ClearPath Middleware Strategy and Products for MCP Systems

By Peter Bye

White Paper
Applications distributed among ClearPath® and other systems use service-oriented architecture (SOA) to provide a well-structured architectural framework.

This paper outlines the architectural ideas behind SOA and explains the role of middleware as the enabling technology. It then reviews the Unisys middleware strategy and describes the middleware products available with ClearPath MCP systems. The main body of the text summarises key product features. Technical briefs provide more detail about the technology and products. Case studies show how ClearPath MCP systems are using middleware in various configurations, including service architectures as service providers, consumers, or both. This paper includes products up to ClearPath MCP Release 16, with an indication of some future developments. A companion paper covers the same ground for ClearPath OS 2200 systems.

Although every attempt has been made to ensure that the information contained in this paper is accurate, it does not constitute a commitment to any specific product or functionality. Contact Unisys for current product information.
Contents
Introduction ........................................................................................................................................... 4
Distributed systems and service-oriented architecture ........................................................................... 5
ClearPath systems, SOA and Unisys middleware strategy ...................................................................... 8
  ClearPath systems and SOA .................................................................................................................. 8
  Unisys middleware strategy .................................................................................................................. 9
ClearPath MCP middleware: the products .............................................................................................. 10
  A taxonomy of middleware .................................................................................................................. 10
  Execution environments ....................................................................................................................... 11
  Integration middleware ....................................................................................................................... 12
  Other middleware ............................................................................................................................... 15
Middleware and ClearPath systems with fabric infrastructure ................................................................. 17
Using the middleware ............................................................................................................................. 19
  Case study 1 .......................................................................................................................................... 19
  Case study 2 .......................................................................................................................................... 22
  Case study 3 .......................................................................................................................................... 23
More information ..................................................................................................................................... 25
Technical briefs ......................................................................................................................................... 27
  Java ...................................................................................................................................................... 28
  ODTP .................................................................................................................................................... 30
  Web Transaction Server for MCP (WebTS) ........................................................................................... 33
  Web Services ......................................................................................................................................... 36
  ClearPath ePortal .................................................................................................................................. 41
  Message Queuing Middleware (MQM) .................................................................................................... 42
  File Transfer Protocol (FTP) ................................................................................................................ 46
  TXINT .................................................................................................................................................. 46
  Data access .......................................................................................................................................... 50
  Sockets .................................................................................................................................................. 55
  Advanced Development Environments (ADE) ....................................................................................... 57
Appendix: Glossary .................................................................................................................................... 60
Revision History ........................................................................................................................................ 67
About the author........................................................................................................................................ 67
Introduction

This paper contains a description of Unisys middleware strategy and products for ClearPath MCP systems: the Libra family. Middleware is the enabling technology for building distributed systems. Emphasis is therefore given to the role of middleware in integrating ClearPath MCP applications with applications in other environments, including integration in a Service-Oriented Architecture (SOA). Applications running in ClearPath systems are valuable assets, representing a considerable investment over a long period in many cases. Their reuse in new environments capitalises on this investment. It can reduce the time and risk involved in developing new applications, while retaining ClearPath strengths such as security and resilience. OS 2200 middleware facilitates application reuse and extension.

The aim of this paper is to provide general information about the middleware and how it is used, not to function as a manual. No attempt is made to describe all software related to ClearPath MCP.

The paper is divided into the following sections:

- **Distributed systems and service-oriented architecture** provides a brief introduction to the concepts and technologies of Service-Oriented Architecture. Readers familiar with the ideas behind SOA can skip this section.
- **ClearPath Systems, SOA and Unisys middleware strategy** contains a statement of the Unisys strategy for ClearPath middleware.
- **ClearPath Middleware: the Products** contains a summary of the ClearPath MCP system middleware technologies and products available to implement the strategy described in the previous section.
- **Using the Middleware** contains some remarks about the deployment of the middleware, illustrated with examples.
- **More information** lists some sources of further information.
- Additional information about selected technologies and products is provided in the section entitled **Technical Briefs**, which contain short descriptions of the products specifically implemented by Unisys. Pointers to the technical briefs are marked in the text. For example, a technical brief on Java is pointed to by [techbrief: Java].
- **An Appendix** contains a glossary.

Some of the middleware products available with ClearPath systems originate from other companies, which are often in partnership with Unisys. New products appear at short notice and may be usable immediately, thereby expanding the middleware portfolio. And existing capabilities may change or be removed. Those interested in the particular products described in this paper should check current availability and supported features with Unisys.

Over the years, many Unisys products have been renamed. This paper uses names likely to be most familiar to readers but also provides the current names as well on first reference. See the appendix for more information.
Distributed systems and service-oriented architecture

Distributed systems, where a number of separate component systems collaborate to deliver the application functions, are now common. Service orientation and SOA provide a framework for the construction of such system, building on a number of conceptually straightforward ideas. Services are delivered by service providers and requested by consumers. In IT terms, a service provider is a software entity delivering a specific function or functions. The consumer, or requester, is also a software entity, for example a web browser or other intelligent client in a PC, the software in a variety of mobile devices such as smart ‘phones and tablets, and increasingly, embedded systems, in vehicles for instance.

A service provider can be characterised as having the following attributes.

- It is independent of any requester. It has an existence of its own as a black box. This means it can be built using any language and run-time environment its creator chooses.
- The various service requesters and providers should be loosely connected. This is a corollary to the first attribute. Without loose connection, they are not fully independent of each other.
- It delivers a precisely defined set of functions and ways to invoke them, together with responses – that is, interfaces. The functions provided may contain business application logic or just provide direct access to databases or other corporate data.
- It has a name and can be located through some kind of directory structure if necessary, where the functions offered and how to invoke them are described. However, not all Web Services need to be accessed through a directory. If the requestor knows the location of the provider and how to access it, no directory entry is required.
- It responds to requests for its services without having any knowledge of an ultimate presentation of the information. This attribute allows it to be reused in different contexts.
- It is possible to replace an existing implementation with a new version and maintain backwards compatibility, without affecting existing users.
- It may contain mechanisms for recovering from errors in the event of a failure somewhere in the environment in which it is invoked, although this is not strictly a requirement.
- Finally, it may also be a requester of services delivered by other providers. This allows an original requester to invoke a service that is ultimately delivered by collaboration between several service providers.

Figure 1 is a schematic of a generic SOA, showing the major components of the architecture. The figure shows the consumer of services at the top. The services are delivered by the infrastructure below, which is divided into a number of components.

The access channel management component admits consumer requests through a variety of channels such as web browsers, intelligent clients, mobile devices and collaborating systems. As can be seen in the figure, it contains two sub-components. The first, user interface management, handles the connection to the access channel and will therefore vary depending on the nature of the channel. The second sub-component is service definition and mapping, which identifies the available services and links them to the appropriate service providers. The link could be as simple as a pointer. It could be much more sophisticated. Each external service should reflect a discrete business task. However, a discrete business task may not be reflected by a single free-standing transaction, screen, or web page from a single service provider. In this case, the service is exposed as a single entity with its discrete input and output. Internally, however, multiple interactions are likely to be performed on multiple systems.
Once there is a collection of services that reflect the business tasks, the business domain is able to connect them together into automated business processes. This is typically known as **orchestration**, which, in effect, creates applications. These applications can be viewed as **external service providers**, which in turn use a number of **internal service providers**, and possible **external partner providers** to deliver their services. The internal service providers could be distributed across a number of systems, with the service bus providing the linking infrastructure.

The **service bus** links the various elements together. It:

- Advertises service definitions in a repository;
- Virtualises service locations;
- Functions as a message broker; and
- Performs data transformation.

The service bus supports a number of protocols and provides facilities for conversion between them. If Web Services technologies as defined by the World Wide Web Consortium (W3C)\(^1\) were used, SOAP\(^2\) would be the interconnecting protocol. It supports the loose coupling and other attributes required to interconnect different technologies. SOAP is usually carried over network connections by HTTP, although the SOAP standard does not specify any specific protocol.

---

\(^1\) The W3C is a vendor-neutral organisation, led by Tim Berners-Lee, the inventor of the World Wide Web, and Dr Jeffrey Jaffe, the CEO. See [www.w3.org](http://www.w3.org) for more information about the organisation and its work.

\(^2\) SOAP, originally an acronym for Simple Object Access protocol, is a protocol specification for exchanging structured information in the implementation of Web Services. The expansion of the acronym is no longer used.
Message queuing could equally be used, for example. An alternative for Web Services is to use RESTful Web services. REST is an acronym for Representational State Transfer. For more detail, see [techbrief: Web Services].

However, the protocols and standards used should be chosen pragmatically, as factors such as performance, efficiency and existing environments may influence the choice. Choices not using Web Services include various forms of remote procedure call. The latter include the protocols supported in Oracle’s (formerly BEA Systems Inc’s) Tuxedo product and the Distributed Transaction Processing (DTP) model defined by the Open Group\(^3\).

Some service providers may not support the protocols available in the service bus, and may need to be modified or extended to implement them. An alternative is to use a connector, which adapts or wraps the application or database to become a service provider, converting the protocols supported in the service bus to a protocol acceptable by the service provider.

The service bus could be implemented entirely within each of the connected components, that is, service providers and access channel management. Each component would support the required protocols. However, this rapidly becomes complicated as the number of services and protocols increases. A better alternative is to implement the service bus within a free-standing server or servers. This approach creates a hub, which acts as a switching point between the various service providers and access channel management. It allows providers to use different protocols, with the hub performing conversions between them.

Systems management is the other element shown in Figure 1. Although it will not be discussed further in this paper, its importance cannot be over-stressed. Distributed environments rapidly become complicated, so tools for diagnosis of problems, gathering performance information, highlighting errors and automating management processes are essential. It is a serious error to try to save on costs by skimping on the provision of management tools. Without adequate systems management, immense amounts of money can be wasted trying to track down problems, whereas the right tools can lead to rapid problem diagnosis and correction\(^4\).

There are many off-the-shelf products available to create service architectures. Products such as portals and enterprise service buses (ESBs) provide much of the capability required. There are also services available, for example Unisys Application Modernization Platform as a Service (AMPS)\(^5\).

This paper is concerned with the technology required to implement service architectures. However, technology alone is not sufficient for successful implementations; organisational factors play a critical role. Organisational silos, within business units of some kind, or technologies silos such as mainframes and other system types, get in the way of progress. Collaboration across the organisation is essential. To make this happen, senior management commitment and a willingness to embrace change are critical to success.

\(^3\) See [http://www.opengroup.org](http://www.opengroup.org) for more information about the Open Group and the Open Group DTP model. Tuxedo, and the Open Group DTP model, which is based on Tuxedo, are excellent examples of SOA implementation: applications are constructed as services (the word ‘service’ is in fact used in both Tuxedo and the Open Group DTP model), which co-operate with each other. Note that while Tuxedo is a product, the Open Group model is a specification. Implementations of the specification, i.e. products, have been made by a number of companies, including Unisys.


\(^5\) For more on AMPS, see [http://www.unisys.com/search/unisys?k=AMPS](http://www.unisys.com/search/unisys?k=AMPS)
ClearPath systems, SOA and Unisys middleware strategy

ClearPath systems and SOA

ClearPath systems may be internal service providers. They are either compliant if they implement protocols in the service interconnectivity subcomponent, or they are non-compliant if they do not support any of these protocols and require wrapping with a connector.

The service definition and mapping component defines the internal services to be invoked and the sequence and rules for invoking them. Much of the application logic resides in the internal service providers. Existing systems, including ClearPath applications, may need some modifications, possibly including wrapping, to allow them to expose services and manage the requests.

However, acting as internal service providers is not the only way in which ClearPath systems can participate in SOA. The approach can be used to extend the capabilities of existing ClearPath applications. In this model, the applications remain in the ClearPath systems, which retain their role as the primary user interface. New facilities are implemented as service providers and accessed by the ClearPath systems, which are consumers. This is not just a theoretical possibility. There are examples among ClearPath users.

As an illustration of the approach, suppose a bank has two ClearPath systems: one contains a mortgage application while the other houses an account management application, handling savings and other accounts. The bank decides to implement the following new functions:

- Introduce a new customer relationship management (CRM) application, replacing inconsistent and duplicated information contained within the two product systems;
- Use an external credit reference system for checking new customers wanting mortgages or wishing to open an account;
- Allow the mortgage and account applications to obtain balance and other status information from each other; and
- Allow the account management system to transfer payments to the mortgage system.

A decision is made to use an SOA approach, with Web Services as the technology selected for service interconnectivity, to be provided by an off-the-shelf ESB product. The mortgage system can easily be modified to handle Web Services. However, the account management system cannot easily be changed, so it requires wrapping with a connector. Figure 2 shows the resulting architecture.

The two applications will continue to serve their current constituencies of users, maintaining the user interface as before. Something like the following is needed to support the new requirements.

- A connector has to be provided for the account management system;
- The two applications need modification to use the new CRM system rather than internal customer information;
- The applications have to be extended to include a request to the credit check application when creating new accounts; and
- Changes are required in both applications to allow the exchange of status information and to allow the account management application to transfer funds to the mortgage system.

The two ways in which ClearPath systems can participate in SOA environments are not mutually exclusive. In Figure 2, the two ClearPath systems extend their functionality by using the new shared services, but also act as service providers to each other for the status information and funds transfer requests. ClearPath middleware provides many features to
facilitate incorporating existing and new applications in either or both of these roles. The next section explains the available technologies and how they can be used.

Figure 2: Shows a bank with two ClearPath systems, which share some common services and communicate with each other, using an SOA approach

**Unisys middleware strategy**

Openness is a guiding principle underpinning the Unisys strategy for ClearPath systems. The strategy regarding SOA is therefore to provide the middleware necessary to allow ClearPath to participate in distributed SOA environments using any of the major open and *de facto* open standards. This is in fact a continuation of Unisys's long-standing commitment to middleware implementation.

The strategy calls for two sets of developments. The first set provides open application execution environments, referred to here as *application servers*, on MCP platforms, including integration with the native COMS (new name: Transaction Server), and databases such as DMS II (new name: Enterprise Database Server for ClearPath MCP). Particular emphasis has been put on comprehensive support for Java. The availability of open application servers on ClearPath systems, especially Java application servers, greatly extends the options available for developing or acquiring new applications.

