propelsystems

CIMERA™

CIMERA ARCHITECTURE

Release 4.2.x

Version 1.0, 13-May 2015

Gwyn Carwardine, Jon Bentley

gwyn.carwardine@propelsystems.com
jon.bentley@propelsystems.com

Table of Content

CIMERA ARCHITECTURE 1
Release 4.2.x 1
Table of Content 2
1 APPLICATION ARCHITECTURE 3
1.1 Overview 3
1.2 Components 3
1.2.1 Cimera Database 3
1.2.2 Metadata 3
1.2.3 Item Library 3
1.2.4 Client 3
1.2.5 Plug-Ins 4
1.2.6 Update Server 4
1.2.7 Query Server 4
1.2.8 Indexing Server 4
1.2.9 Reporting Server 5
2 INTEGRATION ARCHITECTURE 6
2.1 Overview 6
2.2 Client Application Programming Interface (API) 6
2.3 Plug-Ins 6
2.4 Web Services Integration 6
2.5 Windows Powershell 7
2.6 Exits 7
2.7 Reconciliation against an External Data Source (EDS) 7
2.8 Building Custom Integrations for ETL 7
3 DATA ARCHITECTURE 7
4 INFRASTRUCTURE ARCHITECTURE 7
4.1 Deployment Options 7
4.2 Network Architecture 9
4.2.1 Overview 9
4.2.2 Network Traffic Throughput 9
4.3 Backup & Recovery 9
4.4 System Requirements 10
4.4.1 Cimera Client 10
4.4.2 Cimera Server 10
4.4.3 DB Server 11
4.5 Data Storage Requirements 11
5 SECURITY ARCHITECTURE 11
5.1 Authentication and Access Control 11
5.2 Communication Security 11
5.3 Application Security 11
5.4 Application Audit 12
5.5 Alerting and Logging 12

1 APPLICATION ARCHITECTURE

1.1 Overview

Cimera is a .NET based client-server application utilising an industry standard relational database. Cimera has multiple server components hosted as Windows services.

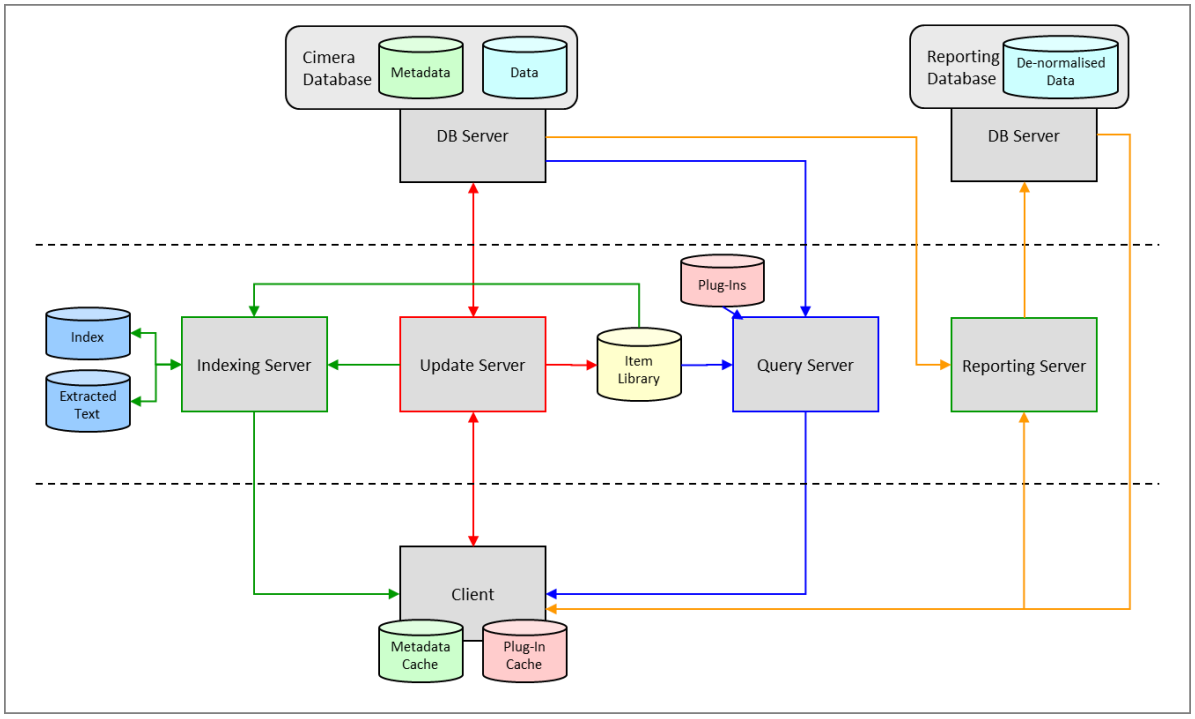

Figure 1 - Application Architecture

1.2 Components

1.2.1 Cimera Database

This holds both the Metadata and Cimera data.

