ClearPath Middleware Strategy and Products for OS 2200 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 OS 2200 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 OS 2200 systems are using middleware in various configurations, including service architectures as service providers, consumers, or both. This paper includes products up to ClearPath OS 2200 Release 15, with an indication of some future developments. A companion paper covers the same ground for ClearPath MCP 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 OS 2200 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 ................................................................. 16

Using the middleware: examples ............................................................................................................... 19
  Case study 1 ............................................................................................................................................. 19
  Case study 2 ........................................................................................................................................... 21
  Case study 3 ........................................................................................................................................... 22

More information .......................................................................................................................................... 24

Technical briefs ........................................................................................................................................... 26
  Java ......................................................................................................................................................... 27
  ODTP ....................................................................................................................................................... 29
  Web Transaction Server for OS 2200 (WebTS) ...................................................................................... 33
  Web Services ......................................................................................................................................... 34
  ClearPath ePortal .................................................................................................................................. 39
  Message Queuing Middleware (MQM) ...................................................................................................... 41
  File Transfer Protocol (FTP) .................................................................................................................. 44
  Distributed Transaction Integration (DTI) .................................................................................................. 44
  Data access .............................................................................................................................................. 48
  Sockets .................................................................................................................................................... 50
  Advanced development environments (ADE) .......................................................................................... 52

Appendix: Glossary ....................................................................................................................................... 56

Revision History .......................................................................................................................................... 63

About the author .......................................................................................................................................... 63
Introduction

This paper contains a description of Unisys middleware strategy and products for ClearPath OS 2200 systems – the Dorado family. Middleware is the enabling technology for building distributed systems. Emphasis is therefore given to the role of middleware in integrating ClearPath OS 2200 applications with applications in other environments 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 OS 2200.

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 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.
- *Information Sources* lists some information sources.
- 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]. The pointers are hyperlinked to the brief for the benefit of readers who are reading from the screen rather than from paper.
- 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. Conversely, 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 (and other company) products are renamed. This paper uses names likely to be most familiar to readers but also provides the current names as well. 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 [http://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 Group³.

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 correction⁴.

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.

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.

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 components 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 to:

- 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.

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 OS 2200 platforms, including integration with the native application servers such as TIP, and databases such as DMS 2200 (now called Network Database Server for ClearPath OS 2200) and RDMS (now called Relational Database Server for ClearPath OS 2200). Particular emphasis has been put on comprehensive support for Java application environments. 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 OS 2200 middleware: the products

This section contains a summary of the middleware products available with ClearPath OS 2200 systems. 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 OS 2200 environment. It contains execution environments for transactional and batch applications. They are referred to as application servers in the figure and include the OS 2200 native transaction servers TIP and HVTIP. Applications in these environments and batch programs access the various databases. These environments are typical of OS 2200 customers, implementing many secure and 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 TIP; 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 transaction environments such as TIP/HVTIP ClearPath OS 2200 systems, rather than just running programs directly under OS 2200. 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 Open DTP and TIP/HVTIP may be accessed through resource adapters provided by Unisys, as may the RDMS, DMS and Business Information Server (BIS) databases. The resource adapters for Open DTP, TIP/HVTIP and BIS 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 OS 2200 JProcessor, which is housed within the ClearPath system cabinet. 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 OS 2200. 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, such as TIP, a number of OS 2200 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 the 2200-series systems, the predecessors of ClearPath, and has continued to be enhanced since. [techbrief: ODTP]

Because the Open Group DTP specification contains all the features necessary for DTP systems, service-oriented applications containing ClearPath OS 2200, and other Open Group DTP implementations may be created. For example, ClearPath systems in two Unisys clients communicate with each other using Open DTP, with one system requesting services from the other. Although both ends are in fact ClearPath OS 2200, this is hidden by Open DTP.
Other application environments

Providing a relatively simple web access to ClearPath TIP/HVTIP 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 (WebTS) is an OS 2200 product [techbrief: WebTS], which provides a web server supporting static pages, Java Applets, and so on, and a very efficient Common Gateway Interface (CGI) into applications.

Web Enabler for Display Processing System (formerly called Java Client for DPS), which is a part of the WebTS product, provides an interface into TIP/HVTIP applications that use the Display Processing System (DPS) software. DPS allows applications to be independent of device type, for example allowing an application to use different display formats. Web Enabler allows such applications to run with a web browser without modification, and they continue to work with the terminals.