The second set of developments in the strategy is the implementation of integration middleware to allow applications in ClearPath systems to collaborate with others in an SOA environment. These environments may contain systems using different technologies in what are called *composite* applications. A range of middleware technologies is therefore required.

A key factor in the implementation of the strategy is the use of Open Source products, extended where necessary and tightly integrated with ClearPath systems. This approach allows Unisys to deliver products much more cost effectively and in a shorter timeframe than developing from scratch. Examples of the approach include Java application servers and integrated development environments.
ClearPath MCP middleware: the products

This section contains a summary of the middleware products available with ClearPath MCP systems. Extra technical details are contained in the section ‘Technical briefs’. For the purposes of this paper, the various middleware technologies have been classified into a small number of different types; this taxonomy is explained in the following subsection.

A taxonomy of middleware

Figure 3 shows the middleware technologies available with ClearPath systems. The upper part of the figure shows a ClearPath MCP environment. It contains execution environments for transactional and batch applications. They are referred to as application servers in the figure and include COMS. Applications in COMS and batch programs access the various databases. These environments are typical of MCP user, implementing secure, high performance systems in a variety of commercial and public sector environments.

Middleware technologies can be divided into three general classes:

- **Execution environments** provide the facilities required to execute applications. The class is divided into two subclasses: open distributed application servers, which support distributed applications using a common technology, and include facilities for access to native applications such as COMS and other application environments.
- **Integration middleware** enables the connection of different technologies. It is divided into two subclasses: loosely-coupled middleware and environment integrators.
- **Other middleware** contains the products that do not really fit into one of the other two classes.

Consider each in turn, starting from the left of Figure 3 with the execution environments.
**Execution environments**

*Open distributed application servers: Java*

Java is ubiquitous, with a massive range of products and services available. Unisys has accordingly made significant investments in support of Java in conjunction with ClearPath systems. [*techbrief: Java*]

Java applications run inside a Java Virtual Machine (JVM), which isolates them from the underlying hardware and operating system. Although Java programs can be written and run as what are called POJOs (Plain Old Java Objects), they may also run in the managed environments provided by Java application servers. The reasons for using a Java application server are exactly the same as those for using COMS rather than just running programs directly under MCP. They simplify development and improve run-time performance by providing a range of facilities that are commonly required across application types. Security and transaction management are examples. Java application servers are implementations of the Java EE (Java Platform, Enterprise Edition) specification. They are written in Java and therefore run inside a JVM. Oracle WebLogic Server, IBM WebSphere Application Server and the Red Hat JBoss Enterprise Application Platform (JBoss EAP) are examples of Java EE-compliant application servers.

Applications and other resources such as databases running outside the Java environment can be accessed through resource adapters, also called connectors. A standard – Java EE Connector Architecture (JCA) – is provided for implementing them. ClearPath applications running in COMS and Open DTP may be accessed through resource adapters provided by Unisys, as may the DMS II and the Relational Database Server. The resource adapters for COMS and Open DTP are bidirectional, allowing ClearPath systems to access external resources as well as provide access to ClearPath-resident resources.

The Java execution environment is supported in a specialty partition, the ClearPath MCP JProcessor, which is housed within the ClearPath system. It provides a high-performance execution environment. Unisys supports the Red Hat JBoss Enterprise Application Platform (EAP), a Java EE-certified Java application server. The Apache Tomcat web application server may also be run on ClearPath platforms. If the application servers are running outside the ClearPath system, they may be any Java EE-compliant implementation. The Unisys resource adapters to ClearPath applications are JCA-compliant and should therefore run with any JCA-compliant application server.

The complete Java offering for ClearPath includes more than a JVM, an application server, and the resource adapters. It also includes the Open Source Eclipse development environment. This environment has been extended through a number of plug-ins to tailor it for MCP. For example, code in COBOL and other languages can be imported into Eclipse, providing a consistent developer environment for Java and other application elements.

*Open distributed application servers: Open Group DTP*

Although many applications in ClearPath systems run in the native technology, a number of MCP users are using Open DTP for some or all of their applications. Open Distributed Transaction Processing (Open DTP) is the Unisys implementation of the Open Group DTP model. It was originally released on OS 2200 systems, then implemented on MCP systems and has continued to be enhanced. [*techbrief: ODTP*]

Because the Open Group DTP specification contains all the features necessary for DTP systems, service-oriented applications containing ClearPath MCP and other Open Group DTP implementations may be created. For example, ClearPath systems installed in two Unisys users communicate with each other using Open DTP, with one system requesting services from the other. Although both ends are in fact ClearPath, this is hidden by Open DTP.
Other application environments

Providing a relatively simple web access to COMS and Open DTP applications is a common requirement. Although this can be accomplished by other means, an on-platform web server with appropriate application interfaces is an effective and popular approach. The Web Transaction Server for ClearPath MCP (WebTS – formerly the NX/Atlas Web Server) provides a web server that supports static pages and various interfaces into MCP applications. [techbrief: Web Server]

For applications ported from other environments, there is a Common Gateway Interface (CGI). For newly written applications that require high performance, there is an ALGOL interface (AAPI). WEBPCM (PCM: Protocol Conversion Module) maps HTML into COMS messages. WebTS also supports Java Servlets.

Web Enabler for ClearPath MCP (formerly NX/Webstation for Java) is a Java Applet that provides a terminal emulation facility. This is a quick way to put an existing MCP application on the web. WebTS offers many other features, including comprehensive security support.

Integration middleware

Integration middleware is required in SOA environments containing a variety of technologies. Figure 3 shows the categories of integration middleware: loosely-coupled middleware and environment integrators.

Loosely-coupled middleware

In distributed systems using loosely-coupled middleware, the component applications communicate by exchanging messages of some form. No assumptions are made about how an application is written or the run-time technology used. All that is required is support of the appropriate middleware and agreement on the form and content of the message – its syntax and semantics. SOA environments comprising applications running in different technologies, Java EE, .NET, COMS and Open Group DTP, for example, may be constructed using this middleware.6

Figure 4 illustrates the architecture of loosely-coupled middleware, using message queuing (WebSphere MQ) to connect a ClearPath MCP system with IBM CICS as an example.

Loosely-coupled middleware includes the following.

- Web Services, as defined by the W3C and others, is specifically targeted at SOA. Its use of XML (eXtensible Markup Language) for data and protocol makes it ideally suited to heterogeneous environments. A variety of lower-level

---

6 Although loosely-coupled middleware is well suited to interconnecting applications running in different technologies, it can of course be used in homogeneous environments. For example, two Java EE systems could communicate using Web Services or message queuing. One reason for choosing a loosely-coupled approach is that it leaves open the option of changing one of the systems to a different technology. The other system is not affected as long as the replacement system implements the messaging technology, and retains the syntax and semantics of the messages.
protocols can be used to transport the SOAP messages; the SOAP standard does not specify any specific protocol. HTTP is the most common.

- Message queuing middleware, for which IBM's WebSphere MQ (formerly known as MQSeries) is the most widely used.
- File transfer using FTP (File Transfer Protocol) is also a loosely-coupled technology, although it may not always be considered to be middleware. It is well suited to shipping large quantities of data and has the advantage of being ubiquitously available. ESB and EAI\(^7\) (Enterprise Application Integration) products support FTP, among other protocols.

Web Services access to MCP-based applications and databases may be implemented in a number of ways, depending on the MCP application server and other technology in use. [techbrief: Web Services]

- The ClearPath MCP ePortal product provides access to COMS, Open DTP, Enterprise Application Environment (EAE) and Agile Business Suite (AB Suite) transactions as Web Services, both W3C and RESTful. It includes orchestration of transactions to deliver more complex services externally. The ePortal is implemented in a ClearPath Specialty Partition, which provides automated, secure, scalable and robust deployment. [techbrief: ePortal]
- Web Services access, including RESTful, to MCP systems, can be provided through Java EE application server environments, with resource adapters to connect to COMS and Open DTP applications, or directly to databases. JBoss Enterprise Application Platform (JBoss EAP), which may run on the ClearPath platform in the JProcessor, includes a Web Services implementation. If used off-platform, there are other choices of application server and Web Services implementation, for example using the Oracle WebLogic Application Server.
- Web Services access to COMS and Open DTP applications can also be provided through Microsoft IIS. Connections to the ClearPath resources are made through the Unisys DTI (Distributed Transaction Integration) product. (DTI is one of a number of environment integration products, which are explained below.)
- A third-party product from MGS, Inc – MGS Web Services for Unisys MCP (MGSWeb) – provides Web Services interfaces, implemented under MCP. COMS applications and DMS II databases can be service providers, while MCP applications can also be service consumers.

Two message queuing middleware technologies are supported WebSphere MQ and Microsoft Message Queuing. [techbrief: MQM] WebSphere MQ for ClearPath MCP interfaces to IBM WebSphere MQ for Windows from the MCP environment, with Message Queuing Interface (MQI) and COMS interfaces. The implementation provides full server capabilities and MQI functionality. WebSphere MQ for ClearPath MCP supports optional message and queue recovery within a local unit of work. Queue updates are not synchronized with database updates in a local unit of work\(^8\). If synchronization is required, WebSphere MQ for ClearPath MCP co-operates with Open DTP to support a global unit of work. Within the global unit of work, transactional integrity\(^9\) can be preserved across DMS II databases and message queues.

ClearPath MCP Interface to Microsoft Message Queuing (MSMQ) provides both a bridge between MSMQ and Unisys COMS applications, and a complete application programming interface (WinMQ API) for MCP-based applications. It enables applications running under MCP to send to and receive messages from these queues, which in turn enable the applications to interact asynchronously with other applications on Windows systems, other MCP, Unix and IBM systems.

---

\(^7\) The ideas of Enterprise Application Integration, and EAI products, arose before SOA and the development of ESB products. The goals of EAI and SOA are similar, so EAI and ESB products have overlapping capability. Oracle's WebLogic Integration is an example of an EAI Product.

\(^8\) A unit of work is a series of operations either committed in their entirety or not committed at all.

\(^9\) That is, all databases and queues are updated or none is.
FTP Service for ClearPath MCP is an implementation of FTP on ClearPath MCP platforms. It is frequently used for cross-platform integration. SFTP (Secure File Transfer Protocol) is supported through the Secure Shell (SSH) for ClearPath MCP product, which provides an implementation of the industry-standard Secure Shell (SSH) protocols.

Environment integrators

Environment integrators connect two different technologies by converting the standards on one side to those on the other. Figure 5 is a schematic of the architecture, using the example of COMTI (ClearPath MCP Interface to Microsoft Transaction Integrator), which links COMS with Microsoft COM+ and .Net. Each of the two environments acts as if it were the only environment present.

Unisys provides a number of off-the-shelf integrators, connecting different technologies to ClearPath applications and databases. In some cases, they are bidirectional in the sense that either end can initiate a sequence of interactions. Depending on the specific technologies, the environment integrators run under Windows, Unix, or Linux.

Table 1 shows the environment integrators available, the environments in which they run, and the technologies they interconnect. Environment 1 is MCP; environment 2 is the other environment – Windows in this case. Notes following the table add extra information where required.

<table>
<thead>
<tr>
<th>Integrator Product</th>
<th>Environment 1</th>
<th>Environment 2</th>
<th>Runs in</th>
</tr>
</thead>
<tbody>
<tr>
<td>Java Resource Adapters</td>
<td>COMS, Open DTP, batch, databases</td>
<td>Java</td>
<td>U/L/W or JP</td>
</tr>
<tr>
<td>DTI</td>
<td>Open DTP</td>
<td>Various</td>
<td>Windows</td>
</tr>
<tr>
<td>Component Enabler</td>
<td>EAE, AB Suite</td>
<td>COM+, .NET, Java</td>
<td>Windows</td>
</tr>
<tr>
<td>COMTI</td>
<td>COMS</td>
<td>COM+, .Net</td>
<td>Windows</td>
</tr>
<tr>
<td>Oracle Tuxedo Mainframe Adapter OSI-TP</td>
<td>Open DTP</td>
<td>Tuxedo</td>
<td>U/L/W</td>
</tr>
<tr>
<td>ClearPath ePortal</td>
<td>COMS, EAE, AB Suite</td>
<td>Various</td>
<td>SP</td>
</tr>
</tbody>
</table>

Table 1: Environment integrators

Notes to Table 1:

1) The last column shows the operating system under which the integrator runs: U = Unix, L = Linux, W = Windows, JP = JProcessor. SP = Specialty Partition.

2) Java Resource Adapters are pure Java, and can therefore run in a Java application server under any operating system. Where they run in any particular case depends on how the ClearPath environment is set up (how much of the
Java environment is running within the ClearPath system. ClearPath databases supported are DMS II and Relational Database Server. [techbrief: Java]

3) DTI (Distributed Transaction Integration) and COMTI variously support a variety of external environments, including COM+/.NET, the Web, Java and a number of packages. [techbrief: TXINT]

4) For Component Enabler, see [techbrief: ADE]

5) Oracle Tuxedo Mainframe Adapter OSI-TP (formerly eLink OSI-TP) connects Tuxedo to Open Group DTP implementations. It can also be used to connect to Oracle WebLogic Server, as Tuxedo and WebLogic are closely integrated. [techbrief: ODTP]

6) Environment integrators can also be used to implement Web Services, by using the Web Services software (SOAP engine and so on) available in Java or .NET environments and connecting to the ClearPath environment using one of the integrators. [techbrief: Web Services]

7) The ePortal allows all the major smart phones (Android, Blackberry, iPhone and Windows), tablets (Android, Blackberry, iPad and Windows), and Web Services. [techbrief: ePortal]

Other middleware

Other middleware includes the following:

- Direct data access products
- BIS
- Sockets
- EAE and AB Suite
- Third-party products

Direct Data Access Products

Remote systems can directly access databases and other data stores in MCP systems and MCP systems can access other databases. This access may be as part of a distributed transaction or for data analysis purposes. The products available and the databases concerned are as follows. [techbrief: Data Access]

Access to MCP databases: DMS II and Other Data Stores

- OBDC Access (formerly called Data Access) supports access through Open DataBase Connectivity (ODBC) into DMS II and other MCP files. The mapping from DMS II to relational structures and the parsing of relational queries are done in a Windows system.
- An OLE DB Data Provider for DMS II, called Enterprise Database OLE DB Data Provider for ClearPath MCP, is available. This provider offers better performance than access through ODBC because it does not require mapping to a relational structure.
- The XML Provider enables DMS II and other MCP data source to interoperate with other XML-enabled applications and data sources locally or anywhere on the Web. The XML Provider can render data in a browser, export data in XML format and import data contained in an XML document to a database.

