that user and performs a search (from a given point in the LDAP directory tree) to find a DN where a specific attribute has the value set to the name of the user logging into CM. If a match is found CM "binds" using that DN. If binding is successful the user is authenticated successfully and can access CM.

- Search anonymously for a matching DN

  If you do not configure a bind user but configure CM to perform an LDAP search, the search is performed anonymously. It functions as described above but you need to enable anonymous searches on your LDAP server.

- Construct a specific DN

  Allows you to configure CM to construct a specific DN instead of searching to find one. Configure the "base" of the DN and an attribute to use as the RDN (Relative Distinguished Name). CM forms a DN by specifying the RDN, followed by the user's name, and then the "base". CM tries to "bind" to that DN using the password provided when the user logged in. If binding is successful the user is authenticated successfully and can access CM.

For more details about configuring LDAP see the *System Administration Guide*.

## Pluggable Authentication Modules

On Solaris and Linux, CM also supports authentication using Pluggable Authentication Modules (PAM) which allows many common authentication systems to be supported. The CM PAM implementation only supports authentication modules that require user name and password (the PAM LDAP module). Modules that require more data to authenticate the user are not supported.

When PAM authentication is configured the pool manager calls PAM APIs to authenticate the user, passing the user name and password when required. PAM calls the authentication module that has been configured for the Dimensions service in `/etc/pam.conf`. CM uses a PAM service called `dimensions_cm` that is configurable. If the PAM conversation succeeds the user is authenticated successfully and can access CM.

For more details about configuring PAM see the *System Administration Guide*.

## Common Access Cards

Dimensions CM supports user authentication using the United States Department of Defense (DoD) Common Access Cards (CAC). Users present their CAC card to the card reader, select a certificate to use to identify themselves, and enter their card PIN number. The validation of the user is performed as follows:

1. The Dimensions CM server sends a cryptographic nonce to the client, which is then securely signed using the private key held on the CAC card via the PKCS#11 API.
2. The signed nonce is sent to the Dimensions CM server along with the user's public x509 certificate.
3. The server verifies that the nonce was signed by the bearer of the public certificate (proving that the user has the correct CAC card).
4. The SSO server validates the certificate, for example, checks that it has not expired, that it is signed by a trusted Certificate Authority (CA), and is not included in a Certificate Revocation List (CRL).
5. The SSO server uses the subject name from the certificate to locate a matching LDAP user account. If the certificate is valid and trusted, comes from the bearer of the CAC card, and has a matching LDAP entry, then authentication is complete.

For more details about configuring CAC see the *System Administration Guide*.

# Access Control and Authorization

Access control and authorization features enable you to divide a product into areas and define roles and responsibilities for each area.

Every operation is controlled by a *privilege*. Each privilege has a set of *rules* which, if met, grant the user permission to carry out the operation. Rules are very flexible and enable administrators to control what each user can do based on the state of the object being operated on, the groups the user is a member of, and the roles that the user holds. Administrators can use additional process rules to add more control. For example, a user may have the *Relate Item to Request* privilege but cannot check-out files against a change request if it is in the closed phase of its lifecycle.

A *role* identifies users with similar responsibilities, for example, Developer or Reviewer. Roles are used in lifecycles to grant permissions to action objects. Roles are assigned to users according to the object's place in the design structure and the user's responsibilities.

Roles and privileges can also be used to control access depending on which *design part* an asset belongs to. A design part is a functional area or logical component of a system. For example, some source code may be owned by a design part called *GUI* and privileges are configured so that only users who hold the *Developer* role on that part can edit the code. If a user has the *Developer* role on a different part, for example *Database*, they can edit code owned by *Database* but not *GUI*.

For information about privileges, roles, and design parts see the *Process Configuration Guide*.

# Application Programming Interfaces

CM supports publicly available Application Programming Interfaces (API) that enable you to create your own applications that make use of CM functionality or to extend its functionality. For information see the *Developer's Reference.*

## C/C++

A set of documented C functions and structures are provided with examples and makefiles to allow you to write C/C++ applications that can query CM metadata and perform CM operations. This is a cross-platform client API that uses TCP/IP to communicate with the CM server.

## Java API

The Java API (`dmclient`) is a fully featured object-orientated API for accessing CM functionality and includes complete documentation and examples. This is a cross-platform API that uses TCP/IP to communicate with the CM server.

## RESTful Web Services

The Dimensions CM REST services API provides methods for listing and getting details of objects in CM, for example, listing requests in a user's inbox. For more information about the REST services provided and their endpoints, see the *Developer's Reference*.

## Event Callout Interface

The Dimensions CM Event Callout Interface gives access to a public function call that is invoked when certain commands are run. This function is called `userSuppliedFunction()` and is resolved in a shared library.

## SOAP Web Services

CM provides a set of SOAP Web Services to perform operations on items, requests, and baselines. The Document Literal Wrapped style of WSDL declaration is used and the WSDL is WS-I Basic 1.0 compliant.

## Application Lifecycle Framework Events

CM can generate Application Lifecycle Framework (ALF) events for a restricted set of operations. Each operation fires a single ALF event and in accordance with ALF guidelines these events are generated post operation. The data provided by each event is a reflection of the command that issued the ALF event.

## Templating Language

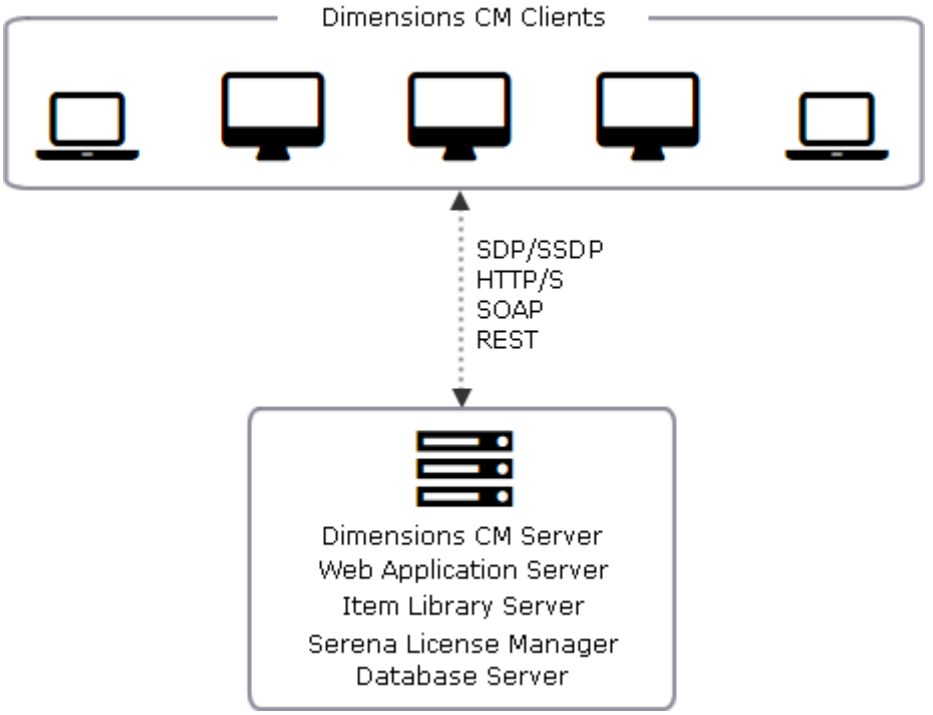
CM templates are used to execute builds, execute general commands through remote job execution, control deployments, and to construct the body of e-mail messages used for notifications. CM supports a common templating language and processor across Windows, UNIX, Linux, and z/OS (USS and MVS). The templating language enables you to customize the processing of templates on all Dimensions CM nodes.

# Server Configurations

A wide variety of configurations are supported by Dimensions CM. This section describes some common configurations.