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, Open Group DTP, TIP and COMS, for example, may be constructed using this middleware.²

Figure 4 illustrates the architecture of loosely-coupled middleware, using message queueing (WebSphere MQ) to connect a ClearPath OS 2200 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

² 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) or secure versions (FTPS and SFTP), 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\(^6\) (Enterprise Application Integration) products support FTP, among other protocols.

Web Services access to OS 2200-based applications and databases may be implemented in a number of ways, depending on the OS 2200 application server and other technology in use. [techbrief: Web Services]

- The ClearPath OS 2200 ePortal product provides access to TIP, DPS, Open DTP and EAE 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 OS 2200 systems can be provided through Java EE application server environments, with resource adapters to connect to TIP/HVTIP 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 IBM WebSphere Application Server.
- Web Services access to TIP/HVTIP 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.)

Two message queuing middleware products are supported [techbrief: MQM]. The first, WebSphere MQ Version 7 for ClearPath OS 2200 (WebSphere MQ), is the follow-on product to MQSeries for ClearPath OS 2200. WebSphere MQ is based on version 7.0 of the IBM product IBM WebSphere MQ Version 7.0 and requires an OS 2200 QProcessor. All the capabilities that are available in MQSeries for ClearPath OS 2200 are applicable.

WebSphere MQ allows TIP/HVTIP or batch programs to use message queuing to participate in message-driven processing. It provides transactional support, with WebSphere MQ acting as an XA-compliant resource manager (RM) under Open Distributed Transaction Processing.

The other MQM product is ClearPath OS 2200 Interface to Microsoft Message Queuing (Interface for MSMQ) is a COBOL-friendly interface to MSMQ message queues. It enables OS 2200 applications on an OS 2200 server 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 OS 2200, Unix and IBM systems.

FTP Services for ClearPath OS 2200 (formerly called Cooperative Processing File Transfer Protocol or cpFTP) is an implementation of the File Transfer Protocol (FTP and FTPS)) for the OS 2200 environment. [techbrief: FTP]

---

\(^6\) 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.
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 Oracle Tuxedo Mainframe Adapter (formerly called eLink OSI-TP), which links Open DTP with Oracle Tuxedo, which can run under a variety of operating systems. 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 OS 2200; environment 2 is the other environment. 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 under</th>
</tr>
</thead>
<tbody>
<tr>
<td>Java Resource Adapters</td>
<td>Open DTP, TIP/HVTIP, batch, databases</td>
<td>Java EE</td>
<td>U/L/W or in JProcessor</td>
</tr>
<tr>
<td>DTI (Distributed Transaction Integration)</td>
<td>Open DTP, TIP/HVTIP, batch</td>
<td>Various</td>
<td>Windows</td>
</tr>
<tr>
<td>Component Enabler</td>
<td>EAE</td>
<td>.NET, Java</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>Open DTP, TIP/HVTIP, ....</td>
<td>Various</td>
<td>Specialty Partition</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 = U, L = Linux, W = Windows.
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 much of the Java environment is running within the ClearPath system. ClearPath databases supported are DMS 2200, RDMS, and BIS. [techbrief: Java]
3) DTI supports a variety of external environments, including COM+/ .NET, the Web, Java and a number of packages. [techbrief: DTI]
4) The Component Enabler connects to EAE on OS 2200; AB Suite is not available. \textit{[techbrief: 4GL]}

5) Oracle Tuxedo Mainframe Adapter 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. \textit{[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. \textit{[techbrief: Web Services]}

7) The ePortal allows all the major smart phones (Android, Blackberry, iPhone and Windows) as well tablets (Android, Blackberry, iPad and Windows). \textit{[techbrief: ePortal]}

\textbf{Other middleware}

ClearPath systems support various other middleware technologies, which may be used in the construction of or in conjunction with, service architectures.

- Direct data access products
- Sockets
- BIS and EAE, the Unisys advanced development environments
- Third party products

\textit{Direct Data Access Products}

Remote systems can directly access databases in OS 2200 systems. 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:

\textbf{RDMS}

- RDMS-JDBC (RDMS-RA) provides both a JDBC API-compliant JDBC driver and a JCA-compliant resource adapter for accessing RDMS. \textit{[techbrief: Java]}
- Open DataBase Connectivity (ODBC) access is provided by UniAccess from Applied Information Sciences (AIS). \textit{[techbrief: Data Access]}

\textbf{DMS}

- The JCA-compliant resource adapter for DMS is DMS-RA. \textit{[techbrief: Java]}
- ODBC access is provided by a product from Data Access. \textit{[techbrief: Data Access]}

\textbf{BIS}

- The JCA-compliant resource adapter is BIS-RA. \textit{[techbrief: Java]}
- BIS also provides extensive facilities for accessing different databases and other data sources. \textit{[techbrief: ADE]}

\textbf{CIFS}

- CIFS (Common Internet File System) provides direct, secure file access both to and from the OS 2200 using the industry-standard SMB protocol. \textit{[techbrief: Data Access]}

\textit{Sockets}

Low-level middleware, in particular sockets, provides a means of interconnecting applications in a distributed environment, although 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.
COMAPI (Communications Application Programming Interface) provides a sockets interface into OS 2200 programs.