Note that Java resource adapters also provide JDBC access to DMS II and Relational Database Server. For details, see [techbrief: Java].
Access to external databases from MCP systems

Application Data Access for ClearPath MCP systems provides access to any database on Windows platforms that provide an ODBC driver. This capability includes all major databases on all Windows platforms—ranging from simple PC databases to enterprise databases such as Oracle.

Client Access Services

Client Access Services enables a workstation to interoperate with the MCP environment in the same way that it interoperates with a Windows Server. It integrates the MCP environment into a Windows-based Microsoft network.

BIS

BIS also provides extensive facilities for accessing different databases and other data sources. Although BIS does not run under ClearPath MCP (it runs under ClearPath OS 2200, Windows and Linux on Intel, plus some variants of Unix), it can access MCP databases.

Sockets

Low-level middleware, in particular sockets, provides a means of interconnecting applications in a distributed environment. Some would argue that it is not really middleware because it leaves too much for application programmers to implement. However, because of its simplicity, sockets is a widely-used technology. In addition, it provides a foundation for much of the more sophisticated middleware.

Historically in ClearPath MCP systems, sockets were implemented by Port Files. By using the CCF (Custom Connection Facility) capability, socket connections can be mapped to a COMS remote station, thereby allowing the client to send messages directly to an application just as if it were a terminal. Port Files and CCF support both Unix-like sockets and Microsoft Named Pipes. For Named Pipes, the usercode and password of the workstation are used for an MCP logon. However, Port Files do not support the use of UDP (User Datagram Protocol) or secure access to Secure Sockets Layer (SSL) functions. The MCP Sockets Services API provides access to these functions. [techbrief: Sockets]

The next most basic level of middleware is the Remote Procedure Call (RPC). As its name suggests, it provides a client program with the facility to execute procedures on a server program. ClearPath MCP systems support two kinds of RPC: ONC+ (Open Network Computing) RPC and the Microsoft WIN RPC. Both RPC standards were designed with the C language in mind.

EAE and AB Suite

EAE and AB Suite, which is an evolution of EAE, are model-driven application development environments, which can be generated for and deployed under MCP as well as various other operating systems. AB Suite will eventually replace EAE. Although EAE and AB Suite are not middleware, they do offer some middleware features or features enabling the applications to participate in distributed environments. [techbrief: ADE]

Third Party Products

There are various third party products on the market for use with ClearPath MCP systems. As far as middleware is concerned, there is a WebSphere MQ product, which runs entirely under MCP. It supports database and queue update synchronisation. MQSWeb (already mentioned above) is another product.
Middleware and ClearPath systems with fabric infrastructure

In 2014, Unisys began delivery of ClearPath Libra systems, implemented in a fabric-based architecture. As well as providing MCP environments, together with Specialty Partitions, the systems support Linux and Windows applications within the fabric. Complete service architectures can be implemented within the fabric, providing very high levels of performance and security. For more detailed information, see the documentation available for ClearPath fabric-based systems.

Figure 6 shows the principal components of an MCP system. The systems are packaged in two 42U cabinets. The left-hand cabinet contains Unisys Intel platforms implementing the ClearPath MCP components forming the ClearPath OS Complex. The operating system and applications are executed in a platform – the Processor/Memory Module (PMM) – containing the Unisys firmware, which implements the required Libra architecture. System input/output is implemented in another platform: the I/O Specialty Partition Module (ISM), which, as its name suggests, can also contain Specialty Partitions. The I/O logic and Specialty Partitions run in Secure Partitions under s-Par®. The number of PMMs and ISMs varies by system performance range and available options. The various systems range from entry level through mid-range to high-end performance.

Figure 6: Schematic of a ClearPath fabric-based system

---

The right-hand cabinet contains up to 12 Enterprise Partitionable Platforms, which are Unisys Intel platforms. Each EPP can support up to 12 secure partitions using Unisys s-Par®, allowing a maximum of 144 partitions. The partitions can be of variable sizes and host either Windows or Linux applications.

A component called the Interconnect establishes the fabric infrastructure, which connects all the components. It provides the means of interworking between the partitions, supporting applications distributed across different operating environments. The Interconnect comprises two parts: software and a high-speed, low latency wire connection. The operating environments have no knowledge of the connection technology. The approach allows the introduction of new wire technologies as and when they become available without any effect on the operating environment using it.

The systems are delivered as an integrated stack containing the following.

- The hardware, including the firmware, as shown in figure 6, together with the fabric infrastructure.
- All of the software for MCP.
- Specialty Partitions.
- Windows and Linux operating systems with recommended configurations. Clients are also able to use their own versions, with their own support arrangements. The ClearPath fabric-based system also provides a secure environment for the Windows and Linux applications. They run in secure partitions provided by s-Par® and have the options of using Unisys Stealth™ technology for additional security.

The fabric infrastructure provides a number of options for using middleware in distributed systems deployments. For example, a Java application could be housed in partitions within EPPs, and use middleware to communicate with MCP applications if required. The files containing the JVM and supporting software would be within the file system used by the partition. In effect, it would be much the same as an external deployment although there would be the extra security available with the ClearPath fabric-based system. Alternatively, if close integration with the MCP applications or databases is required, the Java application can run in the JProcessor. In that case, the JVM and supporting software are supported as part of the MCP or MCP environment.
Using the middleware

There are many examples of ClearPath applications using middleware, including participating in SOA implementations. Various techniques have been used, depending on the user requirements, the design of the existing applications, the technology used in its implementation, and technology preferences and policies. To assist in the design, Unisys provides a series of reference architectures, which identify the appropriate technology to use, depending on the initial environment and requirements. The products are complemented by professional services. This section describes some examples of middleware in action with ClearPath MCP systems.

Case study 1

This example is a large government agency providing a wide variety of services. The organisation uses ClearPath Libra systems for a number of applications. It is interesting because of the strategic direction the agency follows and the ways it uses a variety of the products discussed in this paper to achieve its goals.

Three guiding principles have been driving the strategy over a number of years. The first is to provide access to applications using Web technology, either as an initial implementation, or to replace or complement green screen access. As well as allowing the possibility of external access, internal users benefit from a more intuitive interface, improving user satisfaction and reducing training costs.

The second principle is to provide the technology for integration between different applications, both within the organisation and externally, using standard technologies such as Web Services.

The third is to reduce the number of different platforms and technologies in use by consolidating applications within the ClearPath Libra systems. One goal is to eliminate the costs incurred as a result of using different hardware and software technologies for each application. Another has been to reduce or eliminate data duplication or the need for users to access two systems. From an operational viewpoint, consolidation within the ClearPath environment moves the applications into a single management framework. This simplifies application upgrades and operations, and reduces the risk of operational error caused by complexity. And, of special importance, the applications now reside within a highly-secure environment.

This section looks at how the middleware is being used with some of the applications, in line with the above principles.

Figure 7 is a schematic of the current Libra environment. Starting at the top, a number of the applications currently running under MCP were running in separate platforms, using different software such as Visual Basic or PowerBuilder and, in some cases, were COTS packaged applications. They have been re-implemented in COBOL, using WebTS and WEBPCM to provide access for users, and DMS II as the database. [techbrief: WebTS]

Another MCP-resident COBOL application has been Web enabled using WebTS, reducing training costs. The same application also needed to integrate with an external application, with Web Services technology the preferred method. MGSWeb, a product from MGS Inc, is used to provide the Web Services connection to the MCP-resident application. MGSWeb runs within the MCP environment so no platform outside of MCP is used in the connection. Within the case study 1 environment, MGSWeb is used where the MCP end functions as a service provider and a consumer. Figure 8 is a high-level view of the architecture. (Note that MGS can also provide a Web server.)

A final example of an application running under MCP used green screens, with associated high training costs. The application was web enabled using ePortal. Training costs were reduced, user satisfaction increased and the application opened to wider access. ePortal is also used for Web Services access for a mobile application. [techbrief: ePortal]
Turning now to Java implementations, some applications previously running in other systems and using a variety of technologies have been re-implemented in Java and run within the ClearPath system within JBoss EAP. In some cases,
they run in an integrated Unisys ES7000; in others in the JProcessor. Integration with MCP resources uses a Web Services connection through MGSWeb for COMS and a JDBC driver to DMSQL for DMS II. [techbrief: Java]

Overall, considerable annual cost savings have been achieved, far exceeding the cost of the implementations, which has been modest, and user satisfaction has increased.

What about the future? The environment shown in figure 7 could be moved into a fabric-based ClearPath system, as shown in figure 9.

**Figure 9:** The environment as it could be deployed in a fabric-based ClearPath system

ePortal runs in a specialty partition, on the left hand side of the system shown in the picture. There are two options for Java: the JProcessor or secure partitions in the EPP. The JProcessor is ideal for Java applications which are closely integrated with MCP applications, as the configuration and other files required are within the MCP environment. Java applications not requiring close integration may run in one of the secure partitions in the EPP, on the right hand side of the figure. They may still communicate with the MCP system but their configuration files are not stored on the MCP environment.

Applications apart from Java may run in secure partitions in the EPP, under Windows or Linux as required. They can take advantage of the high performance fabric infrastructure to interwork with MCP and other applications, and benefit from the high levels of security.
Case study 2

New business processes can be implemented quickly using one or more of the integration product types available on the market, for example ESB (Enterprise Service Bus) and EAI (Enterprise Application Integration) products. ClearPath systems can be connected in a variety of ways to these products, where the ClearPath applications function as service providers and/or consumers. The Microsoft .Net Framework is such a product, providing orchestration and other capabilities to enable rapid definition of new business processes.

A bank provides an example of the use of this technology. Figure 10 shows the architecture and components used in an initial proof-of-concept implementation, using a subset of the transactions available from one of the BankApp applications, which run under MCP/COMS. The Customer Relationship Management System provides data about clients from an Oracle database.

Figure 10: Architecture for bank using .Net Framework

The .Net Framework components in the Windows environment are described below.

- Model View Controller (MVC): The Model View Controller is a well-known design pattern offered by Microsoft as an extension to ASP .Net 3.5. It is aimed at rapid development of dynamic data-driven web pages.
- Windows Communications Foundation (WCF): The Windows Communication Foundation is a set of technologies for building connected systems. It allows the separation of the selection and specification of the communications layer
(protocols, destinations etc) of the application from the code into the configuration of the application. It is included as part of the Microsoft .Net Framework 3.0.

- **Windows Workflow Foundation (WF):** The Windows Workflow Foundation is the programming model, engine and tools for defining workflow-enabled applications on Microsoft Windows. It is included as part of the Microsoft .Net Framework 3.0.

The BankApp applications are accessed through the ePortal using Web Services as the interface mechanism. The implementation time used was short, with relatively low effort expended. The ePortal’s ease of use allowed client staff with minimal experience of the technology to become productive very quickly. The resulting architecture provides a basis for expansion, allowing new external services to be defined and new service provider applications to be added.

### Case study 3

This example shows how ClearPath ePortal can be used to modernise applications and implement a service-oriented architecture. The client supplies ICT services to a large city. It is responsible for setting Information and Communication Technology (ICT) policies, consolidating and planning the use of IT, and managing the data centre and network data communications.

The organisation had a number of applications implemented in a ClearPath MCP environment, using green screen interfaces. Two important applications in particular needed modernisation: one provided financial services (an EAE application), the other administration functions (a COBOL application).

The organisation had three primary requirements for the modernisation.

- One was to improve the user interface, making the applications easier to use and so reduce training costs. Additional benefits would be to increase end user satisfaction and, by using a Web GUI, have the option of extending services directly to the public for self-service through the Internet.
- The second was to increase the efficiency of business processes depending on IT services, so reducing cost and increasing client satisfaction.
- The final requirement was to accelerate the development of new application services from conception to delivery, minimising the need to change existing applications.

Finance exhibited a problem that was (and still is) widespread across different organisations: completing a single business process may require several, separate interactions. In the case of paying taxes using the finance application, for instance, citizens would be required to visit up to three different counters. The goal was to deliver the service from one position only, with all of the component parts co-ordinated into one process.

ePortal was chosen for the implementation of a service architecture. A group of green screen transactions (Ispecs) was consolidated into a single, Web-enabled service. ePortal provides the GUI and the orchestration of the Ispecs. A new function was developed to create multiple records from a single keystroke.

Figure 11 shows the architecture, which was also used for the administration application as shown in the figure. As well as providing the GUI, ePortal offers external services to users – the consumers of services – as shown in the upper part of the figure. External services are then delivered by orchestrating a number of internal services, which are Ispecs in the finance application. In addition, new C# code was written to use Web Services to access other systems, the responses from which are then combined with the data from the Ispecs and delivered to the end user.

The implementation for the administration application uses the same architecture, as shown in the figure. In this case, the internal services are COBOL transactions rather than EAE Ispecs. The efficiency gains have proved significant, as the application is widely used across city government departments.
By using the service architecture approach shown in the figure, the organisation is able to develop and deploy new external services quickly by orchestrating existing and new internal services in different combinations.
More information

Information about technologies and products is widely available, at both strategic and detailed levels. The following are just a few of the useful sources, divided into two groups: vendor-neutral organisations, and companies and other organisations providing products and services.