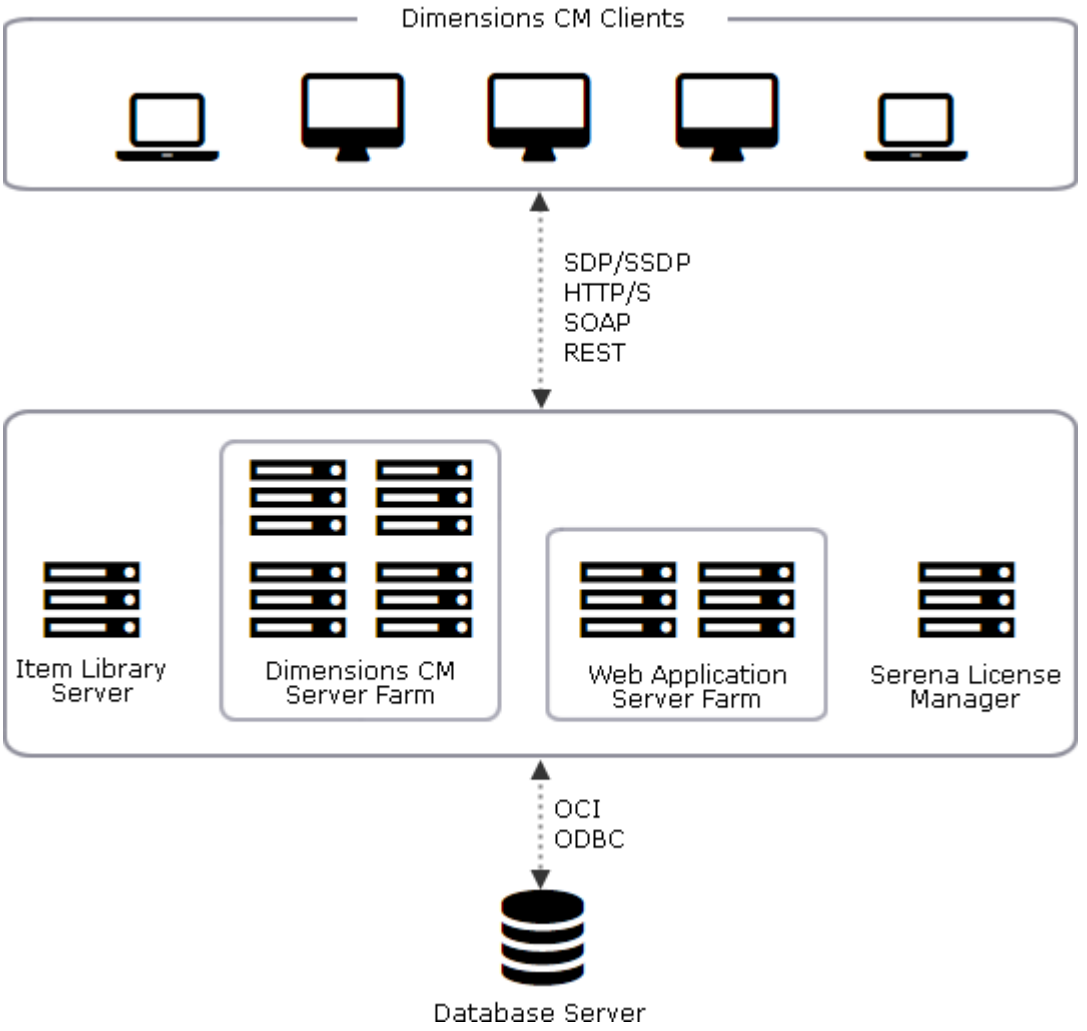
## Standalone Server

In a standalone server configuration, the components from the business logic, database and storage tiers are hosted on a single physical server machine. This configuration works well if you have a small user community or a powerful server machine.



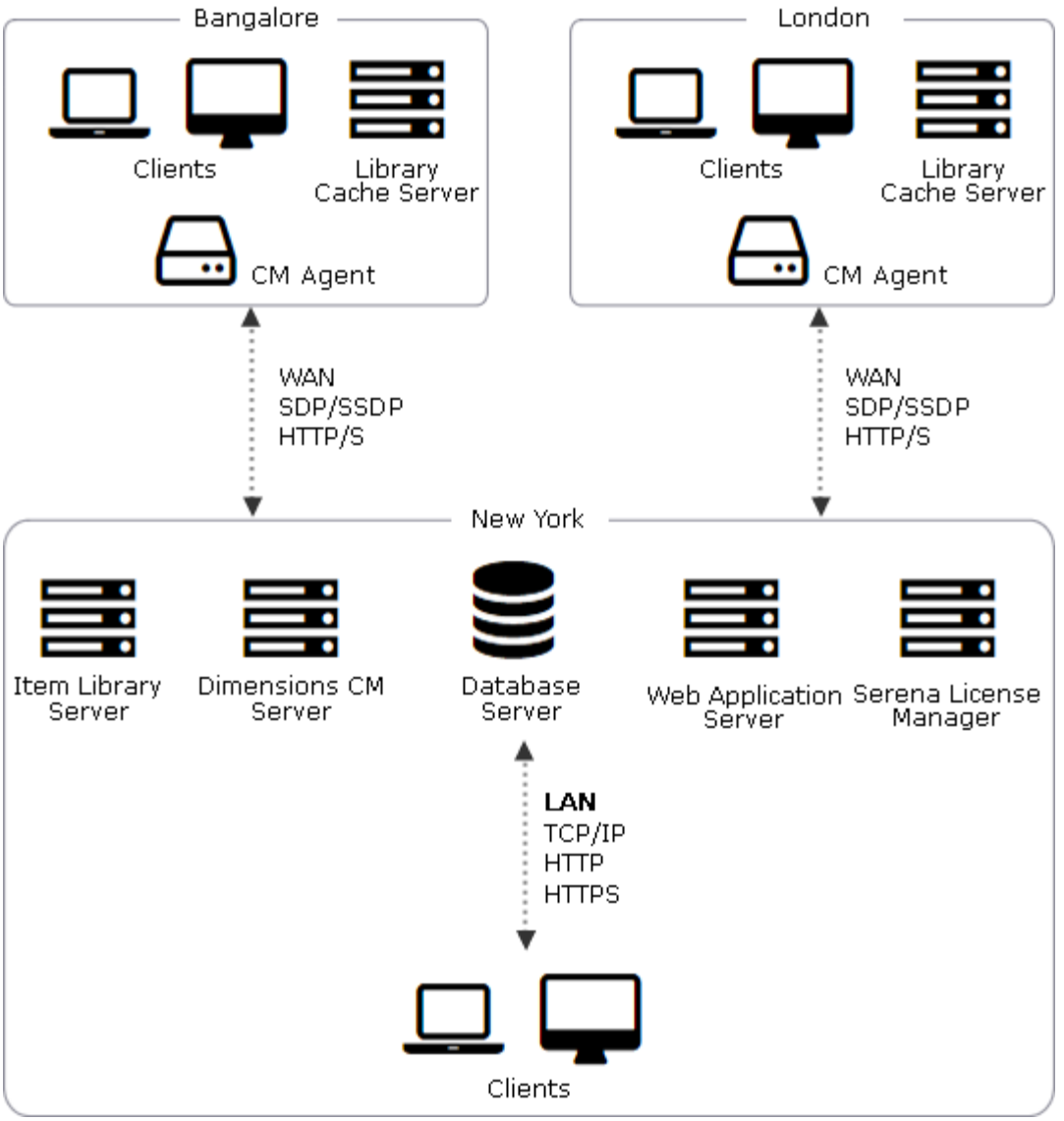

## Multiple Servers

In a multiple server configuration, the database and server are on separate physical machines. This spreads the CPU and RAM requirements of the server components across the machines and improves performance and scalability.


Dimensions CM Clients
SDP/SSDP
HTTP/S
SOAP
REST
Item Library Server
Dimensions CM Server Farm
Web Application Server Farm
Serena License Manager
OCI
ODBC
Database Server

# Distributed Environments

In a distributed configuration, CM users are dispersed across multiple sites. In the diagram below, users in Bangalore and London connect to a central Dimensions CM server located in New York. CM Agents are installed on the remote locations and act as item library caches.



# Vertical Scaling

Vertical scaling adds more power to your hardware to meet increasing performance need. This includes faster CPUs, multiple CPUs, more memory, faster network cards, or a combination of all of these.

## Scaling Processors

Depending on the operations being performed, a Dimensions CM server and database can be CPU intensive. When using multiprocessor or hyper-threading configurations, the server is able to use more system resources than under single processor configurations. In memory intensive situations, the server may even consume all available memory.

## Scaling Memory

There are multiple processes that require memory, check that you have sufficient RAM to enable scaling.

**NOTE** The memory recommendations for virtual machines are the same as for physical machines.

### *Dimensions CM Server Memory*

You can manage the application server processes in the `listener.dat` configuration file by defining the number of active processes and how they start up. Each active connection to the server requires one process. Starting more connections initially and allowing them to run continuously consumes more memory, but reduces the amount of CPU and other activities required to start a process. Each process for an active user requires approximately 150MB of memory. Adding memory to the server increases the ability to support additional concurrent application server processes.