CITA (Communications Interface for Transaction Applications) is a sockets interface specifically for TIP/HVTIP applications.

NTSI (Message Integration Services for ClearPath OS 2200), which is an additional API over COMAPI, provides a simple interface for communication between OS 2200 and Windows programs. [techbrief: Sockets]

Fourth Generation Languages – BIS and EAE

Business Information Server (BIS –formerly MAPPER) and Enterprise Application Environment (formerly LINC) are fourth generation languages for developing applications which can be deployed under OS 2200 as well as other operating systems. Although they are not middleware, they do offer some middleware features or features enabling easy integration with other middleware standards. [techbrief: ADE]

On OS 2200 systems, EAE generates applications to run in an HVTIP environment. By using a different option, it can also generate Open DTP services. In addition to these environments, EAE contains some middleware features of its own, supporting integration with web environments and distributed computing. EAE systems can co-operate in a transaction, including two-phase commit of database updates.

BIS applications can also be generated as Open DTP services as well as in a TIP environment. In addition, BIS also contains a distributed system infrastructure, known as BIS networking. This allows a BIS application to make a remote procedure call to another system within the BIS environment.

While BIS contains its own relational database structure, it has native access to other relational databases, including RDMS in the OS 2200 environment, as well as Oracle and some other well-known databases.

Third Party Products

There are various third party products on the market for use with OS 2200 systems. As far as middleware is concerned, the most comprehensive is the UniAccess suite from AIS, implementing Microsoft client/server architecture.

Middleware and ClearPath systems with fabric infrastructure

In 2014, Unisys began delivery of ClearPath Dorado systems, implemented in a fabric-based architecture. As well as providing OS 2200 environments, together with Specialty Partitions, the systems support running 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 a Dorado system. The systems are packaged in two 42U cabinets. The left-hand cabinet contains Unisys Intel platforms implementing the ClearPath OS 2200 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 Dorado 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, apart from the QProcessor and JProcessor, which run in EPPs. 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.

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 OS 2200.
- 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 OS
2200 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 a higher degree of integration with the OS 2200 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 OS 2200 environment.
Using the middleware: examples

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 OS 2200 systems.

Case study 1

One common approach that has emerged across the ClearPath user base is to use Open DTP. In many cases, applications written using TIP/HVTIP can easily be adapted to expose some or all transactions as Open DTP services. Tools are provided with the Open DTP product to help in the adaptation. New services can also be created to run alongside the existing application, working on the same database or databases. And, if the applications were written using EAE or BIS, or their predecessors, selected functions can optionally be generated as Open DTP services.

There are several advantages in using the Open DTP approach.

- Open DTP services can be accessed by any other implementation of the same standard, in ClearPath or elsewhere. It is very easy to create an Open DTP client, and Open DTP services can be clients to other services.
- There are off-the-shelf connectors to Open DTP from Tuxedo, Java via a JCA-compliant connector, Microsoft .NET and the ePortal. Distributed transactions can be implemented across these technologies, with consistency maintained across databases in the different systems.
- Many of the connectors are bidirectional: the ClearPath end can initiate the transaction in addition to being a service in a transaction initiated elsewhere in the service environment.
- The process of creating services by adaptation forces a clean-up of the application, rationalising the code.

The Unisys Logistics Management System (LMS) is an example of using Open DTP in an SOA implementation. LMS provides air cargo services to a number of airlines, and is implemented as Software-as-a-Service, provided by Unisys. The whole environment comprises a number of components. Those of interest in the context of this paper are described below.

- Front end servers running Windows provide access channel management, service definition and mapping, and the service bus components of the architecture shown in figure 1. Users connect via IIS, while an application offers external services to users. The application is implemented using ASPs.
- The front end application uses an SQL database to hold configuration, security and other data. The data can be held in any language the client requests. The database is held in a Windows cluster.
- The cargo services and database are implemented in ClearPath Dorado systems. There are two systems: one manages airlines in the US time zones while the other handles the rest of the world. The Unisys USAS cargo application runs under HVTIP, exposing its services as Open DTP services. The DTI product in the front end servers is used as the connector to the ClearPath application services. The application also initiates outbound requests to services in the front end systems.
Figure 7 is a schematic of the architecture of the front end and ClearPath components, showing the connections from the front end to the database, and between the front end and the ClearPath systems for the non-US time zones. (The US time zones system follows the same flow.)

