



Calopus Solutions

Calopus Integration Fundamentals

Version 4.x

Calopus Integration Fundamentals, Version 4.x

Part No: CALINT002-1

Copyright © RCL SoftNet Ltd 2005

All Rights Reserved

The information in this document is subject to change without notice. If you find any problems in the documentation, please report them to us in writing. RCL SoftNet Ltd does not warrant that this document is error-free.

Calopus and Calopus HE are registered trademarks of RCL SoftNet Ltd.

All other product or company names mentioned are used for identification only and may be trademarks of their respective owners.

Contents

ABOUT THIS DOCUMENT	5
INTERFACING OVERVIEW	6
WHAT IS APPLICATION INTERFACING?	6
WHAT ARE THE ISSUES WITH INTERFACING?	7
WHAT TYPES OF SOLUTIONS ARE AVAILABLE?	7
SELF-BUILD INTERFACES	7
VENDOR MIDDLEWARE (MESSAGE BROKERING INCLUDING SERVICE-ORIENTED ARCHITECTURE SOA)	7
VENDOR MIDDLEWARE (EXTRACT, TRANSFORM AND LOAD)	7
VENDOR MIDDLEWARE (ENTERPRISE APPLICATION INTEGRATION)	8
DEPARTMENTAL SILOS	8
ENTERPRISE RESOURCE PLANNING (ERP) OR SINGLE VENDOR SOLUTION	8
WHAT ARE THE GOOD AND BAD POINTS OF THESE SOLUTIONS?	8
SELF-BUILD INTERFACES	8
VENDOR MIDDLEWARE (MESSAGE BROKERING AND SOA)	9
VENDOR MIDDLEWARE (EXTRACT, TRANSFORM AND LOAD)	9
VENDOR MIDDLEWARE (ENTERPRISE APPLICATION INTEGRATION)	10
DEPARTMENTAL SILOS	10
ENTERPRISE RESOURCE PLANNING (ERP) OR SINGLE VENDOR SOLUTION	10
WHERE DOES A DATA WAREHOUSE OR A GENERIC REPORTING TOOL FIT IN?	11
WHAT FACTORS DRIVE WHICH APPROACH IS TAKEN?	11
POLITICAL OR STRATEGIC FACTORS	11
COST FACTORS	12
SKILL AND RESOURCE FACTORS	12
TECHNOLOGY FACTORS	13
INDUSTRY FACTORS	13
TACTICAL FACTORS	13
THE CALOPUS APPROACH	14
WHAT IS AND WHO ARE CALOPUS?	14
WHAT APPROACH DOES CALOPUS USE FOR INTEGRATION?	14
WHAT SKILLS DO YOU NEED TO USE CALOPUS?	15
HOW DO YOU GET FAMILIAR WITH USING CALOPUS?	15
WHAT TECHNOLOGY IS CALOPUS BASED ON?	16
WHAT SYSTEMS CAN CALOPUS COMMUNICATE WITH?	16
HOW CAN CALOPUS MAXIMIZE DATA QUALITY AND CONSISTENCY ACROSS SYSTEMS?	18
CHANGE THE RECORDS RETURNED AT THE EXTRACTION STAGE	18
PERFORM A SERIES OF CHECKS AS THE INTERFACE STARTS BUT BEFORE DATA IS EXTRACTED	18

PROVIDE TEST EVENTS AND PROCESS CONTROL DECISIONS FOR EACH CHANGED RECORD DETECTED	18
PERFORM A SERIES OF POST INTERFACE CHECKS	18
CONTROL THE ORDER IN WHICH GROUPS OF INTERFACES RUN	18
ONE SOURCE, MANY TARGETS	19
HOW QUICKLY AND HOW OFTEN CAN DATA BE TRANSFERRED?	19
CAN CALOPUS TRANSFER BINARY DOCUMENTS BETWEEN SYSTEMS?	19
HOW DOES CALOPUS PROVIDE SECURE ACCESS TO DATA?	20
WHAT DOCUMENTATION DOES CALOPUS PRODUCE ABOUT INTERFACES?	20
HOW IS CALOPUS GOING TO DEVELOP IN THE FUTURE?	21

About This Document

This guide has been produced to demonstrate the features of the Calopus Interface Designer toolkit. Because Calopus offers a rich set of functionality in addition to interface design, this document describes other features of Calopus, as and when appropriate, to help understanding of how they relate to the Interface Designer.