For information about managing the server application pool and configuring `listener.dat` see page 48.

### *File Storage Server Memory*

The processes that perform tasks for the file storage layer are called library server processes. If your item libraries are remote to the Dimensions CM server, library server process are started the first time that file transfer, such as update and deliver, is initiated. These processes exist for the duration of the user's session and each one uses approximately 20MB of memory. Adding memory enables more processes to run and additional file transfer activities to take place.

### *Web Application Container Memory*

By default, CM installs a Java based web application container (Apache Tomcat). Adding more memory to the application server can improve your ability to scale vertically and support additional web clients. The amount of memory reserved for Java is configurable, for details see page 41. If you are using another server platform, consult the vendor for details about expanding memory. Tomcat typically requires 1.5GB to 2GB of memory.

### *Database Server Memory*

Check that your database has enough memory. Adding more memory may not necessarily allow the database to utilize it. Consult your database documentation for information on how to optimize memory configuration. Each concurrent process in a database server consumes approximately 20MB of memory. When calculating database server memory requirements, consider your users typical actions. For example, if they frequently run reports, make sure there is enough memory to support those processes and that the database is properly tuned.

## Scaling Single System Configurations

The disk subsystem in a single system configuration can also contribute to scalability. Consider the following:

- Faster disks enable operations to be performed quicker.
- If you use multiple disks or spindles, separate the database data and the database logs to improve efficiency.
- Store the item library on a separate disk to allow the drives to function better in parallel.
- Place the operating system on a separate drive. Reducing data access contention from a physical drive helps your configuration to scale.
- Techniques such as disk striping can help increase the throughput in certain configurations.
- Spread item library files across multiple folders. Too many files in a directory can impact the operating system's ability to serve up files.

## Network Hardware Performance

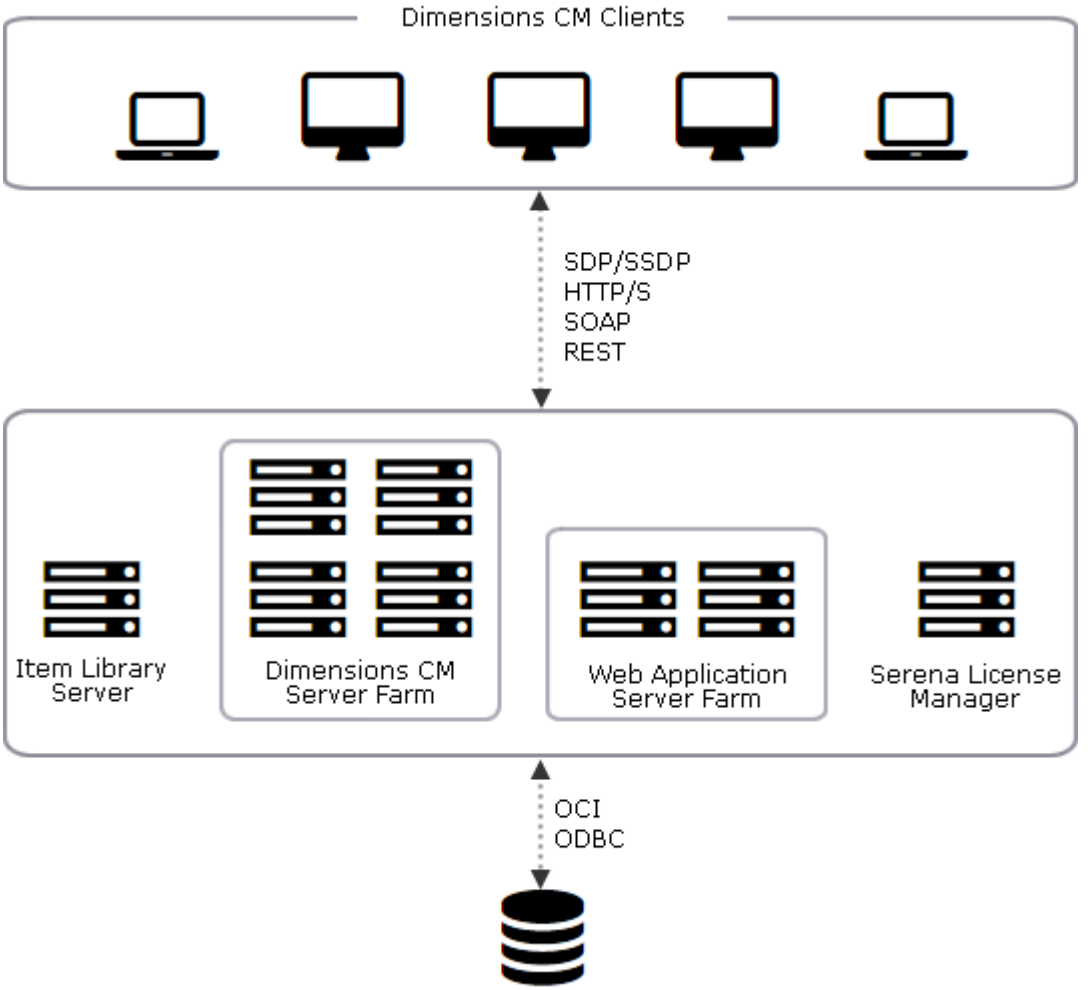
Network hardware is an important aspect of scaling. If a network card is saturated with network traffic, use multiple network cards to scale the application.

# Horizontal Scaling

You can expand the capacity of your server configuration by scaling horizontally and adding more physical machines, for example:

- Move the database and CM server to separate machines as these are the main components competing for CPU, RAM and disk I/O.
- Place the database and CM server on the same network subnet if possible. The closer in proximity they are, the less impact is caused by network traffic.

A horizontally scaled environment may look similar to this:



## Dynamic Load Balancing

Performance may benefit from the implementation of dynamic load balancing. Hardware load balancers require specialized equipment and skills and typically are more expensive than software load balancers. You can set up software load balancing with Microsoft Windows Server. With either type of load balancing, the end user continues to access the system via a single address and sees no difference in client behavior.

A Web Application can take advantage of a server farm that contains multiple containers, for example, Tomcat. Use the following methods to dynamically load-balance a Web Application:

- DNS based

  Directs clients to different servers as they request connections and balance the load across those servers. However, DNS load balancing has a known issue related to the caching of IP addresses. DNS connections are stored by clients for a pre-defined set of time, which may lead to failed connections if specific IP addresses have failed. The clients will not be automatically re-directed to a new IP address until the browser has been restarted or the allotted DNS time has passed.

- Reverse proxy based

  This technique caches content from web servers on proxy servers, accelerating the response time to client requests. You can use:

  - Software techniques as Squid Internet Object Cache, Microsoft Proxy Server, and Sun Netra Proxy Cache Server.
  - Hardware techniques as Cisco Systems LocalDirector and Coyote Point Systems Equalizer.

The Dimensions CM server can also be load balanced using a hardware load balancer that supports the use of "server affinity" algorithms. A CM client connects to the same physical hardware as the server after an "idle" time, meaning that the client communicates to the same IP address for the server for the lifetime of its session.

## Static Load Balancing

If you are unable to implement dynamic load balancing, you can configure multiple, static servers to distribute the load. In this type of configuration you define a set of fixed users that connect to each server. For example, you configure each department in your organization to access a specific system. Depending on your network bandwidth and project organization, you can set up local servers that connect to a central remote database that holds all the configuration management metadata for your organization.

Chapter 2

# Optimizing Dimensions CM


| | |
|---|---|
| Optimizing Your Configuration | 26 |
| Hardware Scaling Recommendations | 27 |
| Improving WAN Performance | 28 |
| Replicating Repositories | 34 |
| Versioned Repository Schema Caching | 35 |

# Optimizing Your Configuration

Dimensions CM performance is most affected by the following:

- Network configuration and topology

  CM is distributed across clients, servers, and a database server. Network hardware and software configuration have a significant impact on performance. Micro Focus recommends that:

  - The database server meets the system requirements for the RDBMS platform and has adequate RAM and system resources (see"Hardware Scaling Recommendations" on page 27).
  - The database server processes are executing on the fastest node in the network. If possible, that node should be dedicated to the RDBMS.
  - The operating system parameters are optimized with as much RAM as possible dedicated to each application server node in the network.
  - The server and database are located on the same segment of the local area network (LAN).

- Server hardware and CPU speed

  Server and database transactions are CPU intensive. Faster CPUs improve server performance. Invest in the highest performing server CPUs and ensure that memory requirements are met or surpassed.