1.2.2 Metadata

This describes the customer specific Cimera configuration, i.e. the Item Types, attributes, Lifecycles and Relationships etc. The Metadata lives in the Cimera Database, is cached in memory in the Update Server and is persistently cached by the Client (in a local JET database). Whenever the Client receives data from a Cimera Server it also receives the Metadata version number. If its own Metadata Cache is behind then it requests the delta from the Update Server.

1.2.3 Item Library

This is where all files that have been attached to Cimera Items are stored. It is implemented as a folder within the Windows file structure. Attached files are stored here rather than in the Cimera Database.

1.2.4 Client

Cimera has a “thick client” that is installed on each user’s computer. The Client includes a standard GUI, a command line interface (CLI) and an application programming interface (API). The standard GUI may be augmented by custom Plug-Ins or a custom GUI may be created.

1.2.5 Plug-Ins

These are customer specific client-side functional modules that are used to extend Cimera's behaviour on the client. Typically these are used to perform custom reporting or compound actions (i.e. a single user activity manifesting in multiple Cimera actions). A master Plug-In folder on the server holds the modules – this is manually updated with any new or changed modules and when the User next logs in to Cimera the Client will then download the modules to a local folder on the client machine.

1.2.6 Update Server

The Update Server is responsible for all updates to the Database and the Item Library. It also manages the Metadata which it additionally caches in memory for performance reasons.

One Update Server can host multiple Cimera instances. Whilst this would seem useful for hosting both Production and Test instances it is actually better to run these on separate infrastructure and is not recommended for this purpose.

The Update Server processes multiple client requests concurrently; the Database is responsible for maintaining transactional integrity and managing any necessary serialisation of updates. However, in the event of a Metadata update all subsequent client requests will be queued (to ensure the requests do not receive an inconsistent view of the Metadata cache). These Metadata updates are typically extremely quick – any longer running updates (there are only a few circumstances in which a Metadata update may be long running) are best performed at quiet times or out of normal business hours.

The Update Server manages the Item Library.

The Update Server contains a reconciliation engine which can extract data from external OLEDB/ODBC data sources and reconcile it with a copy of the data held in Cimera. A simple scheduler built into Cimera manages the execution of the reconciliation jobs.

The Update Server includes a messaging manager. This manages outbound email (SMTP) communication related to alerting and workflow.

The Update Server has two Exit points where custom modules can be called, one is called after a request has been received but before it has been actioned (allowing additional validation or modification of the user request) and the other is called after the request has been actioned (allowing additional logging or the triggering of activities external to Cimera)

There can be one and only one Update Server hosting a Cimera Instance. It is vertically scalable (by addition of RAM and CPU) but not horizontally scalable.

1.2.7 Query Server

The Query Server is used for all read-only activities from the Cimera Database, Item Library and master Plug-Ins folder.

There are two Exit points for the query server – one pre and one post processing.

Query Servers are both vertically and horizontally scalable. Multiple Query Servers may either be manually load balanced (by directing different sections of the user community to different servers) or sit behind an automatic load balancer.

1.2.8 Indexing Server

The Indexing Server is responsible for indexing the Cimera data and contents of the Item Library. It is also responsible for processing full-text search requests from the clients. The Indexing Server is an optional feature of Cimera.

The Indexing Server performs lazy indexing – periodically waking up and seeing what Cimera Items have been updated since it last ran and then retrieving the Items and any attachments for indexing. It will always be slightly behind the actual data, usually only a few minutes. In the rare event of a total index rebuild this may take minutes, hours or days, depending on the number of Items / Item Library members. A total index rebuild would only be required where a serious failure has occurred or where Cimera upgrade requires it due to incompatibility with a previous version. A Cimera upgrade would be a planned event and therefore any disruption could be reduced.