It shows the capabilities of the Interface Designer and also gives practical examples of how to solve common interfacing issues.

It also gives an overview of general integration requirements and issues as well as types of approaches that can be employed to address them. This document aims to provide an informative and balanced view of these challenges and also the relative strengths and weaknesses of each approach.

It is intended to assist the process of evaluating the Calopus Interface Designer against other approaches and vendor solutions by explaining what Calopus is, what it does and how it is placed in the myriad of other approaches.

This guide is based upon the features of version 4.x of the Calopus Solutions Interface Designer product.

Interfacing Overview

This section gives a brief overview of application interfacing. It also describes the challenges and possible solutions to interfacing and attempts to provide a dispassionate analysis of the various models and strategies for addressing application interfacing.

What is Application Interfacing?

As organisations continue to expand their use of information systems an ever increasingly large and complex set of data is being held in computer systems. Managing information across a large set of independent applications is a major challenge for businesses, particularly as there are areas where different applications hold overlapping datasets and these need to be synchronised in order for key business processes to function correctly. For example within Higher Education a student's name and address will be held by at least the student record system, finance system and library lending system. Given the data volumes involved manually entering these details into all systems is not viable but failure to reflect a change of address known to the student system but not the finance could result in delayed or even lost revenue. Therefore each institution needs to devise a coherent strategy for automatic and timely updating of one system using information already entered into another. This form of electronic data interchange (EDI) is commonly referred to as application interfacing.

Interfacing typically consists of three phases:

- Extract
- Transform
- Insert

The *extract* phase identifies the information required e.g. name and address and extracts it from the source system, in the above example most usually the student record system.

Because different systems often store the same information in different ways it may be necessary to *transform* it before the receiving system can use it. In our example it could be that the student record system holds the student name as two different fields 'surname', 'forename' while the library system has a single field 'name'. In this case the simple transformation is to concatenate forename and surname into a single field.

The extracted and transformed data is then *inserted* into the receiving systems data structures taking care to observe the application's rules for data consistency.

The complexity of interfaces can vary from the trivial, as in the example above, to the extremely complex. As well as successfully transferring data, interfaces need to be able to report what they have done and most importantly highlight instances where the data transfer failed to function correctly.

What are the issues with interfacing?

In the interfacing arena all organisations face a number of common business and technical issues:

- Multiple heterogeneous systems;
- Multiple database and application technologies;
- Multiple system vendors all with their own different interfacing capabilities;
- Increasing requirements for data accuracy and timeliness;
- Differing levels and effectiveness of data input validation across systems;
- Different data structures for the same information across systems;
- The requirements for a single "truth" of data at a point in time for management information and reporting....

These issues are common across many industries and regions.

What types of solutions are available?

There are a number of ways of tackling these issues:

Self-Build Interfaces

This is a common solution applied with different levels of sophistication. The local IT department either hand-builds individual interfaces as they are required or develops some kind of common approach and applies it across the organisation.

Vendor Middleware (Message Brokering including Service-oriented Architecture SOA)

This involves purchasing a vendor solution which uses a queuing system to deliver pre-formatted messages from one pre-defined system to another. Service-oriented Architecture is a recent development of message brokering that allows XML document messages to be published and distributed between systems using an Enterprise Service Bus (ESB).

Vendor Middleware (Extract, Transform and Load)

This involves purchasing a vendor solution which uses a common approach to configure interfaces and monitor their progress as they move data between systems. The Calopus Interface Designer tool would fall into this category.

Vendor Middleware (Enterprise Application Integration)

This involves purchasing a vendor solution which leaves the data where it is and provides an application "overlay" to combine together data and processes into combined screens. The Calopus Data Manager and Workflow Designer tools would fall into this category.

Departmental Silos

This is more common than one might think!...Keep the systems running heterogeneously and use double keying and desktop products such as MS Excel and MS Access to provide cross reporting on an ad hoc basis.

Enterprise Resource Planning (ERP) or Single Vendor solution

Buy a large and configurable vendor ERP solution with an integrated internal database model and use it to provide a reasonably large amount of core business processes, any systems outside that framework may or may not be integrated by some of the other solutions above.