The architecture shown in figure 7 provides considerable flexibility. New services can easily be added, either within the ClearPath systems or elsewhere, even in other organisations; Windows provides a variety of connection options. The technologies used can be changed while retaining the same architectural pattern. For example, a Java implementation of the front-end systems could be used, with the Java DTP connector to access the Open DTP applications. The ePortal could also be used to access Open DTP services, using Web Services as the connection technology between the front end and the application services.

Although this paper is concerned with service architecture and how ClearPath systems can use middleware and participate in SOA implementation, it is worth noting that the LMS environment offers an exceptional level of stability and performance. One of the hosts, for instance, ran for over 1,500 days without a single restart; it was only finally stopped as part of a significant hardware upgrade. During the extended period of non-stop operation, in excess of three billion transactions were processed.
Case study 2

This case study is another example of using Open DTP in the ClearPath OS 2200 environment. The organisation is a utility company providing a service to more than 26 million people over a wide area. The company uses OS 2200 systems and a variety of other Unisys systems, including ES3000s and ES7000s, as well as other platforms. The OS 2200 application environment is EAE. The biggest application is billing, which has over 2,000 Ispecs. Around ten other applications have more than 3,000 Ispecs between them.

A number of factors drove the implementation described here. They include:

- Exposing enterprise transactions (Ispecs) to Web-Services accessible from other Java application servers and .NET components.
- Improving the EAE user interface by providing printing of dynamic PDF documents generated by Ispecs.
- Integrating auditing capabilities.
- Generating dynamic reports from RDMS to the Web.
- Providing high performance and reliability.

The resulting implementation is a service-based architecture, exposing the Ispecs as Open DTP services and making extensive using of Java-based open source middleware products. Figure 8 shows the architecture.

Figure 8: Schematic of service architecture showing key open source products used
The Transport Layer, shown in the upper part of the figure, provides the connections to the various clients. The service bus is the Mule ESB, from MuleSoft. Tomcat provides the external connections to devices such as smart phones and browsers. Other connections are provided by XML/SOAP (Web Services) and TCP/IP (Sockets) connections.

The Transaction Layer, shown in the lower part of the figure, implements the services, which are externally exposed by EJBs in the JBoss application server. The internal providers invoked by the EJBs to deliver the services are Ispecs, generated as Open DTP services in the OS 2200 environment, and RDMS and other databases, accessed using JDBC.

The Transport and Transaction Layers are connected by MQM, using the JBoss HornetQ product.

**Case study 3**

The final example is a bank, which uses WebSphere MQ for fund transfers with other banks and financial institutions. The service is regulated by the financial regulator of the country concerned and operates 24 hours per day, 7 days per week. Several types of transaction are supported, including single, immediate payments, forward-dated payments, which are single payments with a specified date, and standing orders for regular fixed amounts. Transfers can be made in both directions, that is, into and out of the bank. Additional services are expected to be added.

A Dorado system holds the core banking data. Figure 9 is a high-level view of the architecture used in the transfer service.

Figure 9: Schematic of architecture for WebSphere MQ fund transfer

WebSphere MQ Version 7 for ClearPath OS 2200 (WebSphere MQ) is used. It executes in a QProcessor specialty partition environment, with the Queue Manager distributed across a single QProcessor High Availability (HA) cluster. Outbound requests – transfers to another institution – use persistent queues. Inbound requests are non-persistent and are tracked by another monitoring system. Outbound requests are triggered via batch process (e.g. for standing orders) or requests via branch, telephone or Internet banking. Inbound requests may be bundled and run as a batch process.
Resilience is at the Queue Manager level. The configuration is active/passive: if a QProcessor fails, the Queue Manager moves to the other QProcessor in the cluster. Messages are resumed once the Queue Manager has moved; it takes less than a minute to move it to the standby QProcessor.

In the event of the loss of the disk containing the queues, or the loss of the live environment for some other reason, service resumes on a remote site. Disk resilience is provided by using EMC SRDF to the remote site, where a QProcessor cluster is left in a ready/wait state.
**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:


IEEE: [http://www.ieee.org](http://www.ieee.org)


The International Organization for Standardization (ISO): [http://www.iso.org](http://www.iso.org)

The International Telecommunications Union (ITU – formerly CCITT): [http://www.itu.int](http://www.itu.int)


The Open Group: [http://www.opengroup.org](http://www.opengroup.org)

UDDI.xml.org (for information about Web Services): [http://uddi.xml.org](http://uddi.xml.org)

World Wide Web Consortium (W3C): [http://www.w3.org](http://www.w3.org)

****

*Organisations providing products and services*

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


*IBM*


*Microsoft*

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


**Oracle**


**Red Hat**

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 ‘TIP/HVTIP program’, ‘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.