Vendor-neutral organisations

There are many vendor-neutral organisations working on standards and other collaborative activities, usually under the domain .ORG. Here is a selected list of organisations, a number of which are mentioned in the text of this paper:

The Apache Software Foundation: http://www.apache.org/
IEEE: http://www.ieee.org
The Internet Engineering Task Force (IETF): http://www.ietf.org
The International Organization for Standardization (ISO): http://www.iso.org
The International Telecommunications Union (ITU – formerly CCITT): http://www.itu.int
The National Institute of Standards and Technology (NIST): http://nvd.nist.gov
The Open Group: http://www.opengroup.org
UDDI.xml.org (for information about Web Services): http://uddi.xml.org
World Wide Web Consortium (W3C): http://www.w3.org

* * * *

Organisations providing products and services

Unisys ClearPath systems: http://www.unisys.com/offering/high-end-servers/clearpath-systems

Unisys

Unisys White Papers provide a variety of additional information on all aspects of ClearPath systems:

http://www.unisys.com/search/unisys?k=UnisysAssetType:Whitepaper#Default=[%22k%22%3A%22UnisysAssetType%3AWhitepaper%22]#1319586C-A6BF-4B17-BFFD-516084782910=[%22k%22%3A%22UnisysAssetType%3AWhitepaper%22]

For the OS 2200 version of this paper: http://www.unisys.com/offering/high-end-servers/clearpath-systems/clearpath-os-2200-software/Whitepaper/ClearPath-Middleware-Strategy-and-Products-for-OS-2200-Systems-id-1745

Unisys product technical documentation:

IBM

IBM’s WebSphere: http://www.ibm.com/web/portal/software/websphere

MGS, INC

Microsoft

Microsoft’s component technology can be found at: http://www.microsoft.com/com/default.mspx  Information about the .NET Framework can be found at: http://www.microsoft.com/net/

Oracle

Information about Java: http://www.oracle.com/technetwork/java/index.html

For Oracle products, including middleware: http://www.oracle.com/us/products/index.html


Red Hat

JBoss middleware (JBoss is a division of Red Hat): http://www.redhat.com/en/technologies/jboss-middleware
Technical briefs

The Technical Briefs (techbriefs) are intended to provide some additional information and clarification; they are not intended to be the last word on the subjects they address. Where appropriate, cross-references are provided among the technical briefs.

A note on the terminology: middleware is generally concerned with communication between software entities – chunks of code in other words. The words ‘program’ or ‘application program’ are used in the technical briefs to denote such an entity in general. More specific entities are identified as necessary, for example ‘EJB’, and ‘Open DTP service’. The word ‘application’ on its own is generally avoided in this context for two reasons. An application may comprise many software entities, possibly distributed over several servers. And secondly, different parts of the same application may use middleware for intercommunication; one part may queue a message to another for later processing, for instance.

The technical briefs are:

**Java**: Explains the basic principles of Java, including JVMs and applications servers, and how they are implemented with ClearPath MCP systems.

**ODTP**: Explains the Open Group DTP model and its implementation in ClearPath MCP systems.

**WebTS**: Describes the Web Transaction Server product.

**Web Services**: Explains Web Services technology and how ClearPath MCP application programs can be accessed through Web Services.

**ePortal**: Describes the ClearPath ePortal Specialty Partition.

**MQM**: Explains the principles of message queuing middleware and describes the implementations available with ClearPath MCP systems, in particular WebSphere MQ (MQSeries).

**File Transfer Protocol (FTP)**: Describes the implementation of FTP under MCP.

**TXINT**: Describes two environment integration products: DTI and COMTI.

**Data access**: Discusses the ideas of direct data access and some of the products available with ClearPath MCP systems.

**Sockets**: Explains the principles of sockets and the various implementations in ClearPath MCP systems.

**Advanced Development Environments (ADE)**: Describes the middleware-related features available with EAE and AB Suite.
Java

The basic Java platform from Oracle is the Java Platform, Standard Edition (Java SE). There are two products under the platform: the Java SE Development Kit, abbreviated to JDK; and the Java SE Runtime Environment, abbreviated to JRE. The Java SE platform does not provide all the facilities required for large-scale applications such as core banking or airline reservation systems. The Java Platform, Enterprise Edition (Java EE) builds on Java SE to close the gap. Associated products are the Java EE SDK and Java Application Platform SDK. Java application servers are implementations of Java EE, providing deployment environments analogous to COMS or Open DTP. They are written in Java and therefore run inside a JVM. Figure Java-1 is a simplified view of the Java EE architecture.

Java EE application servers have two containers for application logic. The web container implements the logic as servlets or Java Server Pages (JSPs), which may be sufficient for some applications; the dashed box in the figure shows this. Applications requiring facilities such as life cycle management, security and transaction management can take advantage of the facilities provided with Enterprise Java Beans (EJBs), which run in an EJB container. Depending on the client type, access to the application logic is through the web container, which may then invoke EJBs or the EJB container directly. For example, a browser using HTML would go through the web container while a request from another Java application would go directly into the EJB container.

Applications and other resources such as databases running outside the Java environment can be accessed through connectors, also called resource adapters. A standard – Java EE Connector Architecture (JCA) – is provided for

Figure Java-1: Simplified view of Java EE architecture
implementing them. As shown in figure Java-1, resource adapters may be accessed from applications running in a Web container or an EJB container.

Figure Java-2 shows a high-level view of the Java environment and how ClearPath applications fit into it (ClearPath systems are represented by the grey shaded rectangles).

- The entire Java environment may run outside the ClearPath system, with resource adapters connecting to ClearPath applications or databases across a network, as shown on the left of the figure.
- A web application server may run outside the ClearPath platform, using a network connection to an application server running in a JProcessor in the ClearPath system. This configuration is shown at the centre of the figure.
- Finally, as shown on the right, the entire application server environment may run inside a JProcessor within the ClearPath system.

![Figure Java-2: Configurations of application servers in ClearPath systems](image)

The ability to configure the Java EE environment in different ways across platforms allows ClearPath systems to run non-Java applications, with connections through resource adapters to a distributed Java EE environment. ClearPath systems can also participate fully in a Java EE environment, running Java applications on ClearPath platforms. It is relatively easy to relocate application code from one platform to another. For example, a Java application running off-platform can be moved into a ClearPath system.

Unisys provides a number of resource adapters to allow Java applications to access MCP resources. Table Java-1 shows the resource adapters and the environments in which they can run. Some clarification is necessary for the JDBC driver.
The resource adapters for Open DTP and COMS are bidirectional, allowing ClearPath systems to access external resources as well as provide access to ClearPath-resident resources.

<table>
<thead>
<tr>
<th>Resource adapter</th>
<th>Managed in App. Server</th>
<th>Unmanaged</th>
</tr>
</thead>
<tbody>
<tr>
<td>JDBC for ClearPath MCP driver</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Relational Database Server for ClearPath MCP JDBC driver</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>MCP Transaction Resource Adapter for the Java Platform</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Open Distributed Transaction Processing Resource Adapter for the Java Platform</td>
<td>Yes</td>
<td>No</td>
</tr>
</tbody>
</table>

Table Java-1: Java resource adapters and the environments in which they can run

**ODTP**

This brief provides an overview of the Open Group DTP\textsuperscript{11} model and its implementation. Figure ODTP-1 shows the model’s architectural elements, as they would appear in two systems communicating with each other over a network. Configurations can extend to a network of many more systems.

---

\textsuperscript{11} The model was initially defined by X/Open, which subsequently merged with the Open Software Foundation (OSF) to form the Open Group.
The elements of the model are as follows:

- The Application Programs, which provide Open Group DTP services, co-operate with each other to perform the work required, that is, transactions. They may be in the same system or distributed among two or more. Co-operating systems do not need to use the same operating system.
- A Resource Manager (RM) is typically a database manager but may include other, less obvious ‘database’ resources, such as message queue managers.
- The Transaction Manager (TM) is used to delineate transactions and to co-ordinate the process of two-phase commit. It co-operates with another element, the Communications Resource Manager (CRM), which provides the means for application programs to communicate with each other. The CRM also provides the means for TMs in different systems to communicate, typically in order to co-ordinate a two-phase commit process.
- Two-phase commit (2PC) preserves database consistency. Because a single transaction may be spread over more than one system, each of which may wish to update a database, it is essential that, on the completion of the whole transaction (sometimes called a global transaction), the databases and other resources such as message queues are left in a consistent state. Either all the databases are updated or none is. Maintaining this consistency is sometimes called preserving the ACID\(^{12}\) properties of the transaction.

Figure ODTP-2 shows the Open Group DTP elements in a single system, with the interfaces indicated on the diagram.

---

\(^{12}\) The ACID properties are as follows. *Atomicity*: A transaction should be completed and committed, or rolled back if there are problems during its execution. It is an atomic entity in that it is indivisible. *Consistent*: Changes made to databases must be from one valid state to another, not something indeterminate. *Isolated*: The results of a transaction should be invisible to other transactions until the transaction is completed. *Durable*: Changes to databases, which have been made during a transaction, should be permanent and survive future media and other failures. Maintaining the ACID properties is what is meant by transactional integrity.
The elements of the Open Group DTP are as follows:

- For the RM, the interface (API) depends on the specific resource manager. SQL is an example for a database RM; MQI is an example for a message queuing RM.
- The TM supports a set of API functions, known as the TX set. They include primitives such as begin transaction, end transaction and commit updates. The total number is small, around 10 in all.
- The CRM provides the additional API functions needed to support communication between application programs. This set is called XATMI and comprises only about 15 primitives. They include functions to get and release buffers, and for client programs to call other programs – services, in the terminology of the model. A service may also function as a client to another service. Three types of communication between programs are supported: a simple call/return structure; an asynchronous version, which allows several services to be called without waiting for a reply; and a conversational mode of operation.
- In addition, there are two sets of system level interfaces invisible to the application programs. The first, XA, is the standard that provides the 2PC capability. A database or other resource manager that supports it is called XA-compliant. The Unisys MCP databases – DMS II and Relational Database Server and, for WebSphere MQ, message queues – are all XA-compliant, as are all the other major databases such as Oracle. The other system level interface, XA+, is used between the TM and the CRM to manage the thread of control. It is an expanded version of XA.

How does a global transaction requiring 2PC flow around an Open Group DTP environment? A client is connected to one system only, as its entry point to the Open Group DTP network. A client application program in the system is responsible for the user interface, as well as starting the transaction. It uses the TX API to start a global transaction. Thereafter, XATMI is used to call various services, which in turn may call other services. These programs may be in any system and may use local RMs to update databases. When the work is done, TX is used by the initiating system to commit the transaction. The initiating TM (the one that received the start transaction) then communicates with all the other TMs to request approval to commit and update the databases. If all agree, the databases are all updated. If any do not agree, all databases are returned to the state they were in before the transaction began. XA-compliant RMs must have the facilities to manage a failure at any point after they have agreed to commit.

When two Open Group DTP systems communicate with each other, they do so using the CRM, as explained above. OSI-TP provides services and protocols, which assist the 2PC process and communication between systems\(^\text{13}\). They in turn operate across lower layers, typically TCP/IP. The interfaces are shown in the diagram, but note that they are not seen by the application programs.

The model may be implemented in two forms:

- A full implementation, containing the TM, CRM, and so on
- A client-only version, which depends on connecting to a TM to gain access to services

The Unisys implementation under MCP, Open Distributed Transaction Processing (Open DTP), is the full model. Figure ODTP-3 is a schematic of the implementation. The TM is implemented as a system library. COMS implements the CRM functions. OSI-TP is used to communicate with other instances of the model.

Open DTP applications can either be written using the XATMI or using the standard COMS direct-window interface. In the former case, there are ALGOL, C, and COBOL versions of the XATMI calls. In the latter case, a single, COMS direct-window application can be used by both existing terminals and by Open DTP. The program can tell the difference through

\(^{13}\) OSI-TP (ISO/IEC 10026) is mandated by the Open Group for interoperability between heterogeneous DTP systems.
the COMS station name. All Open DTP traffic is routed as if from a single station. Note that Open DTP services are stateless in the sense that they do not keep session information.

Figure ODTP-3: The Open Group DTP model as implemented under MCP

MCP application programs using Open DTP are able to interwork with a wide variety of other technologies, including Microsoft .NET, Tuxedo, Java and more, as shown in figure ODTP-3. Global transactions, with full transactional integrity using two-phase commit, are possible with Tuxedo, Microsoft .NET and Java EE application servers. The JCA Resource Adapter for Open DTP and the execution environment integrators DTI and Oracle Tuxedo Mainframe Adapter OSI-TP provide the connections. See also [techbrief: Java] and [techbrief: TXINT].

Web Transaction Server for MCP (WebTS)


The WebTS is a fully-featured web server. A single MCP machine can run multiple websites, each distinguished by different TCP/IP port numbers. Each site has one or more virtual directories, which map to:

- Physical directories – for static web pages
- URLs – to redirect the web traffic to a different web server
- Applications

Security is applied at the website level and the virtual directory level. By default, a website will run without privileged status. The standard installation creates one privileged server for administration and one unprivileged server for normal work. For both the website level and on individual virtual directories, explicit IP addresses or ranges of addresses can be
defined (using subnet masks) which either deny or grant access. On the virtual directory, users can be given read access and denied write access. HTTP usercode is checked using standard MCP facilities.

WebTS also supports SSL. The encryption and decryption are very processor-intensive tasks. On ClearPath MCP systems, the processing is exported to a Windows environment.

All the configuration information is specified in a configuration file and can be changed online. A site manager utility, implemented in a Java Applet which runs in a browser, is provided so the administrator does not have to remember the command syntax. There are facilities for logging and analysing web traffic.

One method of accessing MCP applications over the web is through Java Applets running in a web browser. The Applet code is stored in a static file downloaded to the browser. The Applet can open a TCP/IP session with the web server host that provided the page. This is the technique used by the Web Enabler. The Applet provides T27 emulation and the TCP/IP session connects back to COMS. [techbrief: Sockets] The Site Manager utility uses a similar technique.

The alternative to running code in the browser is to call programs in the server. Figure WebTS-1 shows the programmatic interfaces to the WebTS environment.

![Figure WebTS-1: WebTS environment, showing the various connections possible](image)

The CGI interface, which is implemented as an ALGOL library, was originally designed to be similar to the CGI interface used in other web servers. The AAPI was subsequently developed in response to the need for a more robust interface with higher performance. The AAPI is also an ALGOL interface except it uses connection libraries rather than server libraries. Connection libraries are like back-to-back libraries; both the WebTS task and the AAPI application can send messages to each other simultaneously (subject to internal locking to maintain integrity). A single application can handle many requests simultaneously, leading to better performance, and because most processing runs on the application stack, the WebTS does not stop when application errors occur.