In practise, a variety of these approaches could be found employed within most large organisations.

What are the good and bad points of these solutions?

Each of these solutions has a number of advantages and disadvantages:

Self-Build Interfaces

Self-build is very resource intensive and the sophistication of the solution is dependant on the technical vision and skill of the designers and developers that create it.

Generally, self-build interfaces are 'point-to-point' i.e. direct from one application to another and are commonly implemented via extracts of data to a flat file, movement of that file to a target system and then an upload and translation of the data into a suitable format for the target system. Data transfer is usually an overnight processes and therefore a time lag is introduced into the process of synchronising data. Business logic is of necessity built into the application itself, thus each interface requires the development and testing of significant amounts of new code and the resulting interfaces are therefore inflexible to change and are often not well documented. It is also common for the interface to reflect the developers own 'style' therefore as staff change maintenance and development becomes an issue.

Alternative methods include "trigger" based solutions that attempt to transfer changes in a source system via a database trigger. This approach is generally thought of as too intrusive to the source system and requires very intimate understanding of the database and business functions. In some cases application vendors will refuse to honour support contracts if their applications have been modified in this way.

However, self-build does empower the internal IT department with total control over the process of designing, building and running interfaces.

Vendor Middleware (Message Brokering and SOA)

This solution requires trigger points and receiving processes to be coded into the source and target systems in order to create and decode the messages that it receives and delivers. These messages are also fixed in format and can be costly to change as the structure of these messages must be decoded by all systems that use it.

Often a message brokering solution is supplied by two or more application suppliers who have entered into a strategic alliance. While extremely efficient these interfaces are inflexible as they represent a supplier led view of the business process and do not take account of local factors.

Much hype currently surrounds the Service-oriented Architecture (SOA) concept that has risen in popularity recently. SOA is a design concept that is most easily placed in this category due to its reliance on application trigger points when considered as an integration solution. Full adoption of SOA is not a trivial matter as most vendor applications do not yet support this concept. To implement SOA you will need to first implement Enterprise Service Bus middleware to facilitate the transport of XML messages. Then you will need to purchase or develop connector software for each application system that you wish to integrate. This software either builds and publishes or requests and decodes fixed XML document structures to and from the Service Bus.

SOA integration, therefore, has many points of software failure and is resistant to change due to its reliance on fixed format XML documents and its need for connector applications and pre-defined trigger points.

The SOA paradigm is a compelling concept. It supports a general industry desire to support a unified business process model that ties together disparate applications.

Vendor Middleware (Extract, Transform and Load)

This approach requires capital investment and a good working relationship with a middleware vendor. Depending upon the solution chosen, levels of open architecture, commonality of interface design and minimal intrusiveness are enhanced.

A good solution will be based on a Meta data architecture and will be able to extract once, transform and load into many target systems at the same time and provide total visibility and audit of the whole process. It will also provide the ability to detect, trap and correct data inconsistencies and deliver the data on a flexible time schedule, possibly even down to less than one minute from change.

These solutions are designed to provide commonality of approach, ease of definition and to enhance data quality, coordination and timeliness of transfer.

Some of these solutions can only communicate with known applications using “connector applications” and require significant extra capital investment for interfaces that are new to their framework.

Vendor Middleware (Enterprise Application Integration)

This approach also requires capital investment and a good working relationship with a middleware vendor. It works at the application, rather than the database level, by providing an overlay that can either replace or operate your existing applications user interfaces.

A good solution would provide a user interface that is common in look and feel across all applications and can enhance these applications with common features (such as audit or workflow features) that may or may not be available in the existing systems.

This approach can also require a significant server investment to operate, as your application users will be using the middleware’s user interface. Also, the loss, temporary failure or planned maintenance of one underlying system could completely cripple a critical business process which is dependent on it (the other systems may not be able to safely complete their work independently).

Again, for minimal capital expenditure, the cost justification of this approach may be dependent on your application being “known” to the vendor middleware.

Departmental Silos

In organisations with a high level of departmental autonomy or isolation, this may actually be a highly valid approach.

It will always have the disadvantages of inconsistent data across an organisation and a confused version of data for senior management coupled with highly inefficient data capture processes.