The Indexing Server accesses the Cimera data via a connection to the Update Server.

The Indexing Server uses installed IFilters[1] to extract the text from files stored in the Item Library. This text is then both indexed and stored. Storing the text enables the Indexing Server to return matching text snippets and also to return preview data, allowing the user to quickly review the textual content of a document without having to open the full document.

If the Index is ever deleted then the Index will be rebuilt in the background when the Indexing Server is started. This can be useful after adding additional IFilters which will enable new file formats to be indexed or should the Index ever become corrupted.

The Indexing Server has two data stores, one for the Index itself and one for the extracted text. These are both implemented as a number of flat files and should be local to the Indexing Server.

There can be one and only one Indexing Server hosting a Cimera Instance. It is vertically scalable (by addition of RAM and CPU) but not horizontally scalable.

1.2.9 Reporting Server

The Reporting Server is responsible for copying and transforming the data held by Cimera in its proprietary schema to a second, de-normalised SQL Server database that the user can run standard SQL queries and reports against. The Reporting Server is an optional feature of Cimera.

All updates made to Cimera are then placed in a separate queue within the main Cimera database. Periodically, e.g. every 20 seconds, the Reporting Server looks for entries in the queue and processes them, transforming the data into a de-normalised user-friendly schema. The data in the reporting database should be considered to be *near-realtime*. Complex user-defined queries and reports may now be run against the reporting database without impacting other Cimera users or processes.

Note, the user does not go through the Reporting Server to access the reporting database, the user makes a direct connection with the SQL Server hosting the reporting database. Advanced SQL Server services such as SSRS may of course be used to publish reports and dashboards on the intranet based on Cimera data.

The reporting database may be held on the same server/instance as the main Cimera database or not, as required.

There can be one and only one Reporting Server hosting a Cimera Instance. It is vertically scalable (by addition of RAM and CPU) but not horizontally scalable.

[1] An IFilter is a plugin that allows Microsoft's search engines to index various file formats. IFilters are described here http://en.wikipedia.org/wiki/IFilter

2 INTEGRATION ARCHITECTURE

2.1 Overview

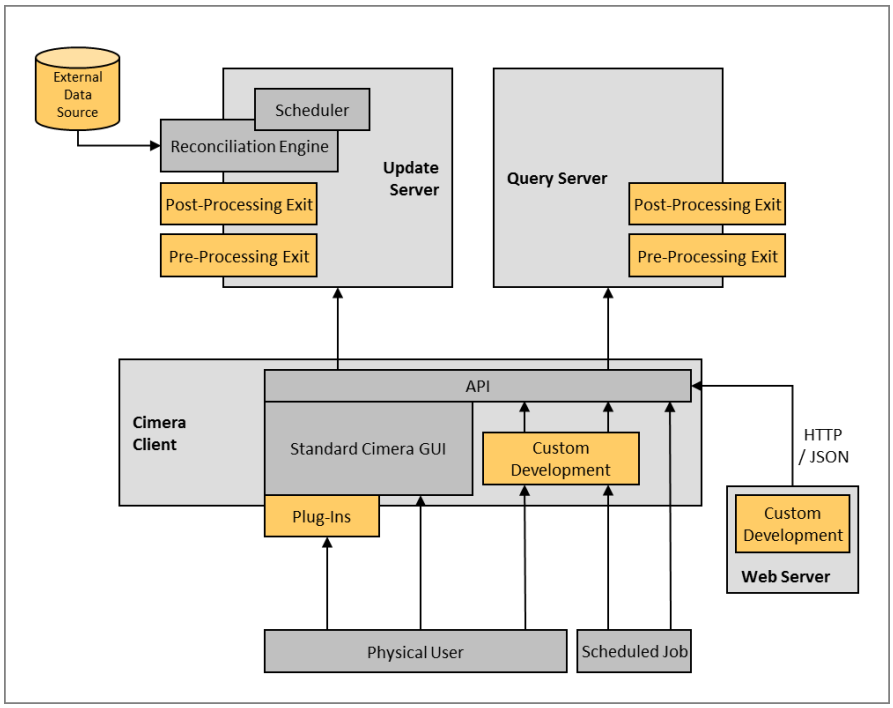

Figure 2 - Integration Architecture

Integration is enabled in various ways:

2.2 Client Application Programming Interface (API)

There is an application programming interface which can be developed against in .NET which allows programmatic access to the full functionality of Cimera. This API is supplied as part of the client installation. Establishing a session with the Update Server through the API requires standard Cimera credentials (userid and password) and any actions will operate within the security privileges of the supplied credentials.