- Wide area network configuration

  CM performs well across a wide area network (WAN), however, consider optimizing your environment as follows:

  - Sharing a central repository across a WAN.
  - Replicating repositories across sites if you have high latency and/or security considerations.

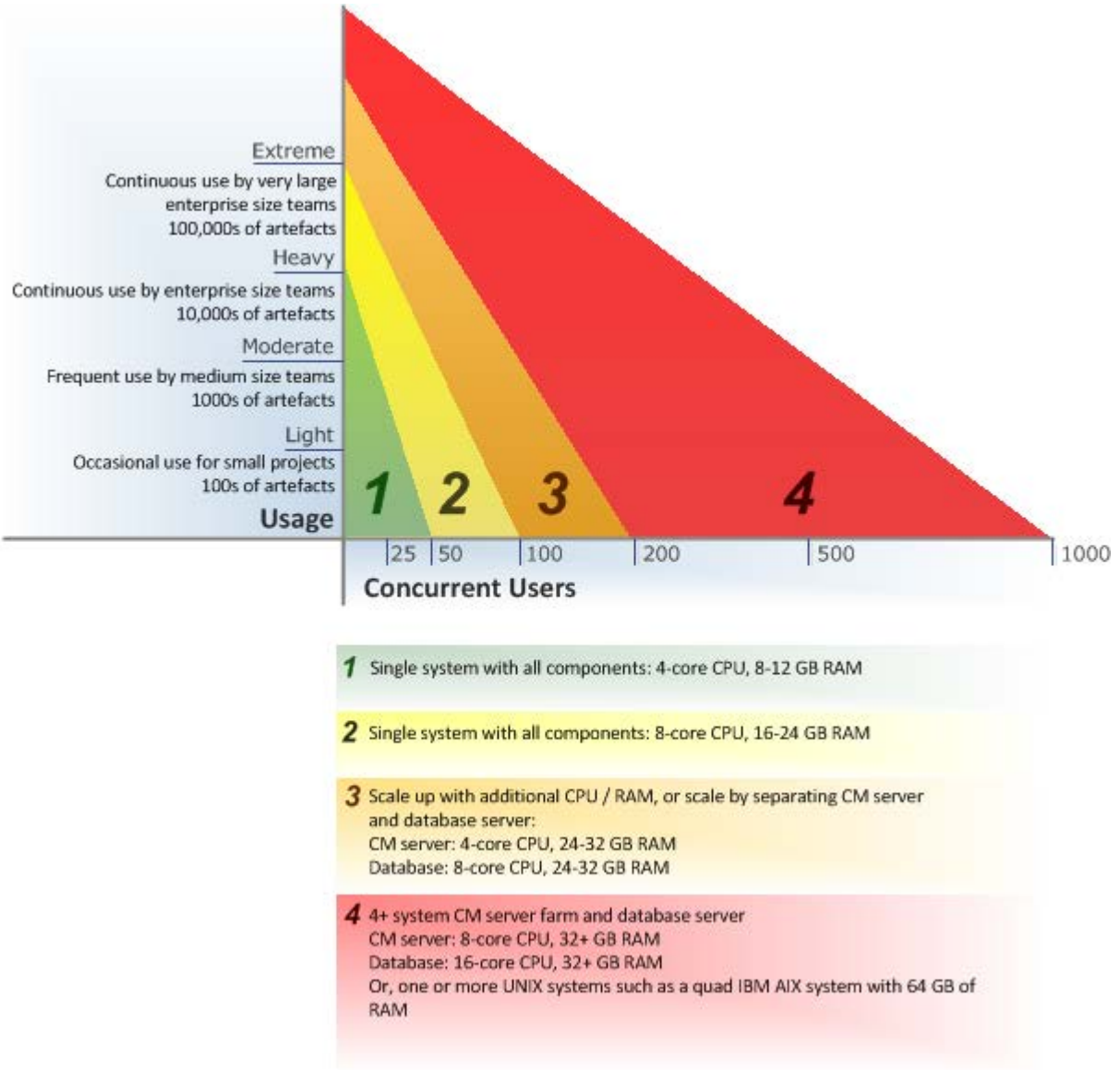
- Performance factors

  A system's network performance, and the optimizations that you can make, depend on a number of factors, including:

  - The size of the files
  - The data they contain
  - The connection speed
  - The hardware on the remote side

Understanding how your team uses Dimensions CM, for example distributed or local, will help you determine the best configuration. Experiment with the tuning optimizations described in this chapter to find the configuration that best serves your environment.

# Hardware Scaling Recommendations




Use the chart above to determine appropriate hardware needs, based on the expected number of active concurrent users. These recommendations are based on Micro Focus internal performance testing. A concurrent user is defined as a user who is actively performing work such as refreshing a local work area. Additional users may be logged in but they are not considered active if they are not currently performing a task against Dimensions CM. For example, if you expect an average of 50 active users making moderate use of the system at any given time, then follow the guidelines for zone 2.

# Improving WAN Performance

Development teams are often geographically dispersed across a WAN. You can use the following technologies to significantly performance and reduce network traffic:

- Library cache
- Personal library cache directory (PLCD)
- Intelligent file transfer

Using a library cache improves performance by caching file contents on a network node that is geographically close to the remote users. When users update files they are transferred from the library cache not the CM repository, which reduces transfer times by eliminating the geographical distance and resulting network latency. If a project or stream has many large files it is more likely to benefit from using library cache areas. Library cache areas are used on a per project/stream basis and each user can optionally configure their client to use them.

Using a personal library cache directory (PLCD) enables even faster access to repository files for distributed teams. PLCD removes network transfers when the same revision has been previously fetched to a work station or initially delivered from a work area on that work station. PLCD makes a local copy in a cache directory of items updated from, and delivered to, CM. This mechanism speeds up transfers when fetching the same revision more than once, for example, fetching a baseline or restoring item revisions.

Intelligent file transfer between all network nodes involved in an operation ensures that multiple files are streamed together as a large, compressed buffer and not individually. Where possible, only the delta between the currently available and the requested version of a file is transferred, and the requested version is automatically re-assembled on the destination network node.

## File Transfer with PLCD and Library Cache

When a command, such as UPDATE, transfers new files from an item library to a work area, it checks if PLCD is enabled on the work area network node and if a library cache is available. The command then checks if the required files have already been cached in the local PLCD.

- Files that are not cached are requested from the library cache. If it is not available, the files are transferred directly from the item library and then cached in PLCD.
- The library cache processes the list of files requested by the client. Any files that are not already cached in the library cache are transferred directly from the item library and then sent to the client.

The command then copies the files from PLCD into the local work area without any additional network traffic.

- If only PLCD is available, the process is the same as described above except that files missing in PLCD are requested directly from the item library.
- If only library cache is available, the process is the same as described above except that files are transferred from the library cache directly into the work area.
- If neither PLCD nor a library cache are available, the required files are transferred into the work area directly from the item library.

## Memory File Cache

CM transfers multiple files in bulk using the memory file cache. Files that need to be transferred are added to an in-memory cache buffer. When the cache buffer fills up, the files stored in the cache are sent to the receiving network node. To modify the size of the memory cache add the symbol `DM_FILECACHE_SIZE` to the server configuration file, for details see "Modifying Configuration Symbols" on page 51. The default size is 4MB. If your data set consists mainly of small files, you can improve performance by increasing the file size threshold of the memory cache. The results will depend on your network connection.

If PLCD is enabled on the client, CM uses delta compression during UPDATE and DELIVER commands to further reduce network traffic. For details "Using Delta Compression" on page 32.

## Automatic Library Cache Population

You can configure a CM server to automatically keep library cache areas in sync with the item libraries by constantly monitoring incoming changes to the item library. When a server detects a new changeset it launches the DLCA (Download to Library Cache Area) command, which updates the library cache area with the new item revisions in the changeset.

Using the current revision of an item in the library cache area and a new revision created in the item library, a delta is generated that only contains the differences between the two revisions. The deltas for changed items are sent to the library cache area and used to regenerate the item revision file. This ensures that the library cache is always automatically in sync with the item library without imposing a heavy load on the network.

**NOTE**

- To determine if PLCD is beneficial for your users on a LAN you can measure performance, for details see page 33.
- File transfers are typically much faster over a LAN than over a high-latency WAN. The savings in network traffic reduction resulting from PLCD usage on a LAN may not offset the overhead of extra local file IO operations required for PLCD operation. This depends on the LAN speed, if users store their work areas on fast SSDs or slower SATA drives, and on the typical CM operations performed. File transfer over a 1Gbps LAN should be faster than a local file copy on a slow SATA hard drive but slower than with a fast SSD.
- PLCD only works with Dimensions CM 14.x clients and servers and is not backwards compatible with versions earlier than 14.x. You should upgrade all library cache nodes and clients to Dimensions CM 14.x.
- The default size of the PLCD is 10GB but you can tune it to accommodate the size of your streams.