Enterprise Resource Planning (ERP) or Single Vendor solution

An ERP solution with a good functional fit to an organisation, or one that can be easily customised on an ongoing basis can provide a business critical amount of enterprise integration, just due to it’s common user interface, database model and sharing of common business processes.

However, every other system outside of this framework is commonly then seen as a poor relation to the ERP by vendors and internal staff alike. Some ERP solutions do not easily support interfaces from outside their framework, requiring you to adopt another of the previously described about to provide 100% integration across the organisation.

Also, after an ERP solution is “bedded in”, gaps in functionality often become apparent and a number of business processes to fill such gaps may be needed. These gaps may establish requirements for new systems in their own right and would need to be included in an integrated approach through interfacing of some kind or other.

Where does a Data Warehouse or a Generic Reporting tool fit in?

Data warehouses are generally used to provide a constant view of data for Management Information purposes at a specific point in time. They sometimes have the ability to provide historical trends in data and can provide the ability to provide greater detail and to identify trends and causal relationships.

Some organisations use generic reporting and analysis tools, such as Business Objects, to provide direct access to operational systems and combine this data into a user definable Management Information reports. This approach guarantees that the data used for reporting purposes is up to date. However, accessing operational systems online often degrades the performance of those systems and sometimes has to be scheduled for out of hours. This approach by its nature can also provide a moving target where the same report submitted by two different users at a similar time produce two different results.

A Data Warehouse is, however, a standalone source of information that generally operates with data at a pre-defined and constant period behind real time. Therefore, a reliable mechanism needs to be employed to extract the relevant data from operational systems into the Data Warehouse. Some Vendor Data Warehouse solutions do provide mechanisms to perform those extractions.

While application interfacing and data warehousing are logically completely separate the extraction of data into a set of independent structures is a very similar process for both. A repository built for interfacing may present significant opportunities for warehousing and management reporting or separate repositories can be built for interfacing and warehousing using the same toolset.

What factors drive which approach is taken?

The key factors that drive this choice can be split into:

Political or Strategic Factors

Integration of corporate systems is quite often discussed at many levels of an organisation and not only within the IT area. Lack of effective data integration increases system administration demands, reduces overall satisfaction across the business in the IT portfolio and can be felt at all levels. It also leads to a fragmented view of the overall business process because key information is tied up in "data silos".

Therefore, the decision to adopt a vendor solution may be taken as a way of decreasing an organisations dependency on scarce and/or overstretched resources within an IT department.

It could also just be as simple as recognising the need to generically improve services and information consistency and availability.

In general, all common and repeatable interfacing approaches that deliver an improvement in data quality and timeliness and that have the strategic and political backing of an organisation at the highest levels have a good chance of success.

Cost Factors

The costs of poor quality and late information delivery are very difficult to quantify as it is rare (but not impossible) that a single incident can create a significant and noticeable result. These costs are a drip feed of expense that is tolerated at the operational level as normal and disparately spread across the whole organisation. As nobody is able to take a holistic view of these costs and each example is small, the total effect is hidden.

With this in mind, a good reason for delivering a solution can be generally found from the list of benefits that improved integration provides.

The choice to build in-house or to buy in a vendor solution is very often influenced by cost.

Vendor solutions can cost from around £20K to £200K and then incur an annual maintenance charge of around 20% of that price. Some vendor solutions may appear initially cost effective, where the software licenses are low. You do, however, have to carefully consider if your particular collection of internal systems would require extensions to be coded by the vendor and how much paid consultancy is required before full integration is achieved. Other vendor solutions are more expensive to license, but have more features or are more likely to provide a fixed and known full project cost.

What vendor solutions can provide is a rich and tested product and feature set that support existing interfacing needs and also supports the systems and technologies that are currently emerging and would be difficult or expensive to incorporate into home grown solutions. This is because, to survive in the marketplace, such solutions are developed to incorporate new requirements and technologies easily and quickly through adopting META Data and/or Open Technology platforms.

Comparing self build with a vendor solution can therefore be difficult. To plan and cost a self build solution that can compete with a good vendor solution on a feature to feature basis is outside the budgetary scope of most internal IT organisations. Therefore, self build can be justified as cost effective but can be restricted in its flexibility, scalability and commonality due to limited funding.

Skill and Resource Factors

Any integration solution is limited by the skill of its designers and developers. It is common for organisations to be so busy supporting their existing disparate integration model that they cannot devote resources to building a better solution. There may also be a shortfall in relevant or available skills to build the solution in-house.