The WEBPCM and Java Servlet interface use the AAPI in their underlying implementation. The AAPI interface was modelled on the Java Servlet interface; therefore, developing a Servlet facility was largely a matter of building an AAPI interface into the JVM. WEBPCM provides an interface from the WebTS to COMS. It is part of the COMS CCF structure. [techbrief: Sockets] Both COMS direct-window and remote file programs can be used.
COMS programs that handle web messages differ from normal COMS programs in that they read and generate HTML. COMS direct-window applications can be developed to handle, for example, T27 and HTML output by using COMS processing items. To make reading and writing HTML data easier, especially for COBOL which is not designed for string manipulation, Unisys has provided a library called WEBAPPSUPPORT. Figure WebTS-2 is a schematic of the WEBAPPSUPPORT environment.

**Figure WebTS-2: WEPCM and WEBAPPSUPPORT**

Input for a web program typically originates from an HTML forms page. The input data is of the form ‘field1 = value1&field2 = value2 …’ where field1 and field2 are the names given in the HTML form to the text boxes or selection boxes displayed in the browser. WEBPCM creates a message object that contains all the information input to the form. This information is placed in the input message. The object gets passed to the application via CCF and COMS. The application should then pass it on to WEBAPPSUPPORT in order to unravel the data field values and the header field values. WEBAPPSUPPORT also has a role to play on output. The program can add content in increments rather than all at once, and the program can set the header values.

WEBPCM has features for supporting sessions. HTTP is essentially stateless; downloading each page uses a separate TCP/IP session. Simulating a session dialog can be done by using cookies or by appending a session id to the output data in such a way that it is returned in the following input. WEBPCM has features to support both these forms of session handling. The web sessions are mapped to COMS stations so that the application detects sessions as with normal COMS sessions.

WEBAPPSUPPORT has features for merging the programmatic output data with a static web page file. The static page can be built with any of the well-known tools.
Web Services

The section ‘Distributed systems and service-oriented architectures’ at the front of this paper introduced the basic concepts of service orientation and Web services. Web Services are best suited for interoperation between different software applications with the following characteristics.

- They operate over the Internet where reliability and speed cannot be guaranteed.
- There is no ability to manage deployment so that all requesters and providers are upgraded at once, that is, the different systems may become incompatible.
- The various systems run on different platforms and vendor products.
- An existing application needs to be exposed for use over a network, and can be wrapped as a Web Service.

Web Services could be used for local interactions, that is, where the systems are not subject to the above characteristics. However, other interaction technologies such as remote procedure calls or method invocations (DCOM for example) may be preferred for performance reasons.

In the document ‘Web Services Architecture’ (WSA), the W3C defines a Web Service as follows14.

‘A Web Service is a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-processable format (specifically WSDL15). Other systems interact with the Web service in a manner prescribed by its description using SOAP messages, typically conveyed using HTTP with an XML serialization in conjunction with other Web-related standards.’

WSA goes on to define two major classes of Web Service, as shown in figure Web Services-1.

As shown in the figure, the two classes are:

- Arbitrary Web Services
- REST-compliant Web Services

With arbitrary Web Services, services can expose an arbitrary set of operations. They use the SOAP protocol, which can be carried by a variety of transports. HTTP is the most commonly used but others are possible; the figure shows some examples. Resource state is represented, described and communicated via representations of the resource in a variety of widely-understood data formats such as XML, HTML and JPEG.

---

14 WSA can be found at http://www.w3.org/TR/2004/NOTE-ws-arch-20040211/. The definition quoted is in section 1.4.
15 Not all Web Services have a WSDL description. Simple RESTful services may do away with it altogether.
REST-compliant Web Services are more constrained and simpler\(^\text{16}\). The primary purpose is to manipulate XML representations of Web resources using a uniform set of stateless operations. The transport is HTTP. SOAP can be used consistently with REST but could also be inconsistent.

There is a great deal of literature about Web Services technology. The World Wide Web Consortium (W3C) provides both tutorial and normative material; see the W3C website for details. (The URL is the 'More information' section of this paper.)

**Unisys implementations: general Architecture**

Unisys implementations of Web Services use either Java or .NET. Web Services is part of the Java SE and Java EE specifications. There is also a .NET implementation. For MCP systems, COMS and Open DTP applications may receive incoming Web Services requests and initiate outgoing requests, using either Java or .NET as the Web Services implementation. Enterprise and Relational Database Servers databases may become service providers. This techbrief describes the options, starting with the general architecture followed by the Java and .NET implementations. (The ClearPath ePortal also provides a Web Services interface. It is described in [techbrief: ePortal].)

There is also a third party Web Services product supplied by MGS, Inc: MGS Web Services for Unisys MCP Systems (MGSWeb), which runs under MCP. It will not be described here; the 'More information' section, above, provides a link to further information on the MGS website.

Figure Web Services-2 shows the general architecture. The MCP environment is at the bottom of the figure, containing the application environments and databases.

![Figure Web Services-2: General architecture for Java and .Net Web Services implementation](http://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm)

\(^{16}\) REST – REpresentational State Transfer – was introduced by R T Fielding in a PhD dissertation, which can be found at [http://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm](http://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm)
The co-operating system at the top of the figure could be a Web Services consumer, which requests services provided by the MCP system, or a Web Services provider if the MCP applications are consumers. The protocol used is SOAP, which is normally carried over HTTP, although the standard does not specify any protocol, only that there has to be one.

The box labelled ‘Web Services Enabling Software’ in between bridges the gap. It contains three elements, which enable the MCP applications to be either Web Service providers or consumers, and the databases to be providers. As shown, the execution environments are either .NET or Java EE.

Starting with the highest shown in the figure, the three enabling software elements are:

- The **Web Services engine** supports the Web Services protocols, including SOAP, enabling the system to interwork with other systems.

- The next element is the **integration component**, which consumes or requests a service. It could in principle provide the entire service (or implement the consumer), but in the context of figure Web Services-2, it bridges between the Web Service engine and the integration middleware.

- The **Integration middleware** connects to the various applications and databases running under MCP, which are the ultimate providers or consumers of services. The integration middleware may be loosely coupled, for example message queuing, or one of the environment integrator products.

There are various possible options for implementing the enabling software elements. The following sections expand on the Java and .NET implementations, showing some representative architecture.

**Java implementations**

Java implementations support COMS and Open DTP application programs as both providers and consumers of Web Services, as well as exposing DMS II and Relational Database Servers as providers. Because Java EE application servers can run outside the MCP environment or within it, a number of options for implementation are available.

Figure Web Services-3 shows two possible configurations.

- On the left of the figure, the entire Java EE environment is outside of the MCP environment, for example running under Unix or Linux. Any Java EE-compliant application server can be used, for example Oracle WebLogic Server or IBM WebSphere Application Server.

- On the right of the figure, the entire environment is running under MCP. In this case, the Red Hat JBoss Enterprise Application Platform (JBoss EAP) must be used; running in a JProcessor.

Other options are possible, for example an external web application server with JBoss running in the JProcessor providing an EJB container – see [techbrief: Java] for a discussion of the possibilities.

The integration component is an EJB, which links between the Web Services engine and the Java resource adapters, which are the integration middleware, connecting to the MCP applications and databases. Web Services engines are available as products with Java EE application servers.
Figure Web Services-3: Two Possible Configurations using Java EE as the Web Services Interface for MCP Applications and Databases

Figure WebServices-4 expands on the right hand side of Figure WebServices-3, showing a possible Web Services implementation using a COMS application environment with JBoss EAP running in the JProcessor providing the Web Services interface. The COMS transactions act as Web Service providers, as shown on the left of the figure, and as consumers of Web Services implemented in other systems, as shown on the right.

For incoming Web Services requests, the Web Services engine invokes an EJB wrapped as a Web Service. The EJB performs the required data mapping and invokes the JCA-compliant Transaction Resource Adapter, which connects to the COMS environment through Network Services and Connector PSH (CONNPSH). [techbrief: Sockets]

For outgoing Web Service requests, as shown on the right of the figure, the transactions invoke the resource adapter via Connector PSH and Network Services to invoke an EJB. The EJB uses the Web Services engine to make the Web Services request.

The same approach can also be used with JBoss EAP running off the MCP platform. The resource adapter would then connect to across a network. The off-platform architecture could also be implemented using another Java EE application server, for example WebLogic Application Server, using a suitable Web Services engine.

Although the above example assumes a COMS application, a similar approach could be followed for Open DTP applications, using the appropriate resource adapter.
Implementations using .NET support COMS and Open DTP as service providers and consumers. In the .NET implementation of Web Services, the enabling software is of course off-platform. Figure Web Services-5 illustrates the architecture.

Two different integration components are available; which component is used depends on whether or not the application uses Open DTP. [techbrief: TXINT]

- Open DTP uses DTI, specifically the OpenTI feature, with the OpenTI run-time environment providing the OSI-TP connection to Open DTP. This would be used for both incoming and outgoing requests.
- COMS uses COMTI as the integration middleware because DTI does not support connections to COMS.

Other approaches are possible for implementing Web Service access to MCP applications with either Java or .NET, or even other technologies. Ultimately, access from Web Services requires the stack shown in Figure Web Services-2. The implementation can be contained entirely within the ClearPath MCP environment using JBoss or spread across several platforms using Java or .NET.

One approach is to use message queuing as the integration middleware. This could even be used on-platform to provide a deferrable request to an application. The entire Web Service Enabling Software could be located in an ESB with a connector to the MCP system.
ClearPath ePortal

ClearPath ePortal is a multi-function product, providing a range of options from simple Web enablement of applications to mobile devices and Web Services. The product enables rapid development and deployment of new functions, requiring few specialised skills.

ClearPath ePortal contains three major components: developer, manager and run-time environment. Figure ePortal-1 is a schematic of the architecture.

ePortal connects to Web browsers, and the smart phones and tablets shown in the figure to applications in the MCP environment. The applications do not need to be modified, so access to them can be extended rapidly, allowing new classes of user. An example could be travelling sales staff who need to check stock availability while with a client. Orchestration features are provided: one external request can be delivered by multiple requests to ClearPath applications.

As shown at the top of the figure, Web Services access to applications is also supported, including orchestration options. Note that, using ePortal, ClearPath applications function as service providers; they cannot be consumers, although applications in the ePortal can consume services. That is, they cannot initiate outbound service requests. If outbound requests are required, other technologies are available, for example using Java. [techbrief: Web Services]
ClearPath ePortal Developer, which is a plug-in to Visual Studio, is used to generate the connections to the applications. Table ePortal-1 shows the data source and the type of input expected.

<table>
<thead>
<tr>
<th>Data source</th>
<th>Type of input</th>
</tr>
</thead>
<tbody>
<tr>
<td>COBOL</td>
<td>COBOL data record layout or SDF screen definition</td>
</tr>
<tr>
<td>EAE/AB Suite</td>
<td>EAE/AB Suite Ispec definition</td>
</tr>
<tr>
<td>T27</td>
<td>T27 screens or SDF screen definition</td>
</tr>
</tbody>
</table>

Table ePortal-1: Data sources and the type of input

The Unisys Mobile Hybrid Application Build Service extends the range of options. The mobility extensions of ClearPath applications can be combined with the native capabilities of smart device services such as GPS, maps, barcodes, scanning, compass, camera image and video capture.

ClearPath ePortal Manager is a browser based point-and-click management environment which automates processes required for deploying, configuring and managing application solutions.

**Message Queuing Middleware (MQM)**

Message queuing is a powerful technology. It can be contrasted with what may be referred to as a request/response technique, such as a remote procedure call. In message queuing, the sending program communicates by placing messages on queues. The target program or programs take the messages off the queues and respond, if necessary, by sending messages back through queues.
Message queuing offers two significant benefits:

- The co-operating programs do not have to be executing at the same time. Message queuing middleware (MQM) provides various options for secured storage, delivery assurance and guaranteed non-duplication of messages.
- The co-operating programs do not have to be implemented using the same technology, apart from using the same MQM. All they need to do is to agree on the message format and content.

These features make message queuing an ideal technique for deferred communications, especially between programs in relatively independent applications. It can also be used for an almost real-time connection, assuming that the appropriate options are chosen in the MQM.

There are some limitations on what can be done with message queuing. For example, if databases are updated in a number of different systems, and message queuing is used to send the messages between the systems, two-phase commit cannot be used to maintain database integrity. Some other strategy is required, such as reconciling databases periodically.

This techbrief describes two MQM products available with ClearPath MCP systems: WebSphere MQ for ClearPath and ClearPath MCP Interface to Microsoft Message Queuing (MSMQ).

**WebSphere MQ for ClearPath MCP**

WebSphere MQ for ClearPath MCP enables ClearPath MCP application to interwork with other WebSphere MQ implementations. The implementation is split between MCP and Windows environments. WebSphere MQ from IBM runs under the Windows environment, connected to the MCP environment through the MQ Gateway. This approach minimizes the load on the MCP by moving all the queue management to a Windows server, which can be an Intel partition within a ClearPath system or a separate Intel server.

WebSphere MQ for ClearPath MCP includes the following features:

- It is integrated with COMS and the Transaction Manager, and it supports the ability to include message queues as well as databases within the context of a global transaction boundary, synchronizing DMS II updates with message queues. The product supports all aspects of bidirectional, two-phase commit transactions, with rollback and full recovery in the event that the transaction cannot be committed or completed.\(^\text{17}\)
- It supports Java Messaging Service (JMS) for Java programs running in a ClearPath environment.
- It provides dynamic workload distribution, sharing workloads among available queue managers in a cluster. Workload balancing is automated for high availability and reliability during peak periods. Administrative tools are provided for managing queues and clusters.
- The product can be managed by standard systems management tools, such as Perfmon.

Figure MQM-1 is a schematic of the architecture of the WebSphere MQ for ClearPath MCP product.

Two major paths to the applications running under MCP are possible.