The technical briefs are:

**Java:** explains the basic principles of Java, including JVMs and applications servers, and how they are implemented with ClearPath OS 2200 systems.

**ODTP:** explains the Open Group DTP model and its implementation in ClearPath OS 2200 systems.

**WebTS:** describes the WebTS Web server product.

**Web Services:** explains Web Services technology and how ClearPath OS 2200 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 OS 2200 systems, in particular WebSphere MQ (MQSeries).

**FileTransfer Protocol (FTP):** describes the implementation of FTP under OS 2200.

**Distributed Transaction Integration (DTI):** describes the structure of Distributed Transaction Integration and how it can be used to connect a variety of technologies to different OS 2200 application environments.

**Data access:** discusses the ideas of direct data access and some of the products available with ClearPath OS 2200 systems.

**Sockets:** explains the principles of sockets and the various implementations in ClearPath OS 2200 systems.

**Advanced Development Environments (ADE):** describes the middleware-related features available with EAE and BIS.
Java

The basic Java platform from Oracle (Sun) 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 TIP/HVTIP 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 JavaBeans (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
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.

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 OS 2200 resources. Table Java-1 shows the resource adapters and the environments in which they can run. The resource adapters for Open DTP,
TIP/HVTIP and BIS 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>Relational JDBC (RDMS)</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Network Database Server (DMS)</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>TIP</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Open DTP</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>BIS</td>
<td>Yes</td>
<td>Yes</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\(^8\) 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.

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.

---

\(^8\) The model was initially defined by X/Open, which subsequently merged with the Open Software Foundation (OSF) to form the Open Group.
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\(^9\) properties of the transaction.

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

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

\[^9\] 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.
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 OS 2200 UDS databases – DMS, RDMS, FCSS 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 systems10. 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 of the model under OS 2200 is the full form and is called Open Distributed Transaction Processing (Open DTP). Figure ODTP-3 shows the OS 2200 environment. TM2200 implements the TM and CRM functions. OSI-TP is used to communicate with other systems implementing the model, including of course other ClearPath systems. The diagram shows an Open DTP client implementation accessing TM2200. (This is done via TCP listener running under OS 2200; it is not shown in the diagram.) The client invokes services using XATMI functions. The XATMI requests are passed to TM2200, which then finds the service and invokes it. This could mean connecting to another system if that is where the service is located.

If the user equipment is not Open DTP-compliant – it does not implement the client or full model – an intermediate program can function as a client. Figure ODTP-3 shows some possibilities for TIP/HVTIP application programs. Terminals such as UTS or other noncompliant devices may be in use with a TIP/HVTIP application. If they want to access Open DTP services, a TIP program can be written to function as a client; the terminal talks to the program, which then uses TX/XATMI to communicate with Open DTP services.

---

10 OSI-TP (ISO/IEC 10026) is mandated by the Open Group for interoperability between heterogeneous DTP systems.
TIP/HVTIP and batch application programs are not Open DTP-compliant. There are various options for them to use Open DTP. They can easily become an Open DTP client by using the TX and XATMI APIs; figure ODTP-3 shows this connection. The TIP/HVTIP program can call local or remote Open DTP services.

To be used as Open DTP services, non-compliant application programs need to be adapted to comply or wrapped (encapsulated) in some way. The complexity of this task depends on the way the programs have been written; some are easier than others. HVTIP applications can become Open DTP services by changing the ICP (Initial Control Program). For EAE applications, Individual ISPECS can be generated as Open DTP services.

To help in encapsulating non-compliant application programs, Unisys provides an Open DTP component called Heritage Application Access (HAA). HAA allows existing application programs that are not compliant with Open DTP and are written without using the TX and XATMI services to be incorporated as named services within an Open DTP environment. HAA provides an interface between the two. Note that HAA only allows access to unmodified application programs as services; they cannot be clients because a client needs to use TX and XATMI.

---

**Figure ODTP-3: The Open Group DTP model as implemented under OS 2200**

OS 2200 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 other techbriefs in this paper. [techbrief: Java] [techbrief: DTI]
Web Transaction Server for OS 2200 (WebTS)

Web Transaction Server for OS 2200 (WebTS) runs entirely within an OS 2200 environment; it does not depend on external software. It provides a high volume, secure web server, together with several other facilities. Figure WebTS-1 shows the major elements of the WebTS environment.