Vendor solutions should have been built by a highly skilled set of individuals with specialist knowledge of creating integration systems and have formed a goal to develop a better solution outside of a constrained internal IT environment – though this is not always apparent!.

However, some IT departments will always prefer to build a solution rather than adopt a vendor product due to their corporate or IT philosophy.

Technology Factors

As many organisations move towards purchasing “Best of Breed” vendor solutions for core IT systems and processes, it is common to find that they have a substantial technology mix. Differing database, application and delivery technologies can be difficult to reconcile into a coherent and integrated whole. Many organisations are following the IT industry trends towards open technologies and web browser delivery of user interfaces. The decision to adopt a vendor solution can be highly influenced by the need to adopt these trends into the organisation. The self build model is likely to require retraining or buying in of specialised skills.

Industry Factors

The need to improve data completeness and internal integration may well be driven by industry reporting, compliance or cross organisation data-sharing requirements. As these requirements change over time, it can become difficult for an organisation with limited resources to keep up with changes effectively. There may be vendor add-ons products that provide a quick and standard way of delivering these requirements.

Tactical Factors

If the goal is to create consistent interface mechanisms across the whole organisation, adopting a vendor solution may limit the scope of individuals to diversify the ways in which they solve their interface requirements. This will make the total integration effort easier to maintain...but it may also produce internal resistance to the introduction of a common, but different, approach.

The Calopus Approach

This section describes the approach that Calopus takes to solving integration requirements.

What is and who are Calopus?

Calopus is an Application Development Framework that allows information held in disparate databases and systems to be synchronized without altering the database structure or application logic. Information transfer can be combined concurrently across many systems. New business processes and forms applications can be rapidly and easily configured to provide value added functionality.

Calopus is a partnership between a software house, RCL SoftNet Ltd, and Leeds Metropolitan University.

The responsibilities within this partnership are as follows:

RCL SoftNet Ltd own, develop and maintain the Calopus software and provide on site training and second line support.

Leeds Metropolitan University deliver the main contractual relationship for Calopus clients, provide server, network and office for Calopus staff and maintain and provide first line support via a Calopus support website (www.calopus.com).

Both organizations join in a marketing, conference and pre-sales effort.

What approach does Calopus use for integration?

Calopus is a very flexible tool and adopts a hybrid approach to integration. That is it provides different strategies depending on the business requirements:

- Calopus will use it's Extract, Transform and Load features where they best fit the requirements;
- Calopus may also use it's Enterprise Application Integration features when this approach is more suited or the information may not be duplicated;
- Calopus could also configure Workflows, eForms and role secured application areas to assist in the integration and data cleansing process where required;

In addition Calopus will also:

- facilitate the definition of, loading of and ongoing maintenance of data warehouses to logically combine information from different applications into a single "point of truth" for management information and P.I. reporting etc;
- provide a common web user interface, designer tools and a common structure to deliver all features.

As Calopus is a hybrid integration platform and has the ability to receive and post HTTP requests containing XML documents, it can be used to fulfil the Service Bus middleware aspect of a SOA approach.

What skills do you need to use Calopus?

Calopus is based on Oracle database technology and uses visual design tools to create and maintain interfaces. To successfully use Calopus an institution needs:

- An oracle database administrator to install, backup and maintain the Calopus database installation and also to create the database links that allow Calopus to communicate with other Oracle or non-Oracle databases;
- An interface designer who understands the business processes involved and is competent to create RDBMS table structures and relationships and also can write at least a basic level of SQL and PL/SQL ;
- Familiarity with the features of the Calopus Interface Designer tool.

How do you get familiar with using Calopus?

Calopus uses a collaborative approach to training. That is:

- We provide an on site consultant to assist in creating "real" interfaces from day one. The goal is to make a client interface designer resource 90% self-sufficient in operating the Interface Designer toolkit before a consultant departs;
- We do not believe that a client should incur extra cost from us before we have personally trained their staff to be 90% autonomous;
- We do not use documentation, such as this manual, as our primary training materials. We use personal communication and assistance as the primary method. This ensures that client resources not only have the theoretical knowledge of how to use the Interface Designer features, but also an enhanced understanding of how of best achieve results using the many methods available with the Calopus framework;
- We have a Web Site that allows technical queries to be logged by clients and worked by our support staff;
- We have established a user group where clients can exchange ideas and bests practices.