# Configuring PLCD on a Server

To specify the default PLCD settings for all clients that connect to a server, add the following variables to the server `dm.cfg` configuration file:

- `CLIENT_DM_PLCD_ENABLED`

  Enables or disables PLCD on all client machines that connect to this server. Accepts 1/0 or Y/N.

- `CLIENT_DM_PLCD_DIR`

  Specifies the default directory for all clients.

- `CLIENT_DM_PLCD_MAX_SIZE_MB`

  Specifies the maximum size for client local cache directories.

- `CLIENT_DM_PLCD_FILE_MAX_KB` and `CLIENT_DM_PLCD_FILE_MIN_KB`

  Specifies the maximum and minimum size of files that can be copied to the PLCD.

# Configuring PLCD on a Client

You can modify PLCD settings on client machines.

**NOTES**

- To reset the cache it is safe to delete the whole PLCD directory. Shut down the Dimensions CM clients before performing this action.
- Files downloaded into a work area using the fetch (FI), download (DOWNLOAD), and update (UPDATE) commands are copied into the cache.
- Files uploaded to Dimensions using the revise item (UI), checkin (RI), edit item (EI), deliver (DELIVER), and upload (UPLOAD) commands are copied to the cache.
- Changes that you make to the PLCD settings are saved in the file `plcd.properties` in: `<home>/.dmwebtools directory`

### *Enabling and Disabling PLCD*

PLCD is enabled by default on a Dimensions CM server and all clients that connect to it, including:

- Clients that run the dmcli command line, desktop client, and Eclipse and Visual Studio integrations.
- Agents including any used for library caches.

To disable PLCD do one of the following:

- Desktop client: select Tools | Preferences | Personal Library Cache tab and unselect Enable Personal Library Cache.
- Eclipse and Visual Studio: right-click the Dimensions CM server connection in Dimensions Explorer, select Properties | Personal Library Cache tab, and unselect Enable Personal Library Cache.

- In the Dimensions CM `dm.cfg` configuration file on a client set the variable `DM_PLCD_ENABLED` to 0.
- In a client such as `dmcli`, enter `SET PLCD OFF`. This disables PLCD for the duration of your session (or until you re-enable PLCD using `SET PLCD ON`).

### *Specifying a Cache Directory*

To specify where the cache is stored on a local machine do one of the following:

- Desktop client: select Tools | Preferences | Personal Library Cache tab. In the Cache directory path box enter the location of the cache directory on your local machine.
- Eclipse and Visual Studio: right-click the Dimensions CM server connection in Dimensions Explorer and select Properties | Personal Library Cache tab. In the Cache directory path field enter the location of the cache directory on your local machine.
- In the Dimensions CM `dm.cfg` configuration file add the variable:

  ```
  DM_PLCD_DIR C:\dimensions\plcd\
  ```

### *Specifying a Maximum Cache Size*

You can specify a maximum cache size (in megabytes) for the PLCD directory. When the directory reaches this size, earlier revisions of items are deleted to make more space.

- Desktop client: select Tools | Preferences | Personal Library Cache tab. In the Maximum cache size field enter a cache size.
- Eclipse and Visual Studio: right-click the Dimensions CM server connection in Dimensions Explorer and select Properties | Personal Library Cache tab. In the Maximum cache size field enter a cache size.
- In the Dimensions CM `dm.cfg` configuration file add the variable:

  ```
  DM_PLCD_MAX_SIZE_MB <cache size in MB>
  ```

### *Creating Additional Messages*

To create additional messages that confirm the use of the cache, add the following variable to the client configuration file:

```
DM_PLCD_BOAST 1
```

### *Specifying File Sizes*

To specify the maximum and minimum size of files that can be copied to PLCD, add these variables to the client configuration file:

- `DM_PLCD_FILE_MAX_KB <size in MB>`

  Default: 500 MB
- `DM_PLCD_FILE_MIN_KB <size in MB>`

# Using Delta Compression

Delta compression on file transfers reduces network traffic by only transferring the sections of files that have been modified between revisions. For this optimization to be applied the original revision of the file must have been fetched to, or saved from, the local work station. The greatest reduction in transfer time is for files that have modifications in only a few sections and large continuous sections of unchanged content. Files that have widely dispersed multiple changes between revisions have a smaller improvement in transfer times.

Delta transfer is automatically enabled on a Dimensions CM server and all clients that connect to it, including:

- Clients that run the dmcli command line, desktop client, and Eclipse and Visual Studio integrations.
- Item library host (if different to the server).
- Agents, including any used as library caches.

**IMPORTANT!** To use delta compression, personal library cache must be enabled.

**NOTE** Delta compression is only used only for files larger than 32 KB.

### *Disabling Delta Transfers*

To disable delta transfer, in the Dimensions CM `dm.cfg` configuration file set the variable `DM_DELTA_ENABLED` to 0.

### *Using Delta Transfers with Text Files*

If you are using delta transfers with text files, add the following variable to the Dimensions CM `dm.cfg` configuration file on the server.

```
DM_DISABLE_REVERSE_IHS 1
```

### *Enabling Delta Compression*

Delta compression is disabled automatically for data formats that cannot be compressed. You can modify this setting for each format:

1. In the administration console select Data Format & MIME types.
2. Select a data format and click Edit.
3. From the Compression level list select the appropriate compression for the data format.
4. Click OK.

# Measuring Performance Benefits

To determine performance benefits when using PLCD and/or library cache areas, measure the following scenarios:

- Perform a full update from a typical project or stream into an empty work area.
- Deliver some new or modified files into the project or stream from another work area.
- Perform an incremental update from the same project or stream into the first work area.
- Perform a full update from the same project or stream into a third empty work area.

You should perform these operations with:

- Both PLCD and library cache areas turned off.
- Empty library cache areas and an empty PLCD.
- Populated library cache areas and an empty PLCD.
- Populated library cache areas and a populated PLCD.

**TIP** Try and measure performance on at least two client systems.

### *Measuring Performance with the UPDATE and DELIVER Commands*

**1** Check that you do not have `DBIO` or `SDP` tracing symbols defined in `dm.cfg` on the client, server, and library cache node. If necessary, comment out the symbols and restart the Dimensions Listener service.

**2** Start a Dimensions CM command-line session (`dmcli`).

**3** From `dmcli`, execute the `SET PLCD OFF` or `SET PLCD ON` commands to manually disable or enable PLCD usage for the duration of the test session.

**4** From `dmcli`, execute a switch workset (`SCWS`) command to ensure that:

- Library caching is on or off (depending on which measurement you are taking).
- The working directory is set correctly.

**5** Enter a `SET AUTO_TIME ON` command

**6** Enter `UPDATE` and `DELIVER` commands as described above to execute the performance measurement steps.

**7** Note the time intervals required to complete various stages of the `UPDATE` and `DELIVER` operations.

**8** After you have made the change to be measured, such as enabling or disabling PLCD or library cache areas, perform the steps again to measure the change in performance.

### *Avoiding File System Performance Issues in Item Libraries*

When you deliver thousands of files to Dimensions CM for the first time, the item library typically puts all files with the same item type into the same library folder. To avoid file system performance issues, add the following flag to `dm.cfg` on the server:

```
DM_RANDOMIZE_LIBRARY_PATHS y
```

The server automatically spreads new files mapped to the same item type across X/Y/ sub folders of the corresponding item library folder, where X and Y are the first two hex digits from a cryptographically secure checksum hash of the corresponding item library path name.

# Replicating Repositories

For most customers, replicating repositories is not the most suitable option. However, if you have high latency and/or security considerations, you should consider using Dimensions CM Replicator. This option allows you to replicate data from one site to another to provide high speed local access to users at each site.

If you decide to implement replication, you have two options:

- Online Replication
- Offline (Air-Gap) Replication

It is possible to use either of these options in read-only mode. For detailed information about Replicator see the *System Administration Guide*. Dimensions CM Replicator requires a separate license.

# Versioned Repository Schema Caching

## Overview

A Dimensions CM server pre-caches Versioned Repository Schema (VRS) data into a directory on disk. Pre-caching is automatic when a project, stream, or baseline is accessed for the first time and is sufficiently large to benefit from pre-caching. The main benefits of data caching are improved performance when:

- Opening, or switching between, projects, streams, and baselines.
- Performing operations such as update, deliver, create baseline, create stream, and merge.