Web browsers use HTTP/S to connect into WebTS, which then loads Java Applets into the browser to provide the user interface. The interface can be simply a “screen” within a browser environment, or it can be enhanced to provide the kind of graphical view expected by web browser users. WebTS also supports static web pages, forms, graphics, Java Applets, and other standard web server features to facilitate implementation of complete web applications. It does not require frontend gateway systems.

WebTS is tightly integrated with the TIP/HVTIP, supporting application programs written in UCS (Universal Compiler System) C, UCS COBOL, and ASCII COBOL. There are two main ways to access TIP/HVTIP application programs:

- Java Client for DPS (Web Enabler for DPS for OS 2200)
- Using the standard Common Gateway Interface (CGI)

If the application uses DPS, Java Client for DPS provides a way of accessing the application programs without any changes. Further, the programs continue to work with any existing terminal or terminal emulation interfaces. The web interface adds to the options; it does not replace terminals unless the user chooses to remove them.

The CGI interface, which can be used if DPS is not used, is highly optimised for performance. The application programs are just standard TIP/HVTIP transactions with interfaces compatible with CGI web format rather than, for example, UTS. WebTS also provides an interface into Open DTP (Open DTP Gateway in Figure WebTS-1).

WebTS supports a number of features, or sets of features, of particular interest in TIP/HVTIP and OS 2200 environments: security, state management, and file naming.
Security

WebTS provides strong security features. The optional Secure Sockets Layer (SSL) 3.0 and Transport Layer Security (TLS) protocols for securing web communications are implemented. SSL/TLS provides for full encryption of all messages flowing between the browser and the server. SSL/TLS also supports a mechanism called certificates for identifying or authenticating clients and servers. When SSL/TLS is enabled, identification is done automatically and transparently at the beginning of each secured session.

The Web Transaction Server standard version (which is part of the integrated operating environment) provides 128-bit and even stronger levels of data encryption. Furthermore, communication between WebTS and its administration program, WebTSA, is also encrypted.

WebTS provides support of TIP Session Control file and subsystem access protection, and is protected by a web-based userid/password sign-on. The Static PID feature allows use of Fixed PIDs (that is, the same device always has the same PID) for applications where they are required to support application-level security.

State Management

The Persistent PID features, which can be used with Java Client or CGI transactions, cause WebTS to use the same PID when calling a sequence of transactions from a given browser. This allows existing transactions that save client state using the PID (all DPS transactions, for example) to continue to do so in the web environment. This also applies to Static (Fixed) PIDs.

File Naming

WebTS supports the Windows and Unix directory and file naming conventions, in addition to the OS 2200 file and element naming convention. This enables web developers to publish their web content on a website hosted by WebTS without converting their files and directories to OS 2200 naming conventions.

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 follows\(^{11}\).

'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 WSDL)\(^{12}\). Other

---

\(^{11}\) WSA can be found at [http://www.w3.org/TR/2004/NOTE-ws-arch-20040211/](http://www.w3.org/TR/2004/NOTE-ws-arch-20040211/). The definition quoted is in section 1.4.

\(^{12}\) Not all Web Services have a WSDL description. Simple RESTful services may do away with it altogether.
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.

REST-compliant Web Services are more constrained and simpler\(^\text{13}\). 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 OS 2200 systems, TIP/HVTIP and Open DTP applications may receive incoming Web Services requests and initiate outgoing requests, using either Java or .NET as the Web Services implementation. DMS, RDMS and BIS 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 another techbrief. [techbrief: ePortal])

Figure Web Services-2 shows the general architecture. The OS 2200 environment is at the bottom of the figure, containing the TIP/HVTIP and Open DTP application environments, and the DMS, RDMS and BIS databases.

---

13 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 OS 2200 system, or a Web Services provider if the OS 2200 applications are consumers.

The box labelled 'Web Services Enabling Software' in between bridges the gap. It contains three elements, which enable the OS 2200 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 OS 2200, 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 ClearPath TIP/HVTIP and Open DTP application programs as both providers and consumers of Web Services, as well as exposing DMS, RDMS, and BIS databases as providers. Because Java EE application servers can run outside the OS 2200 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 OS 2200 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 OS 2200. 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 on OS 2200 (or OS 2200 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 OS 2200 applications and databases. Web Services engines are available as products with Java EE application servers.