Calopus recommends that an interface designer commits to about 2 weeks of consultant supported "on system" training spread over a period of no more than 3 months. Experience shows that this commitment enables this resource to be fully operational and mostly autonomous after this has been completed.

We encourage this resource to fully document their experiences and also to amend the Calopus Help System with any client specific details that they perceive and relevant. Materials such as this manual are delivered in MS Word format within the Calopus system to facilitate this and new versions can be easily uploaded into a product installation.

What technology is Calopus based on?

Calopus is based on Oracle 9i or 10G database technology. Calopus is a collection of PL/SQL database objects all contained within a single oracle instance. It utilizes the apache web services supplied with these database versions and delivers it's web content to a web browser using the mod_plsql apache feature.

Services to other database technologies are provided using the heterogeneous gateway services provided by Oracle. This enables Calopus to treat a remote SQL Server (or other SQL based system) database instance exactly the same as a remote oracle database instance and therefore deliver interfaces across different database technologies.

The Calopus web applications that include the Interface Designer are delivered to the browser as web native HTML and JavaScript pages.

Any XML, LDAP or DCOM communication is delivered seamlessly via an internal Oracle database Java Virtual Machine call from PL/SQL. This method also allows host system programs to be executed.

What systems can Calopus communication with?

Calopus does not use "connector" applications to communicate with source and target databases. All Calopus needs to communicate with a database system is the META Data that describes the structure of the data within the source system and a database link to access it. The configuration information is held and maintained within the Calopus Data Manager which underpins the Interface Designer toolkit.

This approach enables Calopus to connect to a wide range of database technologies in a non-intrusive manner and simplifies both the install process and ongoing maintenance of Calopus and the interfaces you configure within it.

Calopus has been operating within the Higher Education sector and therefore does have working experience of most of the common Student Record systems in that sector. However, based on client experiences, Calopus does not provide a "turn key" solution for these systems as the manner in which these systems are configured and operated by individual institutions are sufficiently diverse to render this approach meaningless.

As well as direct database updates Calopus can also receive, decode and build, transmit XML documents, read and write operating system files, send SMTP and SMS messages and communicate

with LDAP and DCOM services. This collection of capabilities allows Calopus to communicate in some way or another with most source and target information systems.

How can Calopus maximize data quality and consistency across systems?

A key factor in application interfacing is that incorrect or incomplete information in one application is not promulgated across the organisation, Calopus puts the organisation in control of rules that determine whether data is transferred or requires correction in the source system before transfer can take place. When defining an interface within the Interface Designer, there are a number of places where these rules can be applied:

Change the records returned at the extraction stage

Calopus allows you to consider all records on a source table or source view or apply an SQL "Where Clause" to the process that restricts what data records are considered. Any example of this is "when extracting the accounting transactions for a person, extract only those transactions for people who have had new transaction activity over the last week". This clause could also be used to omit records where data is not valid.

Perform a series of checks as the interface starts but before data is extracted

Calopus allows you to optionally add a piece PL/SQL code that performs checks or setup details before an interface is run. This code can access and check any information that the Calopus database can "see" in the source schemas of other databases.

Provide test events and process control decisions for each changed record detected

Once the changes in a source data set are identified by Calopus, a series of events can be defined that provide the ability to make decisions on the validity and state of that changed data record. You can define a number of paths that are applicable to the state of the data and also choose to re-evaluate a record in the next interface run until it is corrected.

Perform a series of post interface checks

Calopus can also optionally allow you to add a piece PL/SQL code that can perform checks or post transfer details after an interface process changes information, but before the interface finishes a run. This code can access and check any information that the Calopus database can "see".

Control the order in which groups of interfaces run

It is common that a number of interfaces from different source systems are logically related to a single data item, such as a person identifier. Calopus can group these interfaces together and run them in strict sequence when a changed source record is detected in any of those interfaces. This ensures that pre-requisite data is processed before it is needed.

For example, if interface A is the master source for a person and Interface B contains some additional information. If interface B detects a change, Calopus will first fire Interface A for that ID only and then Interface B for that ID only until all changes are processed.