- One path provides a bridging capability to COMS direct-window applications. The applications do not need to be changed when using this path. The two different versions of the bridge are shown in figure MQM-2 and discussed in text.

\(^{17}\) When a transaction completes, the decision must be made about whether or not database updates may be performed. (See [techbrief: ODTP].) Message queues may also need to be treated as resources subject to transactional integrity. For example, a transaction may require database updates to be made and a message placed on a queue for subsequent processing somewhere. If the transaction cannot commit, the databases should not be updated and the message should not be placed on the queue.
• The second path is through the MQ Gateway Library, which provides the standard IBM interfaces of MQI to user applications and MQAI to management applications. As shown in the figure MQM-3, COMS applications can also access MQI directly. In this case, some changes are required to applications.

![Schematic of WebSphere MQ architecture](image)

Figure MQM-1: Schematic of WebSphere MQ architecture

Figure MQM-2 shows the two versions of the COMS direct-window bridge.

• The one on the left makes use of the CCF PSH (Custom Connect Facility Protocol Specific Handler). It provides a user exit capability, which allows users to write own code that can be called before forwarding a message to COMS or after receiving a response from it.
• The other makes use of a special PSH written for that purpose. It provides higher performance, but not the flexibility of the user exit capability.

The other interface between applications is the MQ Gateway Library, which uses RPC Server over TCP/IP to provide both the MQI and MQAI interfaces. A language library is provided for COBOL and C Languages. Figure MQM-3 shows the details of the interface. COMS and other user applications use the MQI interface; control applications use the MQAI interface.
Figure MQM-2: Schematic of the two versions of the COMS direct-window bridge

Figure MQM-3: Schematic of the MQI and MQAI using the MQ Gateway library
ClearPath MCP Interface to Microsoft Message Queuing

ClearPath MCP Interface to Microsoft Message Queuing (MSMQ) provides a gateway between Windows and the ClearPath MCP environment. The implementation is similar to that used for WebSphere MQ. The Windows portion of the gateway interoperates with an MSMQ server to provide access to MSMQ queues.

The gateway provides both a bridge between MSMQ and COMS direct-window applications and a complete API (WinMQ) for all applications based on MCP.

The COMS bridge enables unmodified COMS direct-window applications to receive input from and send responses to MSMQ queues.

As with the WebSphere MQ interface, two parts exist to the COMS direct-window interface:

- A CCF PSH
- A Connector PSH

With the WinMQ API, applications can be written using the API calls. Existing applications not using the API must therefore be modified. The API conforms to Microsoft specifications.

File Transfer Protocol (FTP)

File transfer is not usually regarded as middleware but is widely used for communication between different parts of an organization or between separate organizations for exchanging business data. Much of the business conducted using Electronic Data Interchange (EDI) uses FTP, and the Enterprise Application Integration (EAI) and Enterprise Service Bus (ESB) products typically include FTP as one of the supported interfaces. It has the advantages of simplicity, universal availability, including MCP, and the capability to carry large amounts of data.

Although message queuing is likely to replace some of the file transfer activity, and Web Services will replace other forms of electronic business, FTP is likely to persist. A major virtue is that it is even less dependent on how the receiving application works than is the case for message queuing.

FTP Services for ClearPath MCP provides enterprise server TCP/IP file transfer services. It can be used with in a TCP/IP local or wide area networking environment, in both private and public networks, such as the Internet.

FTP Services is an implementation of the File Transfer Protocol (FTP) as described in MIL-STD 1780 (RFC 959). The user interface is provided by the COPY command, which is available interactively through a Menu-Assisted Resource Control (MARC) menu or the Command and Edit (CANDE) message-control system, and in batch mode through Work Flow Language (WFL) or from the operator display terminal. The FTP server allows users at remote hosts to display lists of files.

Secure File Transfer Protocol (SFTP) is provided by Secure Shell (SSH) for ClearPath MCP, which is an implementation of the industry-standard Secure Shell (SSH) protocols. Inbound and outbound file transfers using SFTP are supported.

TXINT

The Distributed Transaction Integration (DTI) product was formed by combining the Transaction Integrator (formerly WebTX) and the Open Transaction Integrator (OpenTI) products. The two products overlapped in handling interfaces to Microsoft systems, and the naming was confusing because the Transaction Integrator was just as open as OpenTI. In combining the products, some rationalisation was carried out, optimising the code and reducing duplication.
When used with ClearPath MCP systems, DTI connects a variety of external clients to Open DTP applications running under MCP. [techbrief: ODTP] DTI does not connect to native COMS applications; a different environment integration product (COMTI) is used. This brief describes DTI as it is used with MCP, and COMTI.

**DTI**

Figure TXINT-1 is a high-level view of DTI, showing the client technology options. DTI itself runs under Windows.

![Figure TXINT-1: High-level view of DTI showing client technologies supported](image)

Once DTI is configured to access an MCP application, all client types supported can access it. The client types can be divided into two groups:

- Component clients, which access the system by making a component request of some form, for example a DCOM request
- Web clients, which access the system through a web server (Microsoft IIS), for example using HTML

Some features have been superseded, in particular for Java and CORBA (Common Object Request Broker Architecture) clients, although the interfaces remain. The Java resource adapters are recommended for Java. [techbrief: Java] CORBA is not currently of great interest so support may be dropped. If it is required, Java EE specifies an interface to CORBA.

The interface for a component client is provided by an adapter, which handles the remote client. The adapters then use one of a number of component gateways, which convert the data from its originating form into that required by the MCP application and vice versa. The component gateways use one of the connectors to access the MCP applications.

Web clients connect to a web gateway; which one depends on the form of the input. The gateway maps the data to the form required by the application and, as was the case for component gateways, uses a connector to access it.

Figure TXINT-2 shows the run-time architecture of DTI, identifying the specific component parts. At the top is OpenTI, which provides connections between COM+/.NET and Open DTP (or other implementations of the Open Group DTP model). The remainder is the Transaction Integrator runtime environment, providing inbound requests only for Open DTP.

OpenTI provides a bidirectional access between Microsoft COM+/.NET and Open DTP. The connection is symmetrical; either end can initiate a request. OpenTI wraps Open DTP services, making them appear to be other COM+/.NET components to the Microsoft end, and similarly wraps COM+/.NET components, making them appear to be Open DTP services.
services to the Open DTP end. By using the Oracle Tuxedo Mainframe Adapter OSI-TP environment integrator, the interworking extends to Tuxedo.

![Diagram showing interworking between OpenTI and Open DTP](image)

**Figure TXINT-2: Schematic of DTI run-time architecture**

OpenTI supports transactional integrity in a global transaction executed over COM+/.NET, Open DTP, other implementations of Open Group DTP and Tuxedo systems, where all the databases can be subject to two-phase commit processes. Although connections to Open DTP can be made from other DTI clients through adapters and GGate, XGate, or CXGate, transactional integrity is not supported.

In addition to the run-time environment, OpenTI provides a development environment for users to produce OpenTI components to perform the mapping between the COM+/.NET data structures and Open DTP views.

The Transaction Integrator part of DTI is made up of the adapters shown in the upper right part of figure, supporting the component clients shown. The adapters function as servers, which are invoked by the clients using the relevant technology, for example a DCOM request.

The adapters then use one of the three gateways shown to perform the appropriate data mapping functions. First, GGate, which is a generalized gateway, handles data as name/value pairs.

The other two gateways are XML gateways. DTI supports XML handing as follows:

- XML may be passed between XML-aware applications in MCP and external environments.
- An external XML-aware application may communicate with an MCP application that is not XML aware; DTI performs the mapping between XML and Open DTP views.
- An external application that is not XML aware may communicate with an XML-aware MCP application; again, DTI performs the required conversions.
- Finally, applications that are not XML-aware may communicate with each other.

XGate, which is a generalized XML gateway, provides automatic conversion between Open DTP views to or from XML. CXGate is an enhanced version of XGate, handling additional data types, not just ASCII.
GGate and XGate can use connectors to access the Open DTP applications, as shown in Figure TXINT-2. The Open DTP connector is used to access Open DTP. The connection is made by using the Open DTP connector provided by OpenTI run time. CXGate uses the Open DTP connector provided by OpenTI.

Web clients use one of three gateways: ViewGate, JGate, or AutoGate. ViewGate is a relatively simple interface, providing access for client input data in HTML. It maps the input to and from the form required by Open DTP.

JGate is more sophisticated, providing facilities for downloading Java Applets to the browser.

AutoGate is somewhat different. It allows programs in any of the application types to publish information to DTI. The information is then available in HTML form for clients to retrieve.

**COMTI**

COMTI (formerly Microsoft Transaction Integrator) provides capabilities for integrating the Windows platform with IBM systems. Unisys worked with Microsoft to create the Unisys ClearPath Interface to Microsoft Transaction Integrator (formerly COMTI for ClearPath MCP) product to provide integration with ClearPath MCP systems. The product name changes reflect the addition of .NET framework capabilities to the original Component Object Model (COM) capabilities. The current products are often referred to as COMTI for simplicity; they are referred to as COMTI in this paper.

COMTI provides a tool called Designer, which is hosted in Microsoft Visual Studio, for defining the interface contract between clients and the server. This interface can be defined in .NET or COM and can be imported and exported from one to the other. The interface contract is key to the integration between the client (consumer) and server (provider). The interface definition encapsulates the data contract, which has capabilities that are exceptional in terms of flexibility. The data contract can include complex data types (User Defined Types), data tables, or record sets. These complex data types are often referred to as records. The provider can return a single record, a fixed number of records, a number of records depending on another variable, or a fully variable number of records. In addition to the structured types, all of the base, or primitive (integer, string) data types are supported. These data types are in turn mapped to data types compatible with the MCP environment. Figure TXINT-3 illustrates the elements of the MCP implementation of COMTI.

---

**Figure TXINT-3: Schematic of COMTI architecture**
An interface can be defined and deployed using Visual Studio as a web service without writing any code in the Windows environment. The same interface definition can be used with a COMS interface, which can be deployed for use by scripting languages such as VBScript using late binding or a desktop application that uses early binding.

Once defined, the interface can be hosted at run time in a COM+ application, as an IIS application or self-hosted.

By default, connections to the MCP server are maintained in a connection pool. This approach significantly enhances scalability. All these connections present the same security credentials to the server, and Kerberos or NTLM are supported. Clients also have the option to control the connection, using the session capability. This capability allows the client to create, destroy, and explicitly log on to the session.

Interactions initiated by MCP are also supported. When a COMS application connects to a COMTI service that is based on Windows, COMTI invokes the defined service. This service could invoke a web service, access data or whatever is required. The output from the service is then returned to the COMS application.

### Data access

Data contained in MCP-resident databases can be remotely accessed in one of two general ways. In the first option, the remote requestor invokes a program, which runs in the same system as the database. The database itself is not seen directly by the remote requestor; the program delivers services, which involves database reads and possibly updates. The second option is to allow the requestor to access the database directly, bypassing any existing application programs.

The two approaches are not mutually exclusive. For example, a transaction system could be serving a wide user population, where the transaction programs read and write to a database. Small groups of analysts could be investigating trends revealed by the data, directly extracting data from the database and manipulating it using suitable analytical tools.

Remote data access for analytical reasons can affect performance if the database is simultaneously being used for transaction processing. For this reason, data in a database used by a transaction system may be copied into another system for query and analysis, rather than just using one online copy for both transactions and analysis.

This technical brief describes three products for accessing DMS II and other MCP data sources:

- ODBC Access (formerly called Data Access)
- Enterprise Database OLE DB Data Provider for ClearPath MCP (formerly OLE DB)
- XML Provider for ClearPath MCP

Note that Java resource adapters provide JDBC access to both DMS II and Relational Database Server. For details, see [techbrief: Java].

### ODBC Access

Figure Data Access-1 shows the architecture of ODBC Access. ODBC Access provides SQL access from Windows clients to non-relational databases, including DMS II, EAE and KEYEDIOII. It permits the use of widely-available tools and techniques to access DMS II and other data.

The following types of access are supported:

- An ODBC driver interface
- Interactive SQL
ODBC Access comprises a Windows ODBC client driver, through which Windows applications and tools such as Excel can access the data sources. The ODBC Access Server provides access into the two databases. It is divided into a part resident in a Windows server and a part running under MCP, as shown in Figure Data Access-1.

A key component of ODBC Access is the relational mapping utility, which generates a relational view of the non-relational DMS II data structures. This view enables ODBC Access to operate directly against current DMS II data; no schema changes are required and no data is duplicated.

ODBC Access provides several levels of security, including mandatory sign-on, the ability to restrict access to data by selected users, and the ability to restrict access to inquiry only or to provide full update capability. It allows user-level limits to be set on queries, allowing different limits to be set for each user.

**Enterprise Database OLE DB Data Provider for ClearPath MCP**

Enterprise Database OLE DB Data Provider for ClearPath MCP is the Unisys implementation of the Microsoft OLE DB for access to DMS II and EAE data. OLE DB is designed to provide universal access to relational and non-relational data sources. OLE DB includes the concepts of a consumer and a data provider:

- The consumer is a typical client application or tool requiring access to data; Microsoft Excel and languages such as Visual Basic and Visual C++ are examples.
- The data provider exposes the OLE DB interface and places data into a tabular format in response to calls from the consumer. The Unisys OLE DB Data Provider for access to DMS II and EAE is an example of such a data provider.

An optional service component implements a common set of OLE DB functionality on top of an OLE DB data provider. Query processors/engines and cursor engines are examples of service components.

The client application/tool can communicate directly with the OLE DB interface or can use the ActiveX Data Object (ADO) interface via Visual Basic. ADO is very simple and makes OLE DB programming very easy.
The ClearPath OLE DB is an implementation of the Data Provider for DMS II and EAE data. Figure Data Access-2 shows the architecture. The implementation consists of a number of components.

The Data Provider Object exposes the standard OLE DB interfaces to the client application in a Windows platform. It handles error messages as well as data conversions from DMS II and EAE data types to user specifiable types. It communicates with the Worker in the MCP system through the Transport Object on the client and the Server Library on the MCP system. These components marshal the data for transmission across the interconnecting TCP/IP network.