![Diagram showing two possible configurations using Java EE as the Web Services Interface for OS 2200 Applications and Databases](figure_web_services-3)

Figure Web Services-4 expands on the right hand side of figure WebServices-3, showing a possible Web Services implementation using a TIP application environment with JBoss EAP running in the JProcessor providing the Web Services interface. The TIP 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 or a POJO wrapped as a Web Service. The EJB or POJO performs the required data mapping and invokes the JCA-compliant TIP Resource Adapter (TIP RA), which connects to the TIP environment through COMAPI and CITA. [techbrief: Sockets]
For outgoing Web Service requests, as shown on the right of the figure, the TIP transactions invoke the resource adapter via COMAPI 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 OS 2200 platform. The resource adapter would then connect to COMAPI across a network, using CPComm. The off-platform architecture could also be implemented using another Java EE application server, for example WebSphere AS, using a suitable Web Services engine.

Although the above example assumes a TIP application, a similar approach could be followed for Open DTP applications, using the appropriate resource adapter.

![Diagram showing TIP transactions functioning as Web Services Providers (left) and Consumers (right) using JBoss EAP](image)

### .NET Implementations

Implementations using .NET support TIP/HVTIP 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.

The integration component is a .NET component, with DTI providing the integration middleware. Depending on the OS 2200 application environment, different DTI features would be used. [techbrief: DTI]

- Open DTP would use 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.
- TIP/HVTIP would use the .NET adapter, the XGate gateway, and the TIP/HVTIP connector for incoming requests.
- TIP/HVTIP would use the OS 2200 outbound feature for outgoing requests.

![Diagram](image)

**Figure Web Services-5: Configuration using .Net to provide Web Services access**

Other approaches are possible for implementing Web Service access to OS 2200 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 OS 2200 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 OS 2200 system. Examples in the ClearPath OS 2200 base of this approach include using WebSphere MQ to connect to TIP/HVTIP applications and using the Open DTP RA in an ESB.

**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. It contains three major components: developer, manager and run-time environment – see figure ePortal-1.
ePortal connects to Web browsers, and the smart phones and tablets shown in the figure to applications in the OS 2200 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 application environments and connections available.

<table>
<thead>
<tr>
<th>Data source</th>
<th>Type of input</th>
</tr>
</thead>
<tbody>
<tr>
<td>Open DTP</td>
<td>Open DTP view files or COBOL data record layout</td>
</tr>
<tr>
<td>TIP/HVTIP</td>
<td>COBOL data record layout or DPS FMLU generated record layout for DPS applications</td>
</tr>
<tr>
<td>EAE</td>
<td>Ispec definitions from EAE applications</td>
</tr>
<tr>
<td>UTS</td>
<td>Screen scraping for applications expecting UTS terminal screens</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, there may be problems in maintaining database integrity. Maintaining database integrity requires that all programs involved participate in a single, global transaction as defined by the Open Group. Using MQM may require two global transactions: one updates a database and puts a message on a queue; the other takes the message off the queue and updates a database.

This techbrief describes two MQM products available with ClearPath OS 2200 systems: WebSphere MQ Version 7 for ClearPath OS 2200 (WebSphere MQ) and ClearPath OS 2200 Interface to Microsoft Message Queuing (Interface for MSMQ), an interface to the Microsoft Message Queuing (MSMQ).

**WebSphere MQ for ClearPath OS 2200**

IBM’s WebSphere MQ product is important because it is feature-rich and available on all platforms of any significance. The Unisys implementation is WebSphere MQ Version 7 for ClearPath OS 2200 (WebSphere MQ), which executes in the OS 2200 QProcessor. OS 2200 applications – TIP/HVTIP, Open DTP and batch – may use the product to communicate with other WebSphere MQ implementations, using the MQI calls to interface to the queue manager. Messages and queues may be recoverable if the appropriate options are chosen. Figure MQM-1 is a schematic of the environment.

WebSphere MQ supports the ability to include message queues as well as databases within the context of transaction boundaries. When a transaction completes, it needs to decide whether or not database updates may be made. If the transaction execution is successful, it commits all the updates to be made. If there are any problems, the databases are not updated (or are rolled back) and left in the state they were in at the start of the transaction. This preserves database integrity and the ACID properties of the transaction. (See [techbrief: ODTP] for more details about transactional integrity.)
Message queues may 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. Database commits within the transaction are independent and not linked to the message queues. If transactional integrity is required, where both messages and databases are to be included within a transaction, Open DTP (TM2200) is used. Message queues have been incorporated within TM2200 as XA-compliant resources, as shown in Figure MQM-2.