One source, many targets

The consistency of data across systems is enhanced by the ability of Calopus to detect changes in source data and then transfer that change to all relevant target systems in one interface. Calopus will let you control whether data is committed or back out if any other those transfer events fail allowing you to be sure that, where possible, data is delivered to all systems or none. As previously discussed a simple example is the need to maintain a single student address from the student record system across multiple applications such as Finance, library and HESA.

How quickly and how often can data be transferred?

Calopus scheduling is very flexible to meet most business requirements and can schedule interface runs on demand or on a resubmission schedule of anywhere between one minute and once annually. You can control the first start time and last submission time for a day and on which days of the week the interface runs. As a result interfaces run as often as the business requires them without placing an unnecessary overhead on the underlying IT infrastructure.

Calopus attempts to minimize the number of times it needs to access the Calopus META data configurations for information and also uses techniques to optimize the reuse of similar SQL queries so that Oracle can cache common query results and minimise the amount of parsing it needs to perform.

The speed of an interface run is dependant on a number of factors:

- General network speeds and bandwidth;
- Complexity and size of source data tables and views;
- Complexity, number and optimisation of defined processing events and tests within the interface;
- Server resource and power available.

To optimize performance, these four factors should be considered and tuned appropriately. Calopus can provide assistance in this process over the initial installation and consulting period.

As a benchmark, an interface that extracts 20000 records of which 200 are changed every time the interface runs with a simple push to a remote table can process all these records on the first run in about 5 minutes. Subsequence runs take less than 20 seconds.

Can Calopus transfer binary documents between systems?

Calopus can transfer Binary Long Objects stored in other Oracle instances or accessible as a file in a file system to any similar destination.

Calopus can also link images and documents to database records, for example a Photo and electronic Contract of Employment can be linked to a personnel record either held within Calopus or resident in a remote database as long as the record is viewed within a Calopus eForm or Workflow.

This feature is a powerful way of utilizing Calopus Enterprise Application Integration features to add binary document content and functionality to a system or application that does not currently support it.

How does Calopus provide secure access to data?

Another area of the Calopus framework that underpins both the Data Manager and the Interface Designer is the Security Manager.

The Security Manager holds Metadata about people, users and roles and how they are combined. Access to information in the Calopus web user interface is controlled by granting access to Metadata roles. These roles control which tables, menu items, eForms and Workflows a user can access and the data that is shown within them.

Vertical and horizontal data security means that the data items can be seen by eForms and Workflows can be restricted and which records are allowed through can be similarly limited by restrictions placed on user roles.

Role based record security can be applied to any registered table structure in the Calopus Data Manager. Further to this, as Calopus knows how tables relate to each other through their foreign keys, a secured record within a master table automatically restricts access to any child table that contains that value. For example, if you define a list of departments and assign a restricted list of only one of those departments to a role, any records about people who are not linked to that department will automatically be eliminated from any data retrieved in an eForm or Workflow process by a user who is assigned that role.

This means that Calopus can provide secure "windows" to the data.

Finally, roles can be arranged in a hierarchy, allowing different access to different sub roles within a application area and also allowing access to common processes and menu items to be inherited from a parent role.

What documentation does Calopus produce about interfaces?

The Calopus Interface Designer is designed to be a visual representation of what an interface does. It also includes a "Documentation Mode" for each defined interface that provides a textual interface definition report. Therefore, these screens are a documentation of an interface in their own right. Calopus suggests that printing these screens from a browser would provide enough documented detail to describe an interface and what it does.

Alternatively, if a corporate standard report is required, any reporting tool that can access Oracle could be employed to read the Metadata tables that store the interface configuration information.

How is Calopus going to develop in the future?

Calopus has already received useful and constructive feedback from clients that has been used to improve the usability and intuitiveness of the Interface Designer and the Calopus framework as a whole.

Calopus is committed to the following ongoing goals:

- Improving usability and intuitiveness of the Calopus framework;
- Reducing dependence upon technical expertise;
- Increasing relevance to business users and professionals;
- Exploring innovative ways of improving the features and scope of our Designer tools;
- Reducing the time and effort needed for a user to be autonomous with operating the framework;
- Identifying common approaches and providing Metadata structures that will allow them to be included in the Calopus framework;
- Identifying and including emerging technologies as they arise.