CM administrators can perform initial pre-caching of the name store, project, or baseline structure data for the current base database.

See also this Knowledgebase article.

## Configuring VRS

To configure VRS, modify the following variables in the server `dm.cfg` configuration file:

- `DISABLE_DM_DBCACHE`

  VRS data caching is enabled by default. To disable it, set this value to 'no':

  ```
  DISABLE_DM_DBCACHE n
  ```

- `DM_DBCACHE_DIR`

  Specifies the location of the VRS data cache on the server. Defaults:

  Windows: `DM_DBCACHE_DIR C:\ProgramData\Micro Focus\Server\db_cache_dir\`

  UNIX: `DM_DBCACHE_DIR %DM_ROOT%db_cache_dir/`

  Users that start application server processes (`DMAPPSRV`) must have read and write permission on the cache directory. User can be:

  - The pool owner if the listener pool is configured to execute in proxy mode.
  - The user group that is allowed to login when proxy mode is not in use (`DM_ROOT/dfs/listener.dat` has the parameter `-dont_use_proxy`).

The following variables are hard coded and cannot be changed:

- `DM_DBCACHE_UPDATE_THRESHOLD 4096`

  Specifies the number of new non-cached names/arcs loaded into a private VRS store section that trigger initial pre-caching/updating of an existing store.

- `DM_DBCACHE_CLEANUP_INTERVAL 24`

  Specifies the interval (in hours) the cache checks if the total file size for a base database exceeds the threshold set by the parameter `DM_DBCACHE_MAXSIZE`. If the threshold is exceeded, the files that have not been accessed for the longest period are deleted from the cache.

- DM_DBCACHE_MAXSIZE 4096

  Specifies the cache size threshold that triggers a cache cleanup.

- DM_DBCACHE_CLEANUP_THROTTLE 3600

  Idle timeout and client sessions can be very short. To avoid frequent, unnecessary cache cleanups, the default throttle interval controls how often a single application server process can schedule a cache cleanup (3600 seconds/1 an hour).

  When a product is deleted via the DWP command, the application server runs a forced cache cleanup irrespective of the last cache cleanup time or interval.

## Pre-Caching VRS Data

**NOTE** Shut down your Dimensions CM server processes before running this operation.

Run the following DMBDA command to perform initial pre-caching of VRS data:

```
PRECACHE
<mode>
[<project-spec> | <baseline-spec> | /BULK_FILE=<file> ]
```

where:

- <mode> specifies the type of data to be pre-cached. This parameter is mandatory and must have one of the following values:
  - NAMESTORE

    Pre-caches namestore data.

  - PROJECT

    Pre-caches project structure data.

  - BASELINE

    Pre-caches baseline structure data.

- <project-spec>

  Specifies a single project or stream.

- <baseline-spec>

  Specifies a single baseline.

- /BULK_FILE=<file>

  Specifies a file containing multiple project or baseline specifications to pre-cache.

**TIP** The project and/or baseline specification has the same syntax as a LIKE expression in SQL.

Examples:

```
DMDBA cm_typical/cm_typical@dim14

CM_TYPICAL> PRECACHE NAMESTORE

CM_TYPICAL> PRECACHE PROJECT %

CM_TYPICAL> PRECACHE BASELINE QLARIUS:%
```

```
CM_TYPICAL> PRECACHE PROJECT /BULKFILE="C:\Temp\A,txt"
```

where `A.txt` contains multiple project specifications, for example:

- `QLARIUS:BRANCHA\`
- `QLARIUS:BRANCHB\`

# Chapter 3

# Advanced Optimization Techniques


| | |
|---|---|
| Using Single Privilege Checks | 40 |
| Optimizing Performance Based on Bandwidth | 40 |
| Optimizing the Web Client | 41 |
| Disabling ACKs and The Nagle Algorithm | 43 |
| Optimizing the File Compression Levels | 45 |
| Non-Encryption of Item Contents | 47 |
| Optimizing AIX Performance | 47 |
| Managing the Server Application Pool | 48 |
| Modifying Server Configuration Symbols | 51 |

# Using Single Privilege Checks

Privileges are required to download and upload files. You can configure Dimensions CM so that these privileges are checked only once, and if the corresponding privilege is granted, then the underlying sub-commands will not check for privileges. This reduces CPU/IO load on the Dimensions CM server and improves LAN and WAN performance.

- To perform a single privilege check when downloading for a project, assign the *Download files from Project* privilege, PROJECT_DOWNLOAD, to the appropriate users.
- To perform a single privilege check when uploading for a project, assign the project *Upload Files into Project* privilege, PROJECT_UPLOAD, to the appropriate users.
- To perform a single privilege check when downloading for a baseline, assign the project *Download Files from Baseline* privilege, BASELINE_DOWNLOAD, to the appropriate users.

For details about assigning privileges see the *Process Configuration Guide*.

# Optimizing Performance Based on Bandwidth

You can optimize your system's network throughput by calculating the bandwidth-delay product (BDP) and modifying configuration symbols. The BDP is the total available bandwidth multiplied by the total round-trip time.

## Calculating BDP

**1** Measure one of the following:

- Total available bandwidth in bits/second and round trip time in seconds
- Total available bandwidth in KB/second and round trip time in milliseconds

You can calculate the total available bandwidth using an average of FTP transfer times.

You can calculate the round trip time using an average of the PING reported approximate round trip times.

**2** Calculate the BDP use one of the following formulas:

- Total available bandwidth in bits/second multiplied by the round trip time in seconds
- Total available bandwidth in KBytes/second multiplied by the round trip time in milliseconds

## Optimizing TCP/IP Buffer Values

- To optimize the TCP/IP send and receive buffer values, follow the procedure on page 51 and add the following values for the symbol DM_SOCKET_OPTIONS:
  - SO_RCVBUF(bdp value)
  - SO_SNDBUF(bdp value)

  where bdp is set to the bandwidth-delay product (BDP).
- To make room for the Dimensions CM message headers, follow the procedure on page 51 and add or edit the following symbols:
  - Modify the symbol DM_NETWORK_BLOCKSIZE to be 1000 bytes (or more) but not exceed the value of BDP.
  - Modify the symbol DM_FILE_BLOCKSIZE to be 4000 bytes (or more) but not exceed the value of DM_NETWORK_BLOCKSIZE.

# Optimizing the Web Client

The web client only connects to the Dimensions CM server and does not connect to a library cache area. To optimize performance for web client users:

- Locate the web server and the CM server on the same fast LAN.
- If a web client user experiences problems downloading a large project, they should download separate parts and sub folders of the project, rather than the whole project at one time.
- Optimize memory setup for Tomcat (see below).

## Optimizing Memory Setup for Tomcat

If multiple developers are using the web client you can improve performance by modifying the memory size setup for Tomcat. If only a few users are using the web client an improvement in performance is unlikely.

### *Java Command-Line Arguments*

You can optimize the memory setup for Tomcat by modifying the following Java command-line argument values for the initial heap size and maximum heap size.

| Command-Line Argument | Description | Default |
|---|---|---|
| -Xms | initial heap size | 640M |
| -Xmx | maximum heap size | 1280M |

To change the Tomcat maximum memory size on Windows (running as a service):

1. At a command prompt enter the following (or similar):

   ```
   "C:\Program Files\Micro Focus\Common
   Tools\tomcat\8.5\bin\tomcat8w.exe"
   //ES//MicroFocusTomcat
   ```

   Edit this example as required depending on the installation location.

   The Micro Focus Common Tomcat Properties dialog box appears.

2. Select the **Java** tab.

3. In the **Maximum memory pool** field enter a value.

4. Click **OK**.

5. Restart the Micro Focus Common Tomcat Service service.

To change the Tomcat maximum memory size on UNIX:

1. Open this file: `${INSTALL_LOCATION}/common/tomcat/8.5/bin/setenv.sh`

2. Find the line that starts with: `"CATALINA_OPTS=..."`.

3. Edit the line and enter an appropriate value, for example:

   - Increase `-Xmx1280m` to `-Xmx2048m` if your system can support that memory size, or
   - Reduce the value if you are not using the web client.

4. Save and close the file.

5. Restart Tomcat using the `shutdown.sh` and `startup.sh` scripts.

# Disabling ACKs and The Nagle Algorithm

You can gain significant performance improvements by disabling delayed acknowledgments (ACKs) and the Nagle algorithm.