2.3 Plug-Ins

Plug-Ins allow integration with the Cimera Client – they become available through the Plug-Ins menu or from the Shortcut Bar (left pane in the main Cimera GUI). Plug-Ins can be context-sensitive in which case they are only available when right-clicking certain Items in the GUI. They may also be secured to certain users, groups or roles. In essence whenever an Item is right-clicked all loaded Plug-Ins are dynamically offered the opportunity to extend the menu. A Plug-In has full access to the capabilities of the Cimera API (the security privileges of the currently authenticated user still apply).

Plug-Ins must be developed in .NET and of a version less than or equal to that of the Cimera Client. The appropriate version of the .NET framework must already be installed on the client computer.

Plug-Ins are stored on the server and the client will download all available Plug-Ins when the user successfully authenticates against (logs in to) a Cimera Database.

2.4 Web Services Integration

The Cimera server services may be called using JSON formatted requests over HTTP. This enables integration with web services and allows consumer services to interact with Cimera, subject to standard Cimera authentication.

2.5 Windows Powershell

The full Cimera API is exposed to Powershell enabling complex automated processing to be developed or scripted.

2.6 Exits

Exits enable requests to the Cimera Server to be inspected, and potentially altered or rejected, prior to processing and also for the results to be inspected post-processing and actions taken that may be internal or external to Cimera (such as updating an external data set or starting an external process).

Exits must be developed in .NET and of a version less than or equal to that of the Cimera Server. The appropriate version of the .NET framework must already be installed on the server computer.

2.7 Reconciliation against an External Data Source (EDS)

The Cimera Update Server contains a reconciliation engine and a basic scheduler. This allows Cimera to pull data from *any OLEDB / ODBC compliant data source* (this includes Excel and CSV files) and to reconcile it with data already in Cimera, adding and updating Items where necessary to bring them into alignment.

This can be used for simple reconciliation where one row returned from a data source SQL Select statement maps to one Cimera Item. There is no translation possible, simply column mapping.

Reconciliation can also delete items that are not in the External Data Source but only if it extracts the full list instead of just those that have been changed (as a deleted item will not exist in the delta). This means that reconciling against a large data source (e.g. thousands of data points) with the option to detect deletes could be an expensive operation. However many of the large data sources that would typically be used would *soft delete* or *close* items rather than actually delete them (because when you delete data in a relational database you tend to lose its history). Defect tracking and change and configuration management systems rarely delete items.

2.8 Building Custom Integrations for ETL

Where complexity or a need for translation prevents using Cimera's reconciliation engine then a custom integration can simply be built in .NET (or scripted using Powershell) utilising the Client API to perform full extract, transformation and load (ETL) as required.

3 DATA ARCHITECTURE

The Cimera Database structure is heavily optimised for performance and the data has to be combined with the Metadata to make sense of it. It is therefore essentially proprietary and not to be considered user meaningful. All data access should therefore be made through Cimera using the Cimera Query Language (CQL) or via the use of SQL queries run against the separate SQL Reporting Database.

Cimera has no built-in data archiving. It would be possible to develop a custom archiving solution but there has been no need to date. Additional data does not materially affect performance. The database is heavily indexed which does mean that inserts and updates are relatively expensive however with a Cimera-type system the majority of the accesses are reads with relatively few inserts and updates.

4 INFRASTRUCTURE ARCHITECTURE

4.1 Deployment Options

Each Cimera Server can be hosted on its own server or they can be hosted on one server. Additional horizontal scalability may be achieved by adding additional Query Servers. The majority of Cimera access will be read-only and via a Query Server.