The Data Provider Object and Transport Object must all be on the same Windows platform as the Client Application, which can be web-enabled.

The Worker receives the marshalled queries from the Provider, makes the appropriate database calls to apply the query, and then returns the results to the Data Provider Object through the Server Library and the Transport Object. It also monitors the network for incoming connection requests.

Figure Data Access-2: Architecture of Enterprise Database OLE DB Data Provider for ClearPath MCP

**XML Provider for ClearPath MCP**

The XML provider enables DMS II and other data sources to interoperate with other XML-enabled applications and data sources locally or anywhere on the Internet. Figure Data Access-3 shows the architecture of the XML Provider, which runs in a Windows environment.

The XML Provider offers these features.

- Renders data in a browser. The data can be sent to the requesting browser in XML format, or the extracted XML can be converted to HTML format.
- Exports data in XML format. The XML document can be returned to a browser, sent to and processed by an application, or forwarded to another data source.
- Imports data contained in an XML document to a database. External data can be sent to DMS II and stored in existing datasets.
- Provides data exchanges with other data sources. The use of XML to exchange overcome format problems between incompatible systems.
- Allows integration of data with other XML data. Data extracted from DMS II can be appended to XML data of similar structures and sent to a browser or other destinations.
- Supports retrieval of database description information in XML Schema format. XML schemas are valuable for data validation specifying XML document structures for the exchange of information with other organizations.
- Supports SOAP messaging. The XML Provider forms have been enhanced to enable specification of responses in plain XML format, document-style SOAP format or RPC-style SOAP format. XML Provider also accepts incoming SOAP messages. The support of SOAP and other standards means that the XML Provider allows DMS II and other MCP data sources (KEYEDIOII files and flat files) to participate in a Web Services environment as service providers.
- Provides enhanced support for embedded structures (embedded datasets) in both data export and schema description.

Figure Data Access-2: Architecture of XML Provider for ClearPath MCP
Two tools facilitate data integration and transformation:

- The Data Integration Tool enables the construction of custom reports.
- The Transformation Design Tool generates XSLT\(^{18}\) style sheets to map a source XML Schema to a target schema. The XSLT style sheet is used to transform data as part of the export or import process.

**Application Data Access for ClearPath MCP**

Application Data Access for ClearPath MCP systems (ADA) is based on the Microsoft ODBC programming interface. It enables application programs on a ClearPath MCP server to execute a subset of standard ODBC function calls, including the ability to use standard SQL statements, against any database on Windows platforms that provide an ODBC driver. All major databases are supported, ranging from simple PC databases to enterprise databases such as Oracle. Figure Data Access-3 shows the architecture.

Figure Data Access-3: Architecture of Application Data Access for ClearPath MCP systems

COBOL, ALGOL, and C applications make calls to Application Data Access to submit standard ODBC and SQL requests, which are then passed on to the ODBC driver manager for access to any configured data source. Access is not restricted to retrieval. Application Data Access MCP applications can also update data in databases and files in the enterprise that are not related to Unisys.

The Application Data Access API is a subset of the ODBC version 2 specification. It includes a few calls to establish the initial connections to the Windows server to control buffering and to handle internationalization.

\(^{18}\) The XSL (eXtensible Style Language) is a family of recommendations for defining XML document transformation and presentation. XSLT is a language for transforming XML.
Client Access Services

Client Access Services provide rather more than just data access. The product enables workstations to interoperate with the MCP environment in the same way that they interoperate with the Windows Server. The MCP environment is integrated into a Windows-based network, including support for client/server distributed applications. No additional workstation-resident software is required.

Client Access Services enables users to access MCP resources and services, including:

- Browsing resources;
- Accessing shared resources, such as disk directory, CD-ROMs, and printers;
- Providing access to server files for workstation applications;
- Using named pipes in distributed applications;
- Sharing workstation-created files with other users;
- Storing data on a centralized server to facilitate backups; and
- Allowing manipulation of existing data by off-the-shelf workstation tools.

Sockets

Sockets interfaces for MCP applications were originally provided through Port Files, with options to use Microsoft Named Pipes. However, Port Files do not allow the use of UDP or access to SSL functionality. The MCP Sockets Services API provides these facilities. This technical brief discusses Port Files and the MCP Sockets Services API.

Port Files

The MCP Network Services provide a Port File connection to TCP/IP sessions. Client Access Services supplies the capability to handle Microsoft Named Pipe connections. The key functional difference between sockets and Named Pipes is that Named Pipes are integrated with the Microsoft security system. The Windows usercode and password are passed to Client Access Services, which then uses them to make an MCP logon request.

Figure Sockets-1 shows the major building blocks for ClearPath MCP Port File and Named Pipes connectivity.

Application programs can choose to handle the Port File directly; alternatively it can terminate in the CCF (Custom Connect Facility) which will convert the dialog into a COMS dialog. From the point of view of a COMS application program, the port is accessed in exactly the same manner as a terminal.

Using CCF and COMS has the following advantages:

- The COMS applications are unchanged.
- Additional COMS features are used such as transaction key routing, security, and running additional program copies when the load is high.
- CCF provides a generic framework for adding interfaces into COMS. The structure of the CCF is illustrated in figure Sockets-2.
Three of the PCMs are used for communicating to the outside world:

- The Named Pipe PCM handles Named Pipe input from the Client Access Services.
- The TCP/IP PCM handles TCP/IP sessions.
- The WEB PCM is an interface into the WebTS.
The CUCI PCM is used for communicating with the COMS kernel.

Two PCMs provide service functionality:

- Term PCM implements terminal services functionality including character set translation and supporting the break command on output
- Logon PCM implements logon security checks for Named Pipes

As an example of how the service functionality is used, WEB PCM traffic is routed straight to the CUCI PCM because it does not use these services but TCP/IP traffic is typically routed through the Term PCM.

**MCP Sockets Service API**

The MCP Sockets Service API provides an interface in the style of the Berkeley Software Distribution sockets interface, rather than the Microsoft Windows sockets interface. The API supports access from ALGOL/NEWP, C programs, COBOL programs, and the JVM. Blocking TCP and UDP sockets are supported; sending and receiving of multicast datagrams and SSL are also supported.

Figure Sockets-3 provides a high level schematic of the MCP Sockets Service.

---

**Advanced Development Environments (ADE)**

In addition to enabling rapid application development, Enterprise Application Environment (EAE) contains a number of infrastructure features that can be considered as middleware. Agile Business Suite (AB Suite) is an evolution of EAE and includes new run-time environment generation options. It will eventually replace EAE. This tech brief looks at the middleware features of the two products.

**Enterprise Application Environment**

EAE has two generation options for MCP systems:

- It can generate for a COMS environment.
- It can generate Open DTP services, where each individual EAE Ispec (transaction) can be generated as a service.
If the Open DTP option is used, EAE applications can take part in an Open Group DTP network just like other applications, co-existing with other Open Group DTP applications and, through environment integrators such as DTI and JCA-compliant connectors, with other systems, such as Tuxedo, .NET and Java EE. [techbrief: ODTP] [techbrief: Java] [techbrief: TXINT]

The Component Enabler is an environment integrator providing EAE facilities that are somewhat similar to DTI. It enables EAE applications to be opened to a number of different client types, using component technology to build the interfaces between the client and the application. The product interfaces to EAE applications on any EAE platform.

The Component Enabler generates a set of Java classes to interface to EAE Ispecs. These may be invoked from Java in Java environments or from C++, VB and so on. The path into the EAE application is an interface called Enterprise Application Remote Access, which provides sockets access to the EAE application. The EAE Ispecs always function as servers with this interface; they cannot function as a client, invoking other systems as services.

There is also a .NET option for Component Enabler. All of the client options except Presentation Client (which is itself a Java application – see the following text) can be deployed with either the Java or the .NET version of Component Enabler. Even custom client applications that have been written to use the Java-based Component Enabler can be easily modified to work in a Java-free environment if desired. In most cases the element that establishes the connection with the EAE host is the only thing that will need to change.

Component Enabler client options are as follows:

- **Presentation Client** – This Java application can be deployed to a desktop or a web browser.
- **ASP Client Generation** – This option deploys GUI forms using Active Server Page (ASP) technology. The generated ASP pages can be modified and customised after they have been generated, if desired.
- **Visual Basic Client Generation** – This option deploys GUI forms as a VB 6 application. The VB project can be customised if desired. The presentation and connectivity modules are defined in separate folders to make it easier to customise the presentation aspects of the client application without interfering with the host connectivity. This makes it easier to maintain the custom client even when the host application is modified and regenerated.
- **Web Services** – WSDL and SOAP interfaces are automatically defined for the selected transactions.
- **ASP .NET Web Forms generation** – This option is similar to the ASP generator except that the selected Ispec transaction forms are deployed using ASP .NET (Microsoft .NET 2.0). An added capability is the ASP .NET Web Form Renderer. This capability provides a custom Visual Studio control with the ability to render the generated forms in any ASP .NET project.
- **ASP .NET Web Services generation** – This option is similar to the ASP Generator. It generates a Web Services interface, which is based on the Web Services infrastructure provided by Microsoft .NET connected technology. It serves as an alternative to the EAE Release 3.3 Web Services generator, which is based on the Microsoft SOAP Toolkit. The SOAP Toolkit is currently on extended support from Microsoft.
- **Visual Basic .NET client generation** – This option is similar to the VB 6 generator but is updated for VB .NET.

Figure ADE-1 is a schematic of the deployment environment. The products may be deployed in a PC, in a two-tier architecture or in a server within a multi-tier distributed application environment. In addition to the inbound client interfaces in which EAE applications act as services, EAE applications can also participate as clients, calling other services.

- **Open DTP** [techbrief: ODTP] – EAE applications can participate as both client and server. When acting as a client, an external XCOMMON or X-CTYPE buffer definition can be imported into the EAE Developer to make calling the Open DTP services very fast.
- **Business Integrator** – This capability consists of a development part and a run-time part. The run-time component resides on a Windows server and enables an EAE application to call external Web Services, .NET, or COM components, plus scripts and other common Windows applications. Part of the EAE Client Tools environment, this allows an EAE application to call out to an external component using the same AUTO syntax that is used with EAE to EAE interactions.

- **EAE-to-EAE HUB** – This capability provides a proprietary interface that allows an EAE application to invoke transactions in another EAE application capability to preserve the integrity of global transactions.

![Diagram of Component Enabler Environment](image)

**Figure ADE-1: Component enabler environment**

**Agile Business Suite**

AB Suite is an evolution of EAE. Like EAE, it provides a powerful, model-driven development toolset, with multiple generation options for run-time environments from the same source. AB Suite generates for COMS on MCP systems, including the Open DTP option. The generation option is .NET on Windows.

The Component Enabler product described above connects to AB Suite as well as to EAE. All of the client options that are available with EAE are offered with AB Suite, plus a native WinForm client option is available when AB Suite applications are deployed to a Windows .NET run-time environment.

In addition to the ability to expose Ispec (transaction) interfaces using the Component Enabler middleware options as previously described, AB Suite offers the ability to define component-level methods that can be exposed as public methods and called from a variety of SOA environments. AB Suite can also consume external services, calling external public methods. When deployed in a ClearPath MCP run-time environment, external methods are typically made available outside of the MCP environment using COMTI.
Appendix: Glossary

This paper has used a number of technical terms, acronyms and product names. This appendix gathers them in one place for easy reference.

Product names can be complicated, as companies may change the name of a product during its lifetime for various reasons, including:

- To refresh the image for marketing reasons
- To expand or change the product so that it is really a new product
- To include it within a family of products, under a new umbrella name

Examples of Unisys product name changes are DMS II to Enterprise Database Server for ClearPath MCP and Communications Management System (COMS) to Transaction Server. A number of separate IBM products were grouped together in the WebSphere family and renamed, so MQSeries, for example, became WebSphere MQ. The evolution of Microsoft’s distributed system technology from its early days to .NET is one of the more complicated sequences of changes.

Although the terms have generally been explained or expanded in the text, they are gathered in this glossary for quick reference.

**AB Suite**: Agile Business Suite is the latest in the evolution of a product originally called LINC then EAE (Enterprise Application Environment). AB Suite is only available on ClearPath MCP systems. EAE is available on both families.

**ACID**: The ACID properties of a transaction are as follows. *Atomicity*: A transaction should be completed and committed, or rolled back if there are problems during its execution: it is an atomic entity in that it is indivisible. *Consistent*: Changes made to databases must be from one valid state to another, not something indeterminate. *Isolated*: The results of a transaction should be invisible to other transactions until the transaction is completed. *Durable*: Changes to databases, which have been made during a transaction, should be permanent and survive future media and other failures.

**ADA**: Application Data Access for ClearPath MCP systems. Provides access to any database on a Windows platform that provides an ODBS driver.

**ADO**: ActiveX Data Objects, a set of Microsoft COM objects for accessing data sources.

**API**: Application Program Interface.

**Applet**: Java program launched from HTML and run in a web browser. Can provide web applications with interactive features that cannot be provided by HTML.

**Application servers (Java EE)**: IBM’s WebSphere Application Server, Oracle’s (formerly BEA’s) WebLogic Application Server, Red Hat JBoss Enterprise Application Platform, and the Open Source JBoss Application Server are the most widely used.

**ASP**: Active Server Page.

**BIS**: Business Information Server. A Unisys product for data mining and rapid application development. It is the latest in the evolution of a product originally called MAPPER.

**CANDE**: Command and Edit language. A language for interactive use of MCP systems.

**CCF PSH**: Custom Connect Facility Protocol-Specific Handler (PSH). Middleware providing a path between MCP transports and COMS.
**CGI:** Common Gateway Interface. A standard method used to generate dynamic content on Web pages and Web applications.

**CICS:** Customer Information Control System (CICS) is a transaction server that runs primarily on IBM mainframe systems under z/OS and z/VSE.

**Client Access Services:** Integrates an MCP environment into a Windows-based network. It does not require any special software on the Windows workstation.

**Component Enabler:** A Unisys environment integrator product connecting Microsoft .NET and Java environments into EAE and AB Suite applications.

**COM:** See .Net.

**COMS:** Communications Management System, the native transaction processing environment on ClearPath MCP systems, now called Transaction Server.