## Disabling Delayed ACKs

The speed of the exchange of data between the Dimensions CM server and connected machines is affected by how that data is received and sent. When data is received there is a delayed acknowledgment (or *delayed ACK*). Instead of acknowledging every packet, a delay is introduced before an acknowledgment is sent, allowing more data to come in and be acknowledged in one ACK. When small packets are being sent this can have a negative impact on Dimensions CM performance. Turning off delayed ACKs should improve performance.To switch off delayed ACKs:

| Operating System | Command or Action |
|---|---|
| Solaris (9 and 10) | `/usr/sbin/ndd -set /dev/tcp tcp_deferred_ack_interval 5` |
| HP-UX (IA64 and RISC) | `ndd -set /dev/tcp tcp_deferred_ack_interval 1` |
| AIX | `/usr/sbin/no -o tcp_nodelayack=1`<br>`/usr/sbin/no -o delayack=0`<br>`/usr/sbin/no -o delayackports={}` |
| Windows | **1** See this web site for instructions on how to set the `TcpAckFrequency` value: http://support.microsoft.com/kb/328890/<br>**2** See this web site to check if the previous instructions apply to your environment: http://support.microsoft.com/kb/321098<br>**3** Using the instructions in step 1, set the registry key `TcpAckFrequency` to 1. |

See the example on page 44.

## Disabling the Nagle Algorithm

The Nagle algorithm is designed to avoid excessive sending of small segments by coalescing small segments into larger ones. The usefulness of the Nagle algorithm is determined by the application network protocol details, especially the granularity and frequency of messages sent by the application, and the expected purpose of the application.

Tests have shown that Dimensions CM performs better with the Nagle algorithm disabled. You should test the performance in your environment. By default the Nagle algorithm is switched off in Dimensions CM and the configuration file `dm.cfg` has the following entry:

```
DM_SOCKET_OPTIONS TCP_NODELAY(1)
```

If you accept the default you also need to disable the Nagle algorithm on the operating system:

**CAUTION!** If there is a hardware network problem you could loose TCP packets.

| Operating System | Command/Action |
|---|---|
| Solaris (9 and 10) | Run the following command:<br>`/usr/sbin/ndd -set /dev/tcp tcp_naglim_def 1` |
| HP-UX (IA64 and RISC) | Run the following command:<br>`ndd -set /dev/tcp tcp_naglim_def 1` |
| AIX | Run the following command:<br>`/usr/sbin/no –o tcp_nagle_limit=1` |
| Windows | Check that the `DM_SOCKET_OPTION` symbol in the dm.cfg file has the `TCP_NODELAY` option value set to:<br>`TCP_NODELAY(1)` |

If you do not add these settings to the system files they may be lost when the machine is rebooted.

### *Example*

The environment in the example below has an AIX Dimensions CM server and a Solaris database server running an Oracle.

On a Solaris database server:

**1** Turn off the delayed ACKs and disable the Nagle algorithm by running this command:

```
/usr/sbin/ndd -set /dev/tcp tcp_deferred_ack_interval 5
/usr/sbin/ndd -set /dev/tcp tcp_naglim_def 1
```

**2** Shut down Oracle.

**3** Restart Oracle. The commands only affect the running system, these settings will be lost when you reboot.

**4** To make the changes permanent add the following lines to the `/etc/system` file:

```
set tcp:tcp_deferred_ack_interval=5
set tcp:tcp_naglim_def=1
```

On an AIX Dimensions CM server:

**1** Turn off the delayed ACKs by running these commands:

```
/usr/sbin/no -o delayack=0
/usr/sbin/no -o delayackports={}
/usr/sbin/no -o tcp_nagle_limit=1
/usr/sbin/no -o rfc1323=1
```

**2** Restart the Dimensions CM listener so that the changes will take effect. The commands only affect the running system, these settings will be lost when you reboot.

**3** To make the changes permanent add the following lines to the /etc/rc.net file:

```
/usr/sbin/no -o delayack=0
/usr/sbin/no -o delayackports={}
/usr/sbin/no -o tcp_nagle_limit=1
/usr/sbin/no -o rfc1323=1
```

# Optimizing the File Compression Levels

Dimensions CM supports compression on file transfers using ZLIB. You can improve performance by modifying the compression levels.

## File Compression Levels

The table below details the file compression levels.

| Value | Description |
|---|---|
| 0 | No compression. Default for binary file formats. |
| 1 | Fastest compression speed but lowest compression obtained. Default for text file formats. |
| 2 to 8 | For each increase in the value, compression is increased but there is a decrease in compression speed. |
| 9 | Highest compression with slowest compression speed. Gives the best network performance but will slow down the performance of the machine. |

Guidelines for modifying default values.

| Resource or file | Compression level |
|---|---|
| ■ Plenty of CPU and server memory<br>■ Binary files<br>■ Multiple smaller packets | Higher level |
| ■ Text files | Lower level |
| ■ Files that are already compressed, for example .jpg, .mpeg, and .zip files. | No compression |

# Setting Compression Levels

Set the file compression values using one of the following:

- Administration console: choose Fast, Normal, or Best
- Dimensions CM command-line interface (`dmcli`): specify a numeric value

The table below details the administrator console and DMCLI equivalent values.

| Administration Console Value | dmcli Equivalent Value |
|---|---|
| Fast | 1 |
| Normal | 6 |
| Best | 9 |

### *Using the Administration Console*

**PRIVILEGES** To set the compression level you need the following privilege: Administration Privileges | Process Management Manage File Format Definitions

**1** From the Administration Console select Configuration Object Management | Data format & MIME types.

**2** Select the **Formats** tab.

**3** Select the appropriate data format.

**4** Click **Edit**. The Edit Data Formats dialog box appears.

**5** From the **Compression level** list select the appropriate compression level. *Best* is not recommended as it may overload system performance until data transfer is complete.

**6** Click **OK**.

### *Using dmcli*

Using the command-line client you can specify if files of the associated format should be considered for compression during transfer (reducing bandwidth) and what level of compression to apply. Use the optional `/COMPRESSION_LEVEL=<`*`level`*`>` qualifier with the following `DMCLI` commands:

- `DDF` (Define Data Formats)
- `SDF` (Set Data Format Flags)

**1** As a user with administrator privileges run `dmcli`.

**2** Enter commands and level values as appropriate, for example:

```
SDF TXT /COMPRESSION_LEVEL=<level>
```

### *Enabling and Disabling File Compression*

Follow the procedure on page 51 and set a value for the symbol `DM_COMPRESS_FILES_ON_TRANSFER`

- To enable compression: `yes`
- To disable file compression: `no`

### *Modifying the Compression Threshold*

Follow the procedure on page 51 and add the symbol `DM_COMPRESSION_MIN_FILELENGTH` *n*

where *n* is the compression threshold in bytes.

For example:

```
DM_COMPRESSION_MIN_FILELENGTH 63990
```

Files smaller than this value will not be compressed.

# Non-Encryption of Item Contents

All Dimensions CM commands passed between the clients and the database server are encrypted. This enhances Dimensions CM security for all commands and particularly any command that contains passwords such as AUTH or CA. You can also encrypt item contents as they are moved around the network using the Secure Sockets Layer (SSL) protocol. This is particularly important in high security environments. For information about using SSL see the *System Administration Guide*.

**NOTE** If the Dimensions CM listener is started with Secure Sockets Layer (`-ssl`) defined in `listener.dat`, all Dimensions CM clients connecting to this listener will use SLL mode. All `dmlibsrv` processes spawned by the listener will also use SLL mode. However, any connections from `dmappsrv` or `dmlibsrv` processes on this node to `dmappsrv` or `dmlibsrv` processes on other nodes will be unencrypted unless the other node's listener is also in SSL mode.

# Optimizing AIX Performance

By default, network performance with Dimensions CM servers on AIX may be significantly slower than performance on other platforms when the item libraries are located on the server. This is because network performance is inefficient when getting files out of the repository. You can significantly improve performance with AIX servers with local item libraries by adding the following configuration symbol to the `dm.cfg` file on the server:

```
DM_DONT_PREFETCH_FROM_LIBRARY y
```

# Managing the Server Application Pool

The Dimensions CM server offers an advanced pooling feature for Dimensions CM application servers. Server pooling manages connection requests from Dimensions CM clients. It is automatically invoked when Dimensions CM starts up and stopped on shutdown. Server pooling is invoked with various default parameters, which for many users will be adequate. For information on how the server pooling configuration may benefit performance see page 18.