The simplest and highest performing configurations are shown in the following illustrations:

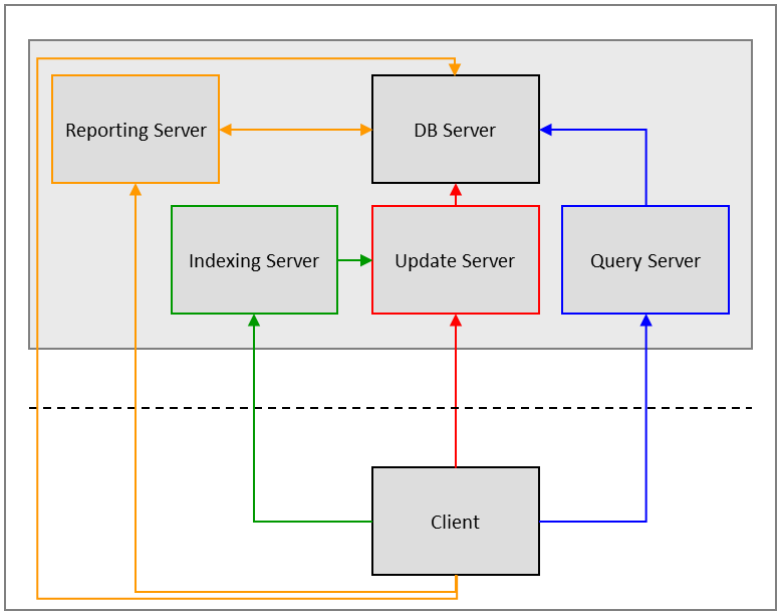

Figure 3 - One server hosting all

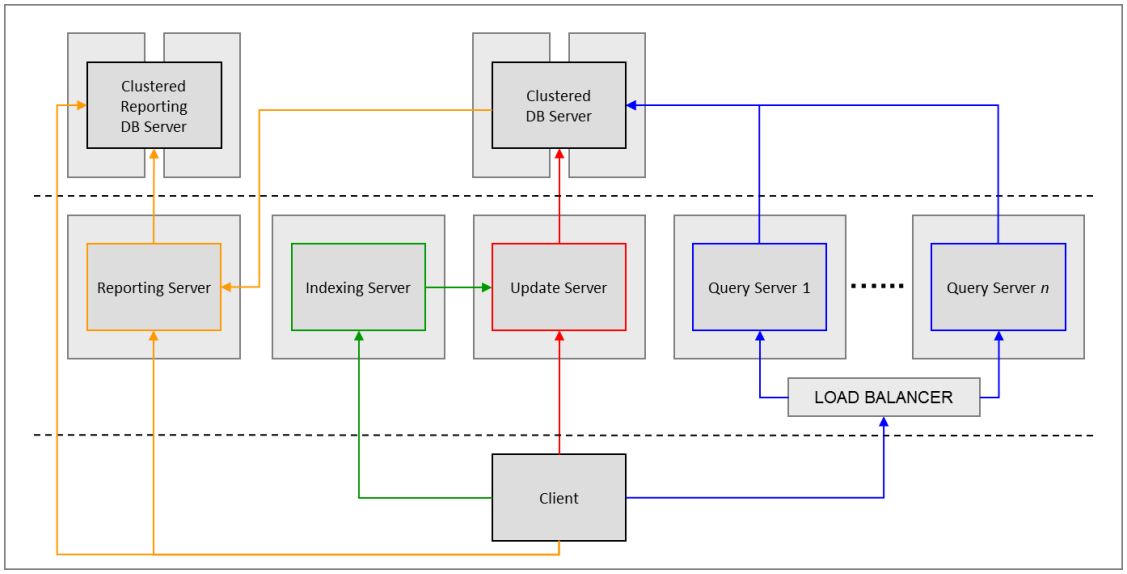

Figure 4 - Maximum performance

The recommended configuration is to run all Cimera Servers on a single server. If this does not perform sufficiently then analysis dependent on the actual usage profile would determine whether it would be better to scale vertically or to horizontally.

In the event that Cimera Servers are run on separate physical / virtual servers then the Item Library and Plug-Ins folder must be placed somewhere mutually accessible (perhaps a on a separate NAS server)