**COMTI:** (for ClearPath MCP). A Unisys environment integrator product connecting Microsoft COM+.NET to COMS. Now called ClearPath MCP Interface to Microsoft Transaction Integrator.

**Container:** Used in this paper to indicate an application server environment: a web container, in which JSPs or Servlets execute, and an EJB container, in which EJBs execute.

**CORBA:** Common Object Request Broker Architecture. A standard, defined by the Object Management Group (OMG), for building distributed systems using object principles.

**CRM:** (1) Customer Relationship Management. (2) Communications Resource Manager: an element in the Open Group DTP model. The CRM provides the API functions needed to support communication between application programs. The API set is called XATMI. (Other APIs are possible for different CRMs.)

**DMS II:** Database on ClearPath MCP systems, now called Enterprise Database Server for ClearPath MCP

**DTI:** Distributed Transaction Integration. A Unisys environment integrator connecting a variety of external technologies to Open DTP.

**DTP:** Distributed Transaction Processing.

**EAE:** Enterprise Application Environment. See AB Suite.

**EAI:** Enterprise Application Integration. A concept arising before the ideas of services and SOA but is similar and overlaps with service concepts.

**EDI:** Electronic Data Interchange. A generic term for conducting business electronically.

**EIS:** Enterprise Information System.

**EJB:** Enterprise Java Bean. A software entity defined by Java EE. Executes in an EJB container.

**(ClearPath) ePortal:** Unisys product providing access to COMS, Open DTP and EAE transactions as Web Services, as well as access from web browsers and a variety of mobile devices such as smart phones and tablets. Includes orchestration of transactions to deliver more complex services externally. The ClearPath ePortal is implemented as a specialty engine.

**EPP:** Enterprise Partitionable Platform. A part of a ClearPath fabric-based system containing secure partitions able to run Linux or Windows.
**ESB**: Enterprise Service Bus. A generic name for products aimed at the implementation of SOA.

**FTP**: File Transfer Protocol. A standard defined by the Internet Engineering Task Force (IETF).

**GUI**: Graphical User Interface.

**HTML**: HyperText Markup Language.

**HTTP/S**: HTTP is HyperText Transfer Protocol. HTTPS is a secure version of the protocol.

**IEC**: International Electrotechnical Commission.

**IETF**: Internet Engineering Task Force: a body of volunteers defining technical standards and guidance documents.

**IIS**: Internet Information Services. Microsoft’s web server product.

**ISM**: I/O Specialty Partition Module. A part of a ClearPath fabric-based system containing I/O and specialty partitions.

**ISO**: International Organization for Standardization.

**Ispec**: Interface Specification. An EAE or AB Suite transaction.

**Java**: An object-oriented programming language.

**Java EE**: Java Platform, Enterprise Edition, formerly known as J2EE (Java 2 Enterprise Edition). Set of standards defining a Java environment suitable for large scale applications. Implementations are called Java EE application servers.


**JAX-WS**: Formerly JAX-RPS (Java API for XML-based RPC) but JAX-RPC 2.0 was renamed JAX-WS 2.0 (Java API for XML Web Services). Allows a Java application to invoke a Java-based Web service with a known description.

**JBoss EAP**: Red Hat JBoss Enterprise Application Platform is an enterprise version of the popular open source Java EE application server known as JBoss Application Server. The product on ClearPath systems is JBoss Enterprise Application Platform for ClearPath MCP or MCP.

**JCA**: Java EE Connector Architecture. Standard application programming interface (API) for connecting Java EE application servers to other architectures, for example COMS.

**JDBC**: Java DataBase Connectivity. Standard defined in Java SE for accessing databases.

**JDK**: Java Development Kit.

**JMS**: Java Messaging Service. An API, defined in Java EE, for sending messages between two or more clients.

**JProcessor**: Provides an environment for running Java applications. Implemented as a specialty partition in ClearPath systems supporting s-Par®, or as an appliance in other ClearPath systems.

**JRE**: Java Runtime Environment.

**JSP**: Java Server Page. A software entity defined in Java EE. Executes in a web container.

**JTS**: Java Transaction Service. Specifies the implementation of a Transaction Manager which supports the Java Transaction API.

**JVM**: Java Virtual Machine. Software environment in which Java programs execute. Can be implemented on any computer. Implementations are available for ClearPath systems.

**LINC**: See AB Suite.
Linux: Open Source operating system, gaining increasing popularity on Intel® hardware. There are several supported releases (for example Red Hat Enterprise Linux).

MAR C: Menu Assisted Resource Control. A facility for interactive use of MCP systems.

MCP: Master Control Program. Operating system for ClearPath Libra and earlier (A-series) systems.

MG SWeb: Web Services software product for MCP systems, supplied by MGS, Inc.

MQAI: Standard IBM interface to message queuing management applications.

MQI: Message Queuing Interface.

MQM: Message Queueing Middleware.

MSMQ: Microsoft’s message queueing middleware.

ODBC: Open DataBase Connectivity. API for accessing database management systems, independent of programming language, database system, and operating system.

OLE DB: Object Linking and Embedding DataBase. A set of COM-based interfaces exposing data from a variety of sources, such as spreadsheets, project plans, SQL, legacy databases and more.


ONC RPC: Open Network Computing Remote Procedure Call (RPC).

Open DTP: Full name is Open Distributed Transaction Processing. Unisys implementation of the Open Group Distributed Transaction Processing (DTP) model

Open Group: Vendor-neutral industry consortium, formed by the merger of two earlier consortia with a similar purpose: X/Open and the Open Software Foundation (OSF). Its specifications include the Open Group Distributed Transaction Processing (DTP) standard and UNIX.(UNIX – all caps – is a trademark held by the Open Group.)

Open TI: See DTI.

Oracle Tuxedo Mainframe Adapter OSI-TP: An environment integrator for connecting Tuxedo to implementations of Open Group DTP. Uses OSI-TP on the Open Group DTP side, converting from the equivalent technology used by Tuxedo. Formerly called eLink OSI-TP

OSI-TP: A protocol defined by the International Organisation for Standardisation (ISO) for distributed transactions. Includes facilities for ensuring consistency of multiple database updates.

OS 2200: Operating System for ClearPath Dorado and earlier (2200-series) systems

PMM: Processor Memory Module. A part of a ClearPath fabric-based system containing the firmware to house the MCP environment.

POJO: Plain Old Java Object. Name given to Java programs running in an unmanaged environment in a JVM, as opposed to within a managed environment, usually provided by a Java EE application server.

RA: Resource Adapter. A name for a connector from Java environments to applications or databases.

Relational database server: Relational database server product for ClearPath MCP systems.

Resource adapter: Java logic to allow access to external resources – applications and data – from Java runtime environments, e.g. from Java EE application servers.
**REST-FUL:** REpresentational State Transfer. An architectural style originally defined by R T Fielding. Fielding defines an architectural style as follows: ‘An architectural style is a co-ordinated set of constraints that restricts the role/features of architectural elements and the allowed relationships among those elements within any architecture that conforms to that style’.

**RM:** Resource Manager. An entity defined in the Open Group DTP model. It includes databases and message queues.

**RPC:** Remote Procedure Call.

**Servlet:** A small Java program that runs within a Web server. Servlets receive and respond to requests from Web clients.

**SOA:** Service-Oriented Architecture.

**SOAP:** Defined by the W3C, SOAP is a protocol specification for exchanging structured information in the implementation of Web Services. It uses XML Information Set for its message format, and relies on other application layer protocols, most notably Hypertext Transfer Protocol (HTTP) or Simple Mail Transfer Protocol (SMTP), for message negotiation and transmission. Was originally an acronym for Simple Object Access Protocol, but is now just a name; the expansion is no longer used.

**Socket:** An endpoint of an inter-process communication flow across a network. Most sockets are Internet sockets.

**S-Par:** Secure Partition. Unisys-patented technology for providing software partitions with dedicated resources, i.e. it is not the same as a virtual machine as provided by, e.g. VMWare.

**SQL:** Structured Query Language. A language for accessing databases.

**SQL Server:** Microsoft’s relational database management product

**SSL:** Secure Sockets Layer. Standard security technology for establishing an encrypted link between a server and a client

**Stealth:** Unisys patented technology providing a variety of security features for ClearPath and other systems.

**TCP/IP:** Transmission Control Protocol/Internet Protocol. TCP and IP are important protocols defined by the IETF. The term ‘TCP/IP’ is also used to indicate the entire set of protocols for networking defined by the IETF.

**TM:** Transaction Manager. A component of the Open Group DTP model.

**Tomcat:** Open Source web application server from the Apache Software Foundation

**Tuxedo:** Distributed transaction application server (not Java EE). Developed originally by AT&T as a transaction processing monitor for Unix, Tuxedo was owned by BEA for a number of years until BEA was acquired by Oracle. The Open Group DTP model is very similar as it was derived from Tuxedo.

**TX:** An API set for the Open Group DTP TM for delineating transactions.

**T27:** A family of green-screen terminals with defined link and display protocols. Emulators available for PCs.

**UDP:** User Datagram Protocol. A simple connectionless transmission model with a minimum of protocol mechanism, carried over IP.

**Unix:** Operating system originally developed by AT&T at Bell Laboratories in the early 1970s. The specification is now owned by the Open Group. There are many variations of Unix.

**URL:** Uniform Resource Locator. A reference (an address) to a resource on the Internet.
**VB**: Visual Basic. A third-generation event-driven programming language and integrated development environment (IDE) from Microsoft.

**Visual Studio**: An integrated development environment (IDE) from Microsoft. Used to develop computer programs for Microsoft Windows, as well as websites, web applications and web services.

**WCF**: See .Net.

**WEBAPPSUPPORT**: A library providing services for developers of Web applications in MCP systems.

**WebLogic (Server, Integration)**: WebLogic is the name of a family of application infrastructure products from Oracle (originally BEA). WebLogic Server is the name of the Java EE application server product. WebLogic Integration is an enterprise application integration (EAI) product.

**WebPCM**: An interface from COMS to Web users through the WebTS.

**Web Services**: A set of technologies and standards defined by the W3C and others for implementing SOA. Web Services are specific case of SOA, not the only case.

**WebSphere Application Server**: A Java EE application server from IBM.

**WebSphere MQ**: IBM’s message oriented middleware product, formerly known as MQSeries but now included within the WebSphere family.

**WebTS**: Web Transaction Server for ClearPath MCP. Provides a Web-compatible interface directly to the ClearPath MCP series transaction processing environment. Also supports static Web pages, forms, graphics, Java applets, and other standard Web server features to facilitate the implementation of complete Web applications.

**WebTX**: See DTI.

**WFL**: Work Flow Language. A job control language for MCP systems.

**WinMQ API**: ClearPath MCP Interface to Microsoft Message Queuing (WinMQ)

**WIN RPC**: Microsoft Windows Remote Procedure Call.

**WSDL**: Web Service Definition Language. An XML format for describing network services as a set of endpoints operating on messages containing either document-oriented or procedure-oriented information.


**XA**: A system-level interface between a resource manager (RM) and the transaction manager (TM). XA provides the 2PC capability. A database or other resource manager that supports it is called XA compliant.

**XATMI**: An API set in the Open Group DTP model. See also CRM.

**XA**: A system-level interface from the Open Group DTP model. Used between a resource manager (RM) and the transaction manager (TM). XA provides the 2PC capability. A database or other resource manager that supports it is called XA compliant.

**XA+**: System level interface from the Open Group DTP model. Used between the TM and the Communications Resource Manager (CRM) to manage the thread of control.

**XML**: eXtensible Markup Language. A language defined by the W3C for encoding messages and protocols in a machine-independent way.
**XSLT**: Extensible Stylesheet Language Transformations. A language for transforming XML documents into other XML documents, or other formats such as HTML

**2PC**: Two-phase commit. A technique for ensuring database consistency if two or more databases are updated in a distributed transaction.

**.NET**: A Microsoft umbrella name covering a number of related technologies, which are concerned with building (transaction) systems from distributed components under Windows. Originated with OLE (Object Linking and Embedding) and COM (Component Object Model), and passed through various stages, including DCOM and COM+, to become the .NET Framework. Elements include: Model View Controller (MVC), the Windows Communications Foundation (WCF) and the Windows Workflow Foundation (WF).
Revision History

<table>
<thead>
<tr>
<th>Revision</th>
<th>Date</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>April 2001</td>
<td>First published version</td>
</tr>
<tr>
<td>2</td>
<td>April 2002</td>
<td>Updated for ClearPath MCP 7.0 Release</td>
</tr>
<tr>
<td>3</td>
<td>October 2007</td>
<td>Major revision to bring content up to ClearPath Release 11.1</td>
</tr>
<tr>
<td>4</td>
<td>May 2010</td>
<td>Updated for ClearPath Release 13.0, edited and reformatted</td>
</tr>
<tr>
<td>5</td>
<td>February 2015</td>
<td>Major revision to ClearPath release 16</td>
</tr>
</tbody>
</table>

About the author

Now an independent consultant, Peter Bye was a senior system architect in Unisys, based in London. His special area of interest is networked computing, including communications networking, middleware, and architectures. He has many years of experience in information technology, working as a programmer, analyst, team leader, project manager and consultant in large-scale customer projects in banking, transportation, telecommunications and government. He has also worked in software development centres, during which time he spent two years as member of an international standards committee working on systems management.

He has worked for extended periods in Sweden, Denmark, Finland, Norway, the USA, France and Spain, as well as the UK. He has presented at a wide variety of conferences and other events and is the author of a number of papers on networking, systems management and middleware. He is the co-author of a book on middleware and system integration – *IT Architectures and Middleware: Strategies for Building Large, Integrated Systems (2nd Edition)* – which was published by Addison-Wesley.

His recent white papers include:

- *ClearPath as an Open System*
- *Service-Oriented Architecture: Delivering for Business*
- *Understanding IT system state: Experiences from the ClearPath Appraisal process*
- *Unisys ClearPath Systems Management: Maximising IT Service Availability*
- *Unisys ClearPath Systems: Integrated Hardware/software Stacks*
- *Delivering value: the Economics of ClearPath Systems*
- *Understanding IT cost*