You can configure the pool manager by specifying parameters such as:

- The service name (or the TCP/IP port number) to be used for listening to client requests.
- The number of application servers to be created when the pool manager is first started up.
- The minimum and maximum number of application servers allowed in the pool.
- The timeout period for an idle application server to be returned back to the pool.

The server utility `refreshpoolconfig` is also available to re-read any new changes to the configuration parameters. The utility `getpoolstats` is available to obtain pool status information.

**UNIX server:**

1. Log into the Dimensions CM server as user `root`.
2. Set up your access to Dimensions CM.
3. Type the following command at the operating-system prompt:

   ```
   dmlsnr -param $DM_ROOT/dfs/listener.dat
   ```

4. To update the pool configuration while Dimensions CM is running, edit `listener.dat` and enter one of the following commands (as user `root` or `dmsys`):

   - `refreshpoolconfig -host <machine>:<port>`

     Use this command if you are not using the default TCP port of 671 for the Dimensions Listener Service, in which case you must specify the appropriate port.

   - `refreshpoolconfig -host <machine>`

     Use this command if you are using the default TCP port of 671.

5. To display server pooling statistics, enter the following command (as user `root` or `dmsys`):

   ```
   getpoolstats
   ```

**Windows server:**

1. Log into the Dimensions CM server as a Windows administrator.
2. Set up your access to Dimensions CM.
3. Start the Dimensions Listener Service Windows service.

4 To update the pool configuration while Dimensions CM is running, edit `listener.dat` and enter one of the following commands (as user `root` or `dmsys`):

   - `refreshpoolconfig -host <machine>:<port>`

     Use this command if you are not using the default TCP port of 671 for the Dimensions Listener Service, in which case you must specify the appropriate port.

   - `refreshpoolconfig -host <machine>`

     Use this command if you are using the default TCP port of 671.

5 To display server pooling statistics, enter the following command (as user `root` or `dmsys`):

   `getpoolstats`

# Listener Options

The `listener.dat` file is a text file containing a series of lines in the following format:

`-<param><white-space><param-value>`

### *Parameters*

`-service` Default: `pcms_sdp`

Specifies the port that listens for client connections. You may run more than one pool using different ports.

`-user` Specifies the operating system account name that will own every application process in the pool. For Microsoft SQL Server Enterprise this must be the database owner. For all other databases this can be any valid operating system account.

`-dsn` Specifies the Dimensions CM base database name and the ODBC data source name (DSN) to use when starting up the initial applications servers.

`-min` Specifies the minimum number of application servers that the pool will shrink to.

Default: `5`

`-max` Specifies the maximum number of application servers that the pool will expand to.

Default: `120`

`-initial` Specifies the initial number of application servers that will be created in the pool at startup time. These will use the database connection parameter `-dsn` for identifying the base database to connect to.

Default: `5`

`-free` Specifies the maximum number of application servers that may be left idle. Once this limit is reached, the pool will start shrinking as necessary by destroying idle application servers.

Default: `20`

`-idle_timeout` Specifies the amount of time, in seconds, that a client can be idle before it is disconnected.

Default: `300` at installation

Minimum: none (`60` is recommended)

`-session_timeout` Specifies the amount of time, in seconds, that a client's session state is maintained by the pool manager before being discarded. The application server and the Dimensions CM license token are then returned to the pool.

Default: `86400` (`24` hours) at installation

Minimum: `3600` (`1` hour)

`-cert_timeout` In Dimensions CM for z/OS, a certificate mechanism is used to connect back to Dimensions CM from the mainframe. This parameter specifies the internal that must pass before the certificate expires.

Default: `3600` seconds at installation

Maximum: none

`cert_purge_timeout` In Dimensions CM for z/OS, a certificate mechanism is used to connect back to Dimensions CM from the mainframe. This parameter specifies how often the pool removes expired certificates.

Default: `3600` seconds at installation

Maximum: `3600` (1 hour)

`-max_auth_attempts` Specifies the maximum number of attempts that can be made to pass an authentication point. Once the limit is reached, the user's session is terminated and they have to log in again.

Specifying zero disables the functioning of this parameter allowing an unlimited number of attempts to be made to pass an authentication point.

To assist customers working in an environment where verification of a user's identity is required as part of the process, Dimensions CM offers authentication facilities for "sensitive" changes to an object's lifecycle state and attributes, improved audit trail generation, and new reporting facilities.

Default: `3`

Minimum: `0`

`-restricted_mode` Use this parameter to start a Dimensions CM server or agent in restricted mode, not as the default UNIX user root or a Windows local administrator, for example, `dmsys`. Also set the `-user` parameter to run as the appropriate user.

`-ssl` Used for Secure Sockets Layer (SSL) connections.

`-ssl_password` Used for Secure Sockets Layer (SSL) connections.

# Modifying Server Configuration Symbols

You can optimize performance by editing configuration options in the `dm.cfg` file on the Dimensions CM server. When making network changes it is important to consider what impact the change may have on other applications using the network and the server.

- Each Dimensions CM server or client installation contains a `dm.cfg` file, located in the `%DM_ROOT%` (Windows) or `$DM_ROOT` (UNIX) directory. If the installation is part of a larger network of Dimensions CM installations, check that configuration settings that affect communication with other installations are the same for all installations.
- Settings in the `dm.cfg` file are installation-wide so parameters that are specific to you must be set in your environment. If settings are made in the `dm.cfg` configuration file and in your environment, your environment takes precedence.
- The configuration parameters and environment variables are generically referred to as *Dimensions CM Symbols*. Most Dimensions CM symbols are supported on all the operating systems.
- Some symbols require the Dimensions CM listener to be stopped and started before the change takes effect.

## Modifying Configuration Symbols

1. Navigate to the `%DM_ROOT%` (Windows) or `$DM_ROOT` (UNIX) directory.
2. Make a copy of `dm.cfg`.
3. Open `dm.cfg`.
4. Modify, add, or remove symbols.
5. Save and close the file.
6. If you have modified `dm.cfg` restart the CM listener.

# Configuration Symbols Reference

The following table describes the configuration symbols in `dm.cfg` that you can modify to improve performance:

| Configuration Symbol | Description |
|---|---|
| `DM_SOCKET_OPTIONS` | The values of the options of this symbol can be customized to optimize network throughput.<br>The values for the for the options below are used to configure the TCP/IP send and receive buffers:<br>■ `SO_RCVBUF(bdp value)`<br>■ `SO_SNDBUF(bdp value)`<br>where *bdp* (bytes) is set to the bandwidth-delay product (BDP). See also page 41.<br>The option `TCP_NODELAY` is used to control the turning off or on the delayed acknowledgments (ACKs). See also page 43.<br>Example:<br>`DM_SOCKET_OPTIONS TCP_NODELAY(1),SO_LINGER(1,5),SO_REUSEADDR(1),SO_KEEPALIVE(1),SO_RCVBUF(125000),SO_SNDBUF(125000)` |
| `DM_NETWORK_BLOCKSIZE` | Should be 1000 bytes (or more) but not exceed the value of BDP. Makes room for the Dimensions CM message headers. File blocksize must not exceed the network blocksize. |
| `DM_FILE_BLOCKSIZE` | Should be 4000 bytes (or more) but not exceed the value of `DM_NETWORK_BLOCKSIZE`. Makes room for the Dimensions CM message headers. |
| `DM_FILECACHE_SIZE` *n* | All files that are not library cached go through the memory file cache. When the cache is full, the files stored in the cache are sent to the client. The memory file cache buffers data up to a maximum size.<br>*n* is equal to one of these values:<br>■ Default value: 4194304B (4MB)<br>■ The following value may improve performance: 8388608B (8MB)<br>See also page 29. |
| `DM_COMPRESSION_MIN_FILELENGTH` | Dimensions CM supports stream compression on file transfer using ZLIB. Specify a compression threshold. Files smaller than this value will not be considered for compression. The default minimum file size for compression is 2K.<br>See also page 45. |
| `DM_COMPRESS_FILES_ON_TRANSFER` | Enables file compression to speedup file transfers. |
| `DM_DONT_PREFETCH_FROM_LIBRARY` | Improves performance when getting files from a Dimensions CM server on AIX. Add this symbol and set it as follows:<br>`DM_DONT_PREFETCH_FROM_LIBRARY y` |
| `DM_RANDOMIZE_LIBRARY_PATHS y` | The server automatically spreads new files mapped to the same item type across X/Y/ sub folders of the corresponding item library folder, where X and Y are the first two hex digits from a cryptographically secure checksum hash of the corresponding item library path name. |