4.2 Network Architecture

4.2.1 Overview

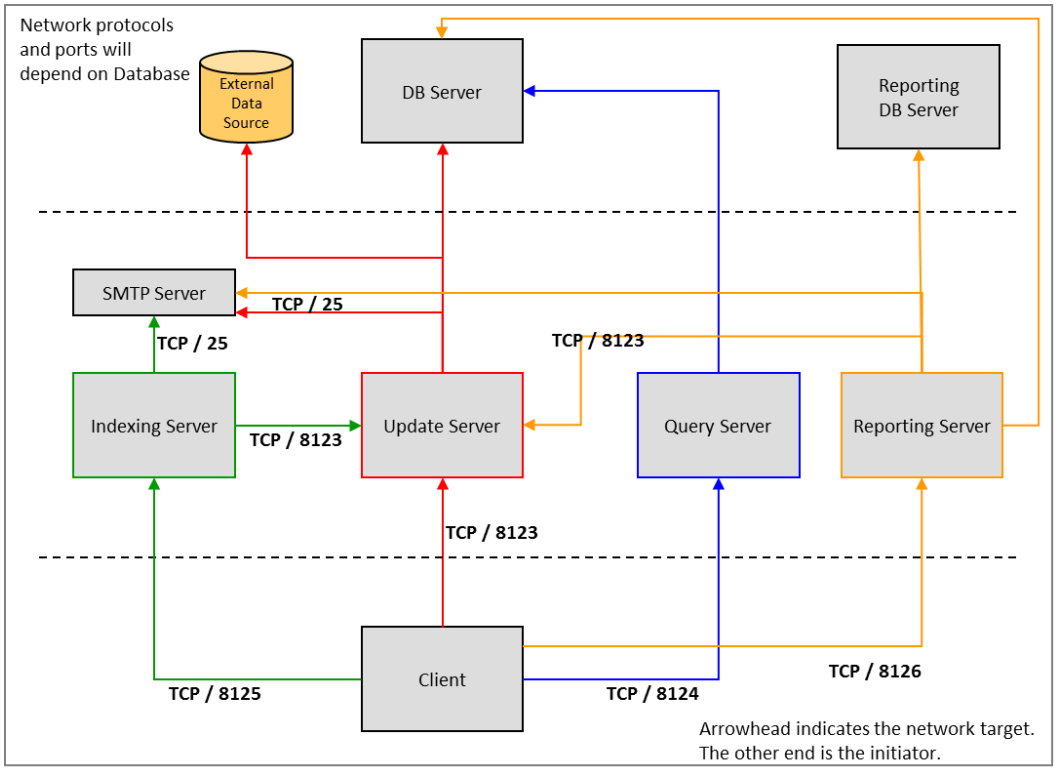

Figure 5 - Network Architecture

4.2.2 Network Traffic Throughput

It is not possible to predict network traffic as this will depend entirely on how many users are using the system, how frequently and the profile of their activity.

Cimera was initially developed to be usable over low bandwidth broadband and the amount of data requiring sending to / from the Client reduced as much as possible (but without using compression to avoid impacting performance). Cimera sends almost all of its data in a binary format and avoids bloated XML/SOAP.

The majority of Cimera CRUD (create read, update, delete) activity will involve minimum data transfer (a few KB) but any Item attachments that are stored or fetched will be transported, with no additional file compression, across the network and these may of course be KB to MB in size.

Cimera limits search results to a configurable maximum number of Items (default 500) to prevent accidental requests for large amounts of data which will put unnecessary load onto the DB server, the Query Server and the network. A maximum (500 Items) query response size would typically be in the region of 1-2MB. However this will depend on the number of attributes an Item has and the amount of data stored in each attribute.

The first time a user logs in to a Cimera Database the Client will download a full copy of the Metadata and store it in a Local Cache. This is typically under 1MB in size. Any further changes to the Metadata will be synced with the Local Cache and will be trivial in terms of network traffic.

4.3 Backup & Recovery

There are three elements that should be regularly backed up:

- Cimera Database (for example in SQL Server)
- Item Library

- Index & extracted text

The database must be backed up before the Item Library. It is expected that the customer will already have backup mechanisms and policies in place to enable point-in-time recovery (PITR) of the database. The Item Library may be backed up by simple file copy.

There is no mechanism within Windows or Cimera to allow point-in-time recovery of the Item Library and so this should be stored on resilient hardware. Cimera never deletes from, or updates members of, the Item Library and providing file level security prevents access to other users then there should never be a concern of accidental corruption occurring.

The Index related data may be backed up by simple file copy.

For maximum safety the Indexing Server (Windows service) may be stopped for the duration of the index backup (cold backup). There is no requirement to stop, or benefit in stopping, the Update or Query Servers.

Bear in mind that the index data is derived data and can always be rebuilt from the Cimera Database and Item Library.

The SQL Reporting Database can be backed up but it is not essential as it can relatively quickly be rebuilt during a short outage of the main Cimera service.

4.4 System Requirements

4.4.1 Cimera Client

Supported Operating Systems	Windows XP SP2+, Windows Vista, Windows 7, Windows 8.x Both 32 and 64 bit variants supported *Note: Best-endeavours support applies to platforms which are no longer supported by Microsoft*
Architecture	x86 (32 bit)
RAM	As per OS minimum requirement
CPU	As per OS minimum requirement
Disk Space	50MB
Pre-requisites	.NET Framework version 3.5 (full, not Client Profile)
Packaged as	MSI (Windows Installer)

4.4.2 Cimera Server

Supported Operating Systems	Windows 2003 Server, Windows 2008 Server, Windows 2012 Server Both 32 and 64 bit variants supported *Note: Best-endeavours support applies to platforms which are no longer supported by Microsoft*
Architecture	x86 (32 bit)
RAM	The higher of OS minimum requirement and 2GB (for up to 50 registered users) Add 512MB for each additional 50 users
CPU	The higher of OS minimum requirement and 2GHz (for up to 50 registered users) Add 1GHz for each additional 100 users
Disk Space (excludes data)	30MB
Pre-requisites	.NET Framework version 3.5
Packaged as	MSI (Windows Installer)

Note: Cimera Server installation package includes all three Cimera Servers and the figures above are for all three running on the same server.

4.4.3 DB Server

Supported DB Servers	SQL Server 2005 and later *Note: Best-endeavours support applies to platforms which are no longer supported by Microsoft*

4.5 Data Storage Requirements

These are typical figures based on a sample DB with 10,000 Items and 7,500 Item Library members.

Cimera Database	55KB per Item
Cimera Item Library	150KB per file
Cimera Indexing Server	15KB per Cimera Item Library member
Cimera Reporting Database	45KB per item

5 SECURITY ARCHITECTURE

5.1 Authentication and Access Control

Cimera Users have to be registered within Cimera.

Users can be designated as Cimera Administrators. New users may only be registered by an Administrator.

Cimera has its own native authentication and password mechanisms however Cimera can integrate with Windows SSPI / NTLM so that password authentication is managed by the Windows Domain.

Cimera native authentication does not enforce any password policies (length, complexity, history or expiry). If Cimera authentication is integrated with Windows Domain security then any group policies set at that level will apply.

5.2 Communication Security

Passwords are never sent across the network in clear text. They are only ever sent across the network having been subjected to a one-way hash.

Cimera sends all other data in clear text.

Any requests sent to the Update Server are digitally signed using a one-way hash. This digital signature ensures that the request has not been tampered with and that it comes from the identified user. Each request also includes a time-stamp which the server records to prevent replay of messages (each request must have a time-stamp greater than the last)

For performance reasons read-only requests to the Query Server and Indexing Server are not subject to any authentication (all Cimera users should be considered to have read access to all data although the Client may prevent read access via the GUI).

5.3 Application Security

In general Cimera allows read access of all Items to all registered users however users can be denied access to particular “Buckets” (buckets are used to group Items – this could be for example by Project or by functional area) but in general it should be assumed that all users can read everything.

Cimera is very granular in determining create / update / delete privileges. Cimera Users are assigned to one or more Groups via one or more Roles.

Access to Items or to individual attributes may be restricted according to User / Group / Role in addition to the Lifecycle state of an Item.

A user designated as a Cimera Administrator can manage the entire Cimera configuration, including Security Profiles. However a Cimera Administrator is still subject to standard application security. This prevents accidental damage but will not prevent wilful damage as the Administrator can change the security rules to permit themselves access.

5.4 Application Audit

An audit trail of all creations / updates / deletions is maintained in the Cimera Database, identifying the user, the date & time of the activity, what Items were affected and the attributes that were modified.

5.5 Alerting and Logging

The Cimera Servers write significant events to the Windows Event log. Application components such as the Messaging Manager, Indexer and Scheduler maintain more detail in separate logs. Cimera may also be configured to send email notifications when certain events occur such as shutdown / startup or scheduled jobs completing (successfully or in error).

Any errors (either returned to the user or at the back-end server) have a wealth of diagnostic information. Cimera will never produce a simple pop-up “there was an error”.