Figure MQM-1: Schematic of WebSphere MQ environment, showing the OS 2200 QProcessor

Figure MQM-2: WebSphere MQ with transactional integrity between database updates and message queues
The OS 2200 application program runs under Open DTP. It uses the XATMI interface to communicate with other services (not shown), which may update UDS databases. The application program may also use the MQI for message queuing services. At the completion of a transaction, a commit is issued if everything is in order. In that case, UDS and the WebSphere MQ resource manager both use the XA interface for coordination of a two-phase commit process. The databases are updated and messages are put on or taken off queues. Messages intended for other systems are then available for processing; they are not seen before the commit. If the commit cannot be issued, the databases and queues are left in the state they were in at the start of the transaction.

There are many options for queues and messages, to manage recoverability and initiation of recipient applications. The features supported include triggers, which allow the recipient of a message to be initialised as soon as a message is put onto a queue for it. Other features include queuing messages for multiple recipients, and destructive and non-destructive message retrievals. The first allows messages to be read and left on the queue, and the second removes the message from the queue when read.

**MSMQI – ClearPath OS 2200 Interface to Microsoft Message Queuing**

MSMQI is an interface to Microsoft Message Queuing (MSMQ) for ClearPath OS 2200 systems. Figure MQM-3 illustrates the architecture.

As shown in Figure 28, there are two components: the MSMQ Interface Agent, which runs under OS 2200, and the MSMQ Interface Adapter, which runs under Windows and acts as a client to the MSMQ Server, to which it connects through the MSMQ COM Interface. The MSMQ Server may be in the same server or remote. The configuration may be set up as a Dependent Client, where the MSMQ Server must be connected. It may also be set up as an Independent Client, where the messages are staged.

![Figure MQM-3: Schematic of the MSMQI architecture](image-url)
File Transfer Protocol (FTP)

File transfer is not usually regarded as middleware but is widely used for communication between different parts of an organisation or between separate organisations for exchanging business data. Much of the business conducted using Electronic Data Interchange (EDI) uses FTP, and the ESB products typically include FTP as one of the supported interfaces. It has the advantages of simplicity, universal availability and the capability to carry large amounts of data. Although there are alternative technologies for electronic business, FTP is likely to persist.

FTP Services for ClearPath OS 2200 (formerly called Co-operative Processing File Transfer Protocol or cpFTP) is a high-performance FTP application. Figure FTP-1 is a schematic of the architecture.

![Figure FTP-1: Schematic of the FTP Services for ClearPath OS 2200 architecture](image)

FTP Services for ClearPath OS 2200 supports both encrypted and unencrypted data transfer.

- The CPFTP process in CPComm or CPCommOS can be configured to use FTPS. If the remote FTP client uses FTPS, cpFTP will too.
- For the @CPFTP client, the ‘S’ (secure) or ‘I’ (implicit security) option makes it secure.
- APIs support secure FTP from UC, UCS COBOL and UCS Fortran programs.

Distributed Transaction Integration (DTI)

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.

DTI connects a variety of external clients to TIP/HVTIP, Open DTP and batch applications running under OS 2200. DTI can also be used to connect to ClearPath MCP applications and, via another environment integrator product, Oracle Tuxedo Mainframe Adapter OSI-TP [techbrief: ODTP], to Tuxedo.

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

In principle, any client technology can be mapped onto any application type, although there are some invalid combinations depending on the required functionality. For example, COM+ or .NET to OS 2200 must use Open DTP if two-phase commit is required. Once DTI is configured to access an OS 2200 application, all client types supported can access it.
Clients initiate requests to access the OS 2200 applications as services through the DTI. Outgoing requests, where the OS 2200 application initiates a connection, are possible in many cases.

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 between DTI and the various clients is designed to make it simple to add new client types. The run-time architecture is based on three architectural entities: adapters, gateways, and connectors. (In fact, this is the architecture of the original Transaction Integrator.)

Figure DTI-2 shows how the run-time architectural elements fit together.
The client types can be divided into two groups, as shown in Figure DTI-2:

- 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

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 OS 2200 application and vice versa. The vertical bars represent a software bus; anything on the right can use anything on the left, depending on the requirements. Similarly, the component gateways use one of the connectors to access the OS 2200 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 OS 2200 application and, as was the case for component gateways, uses a connector to access it.

Figure DTI-3 shows the run-time architecture of DTI in more detail, identifying the specific component parts. At the top is OpenTI, which provides connections between COM+/.NET and implementations of the Open Group DTP model. At the bottom is the OS 2200 outbound capability. The remainder is the Transaction Integrator run-time environment, providing inbound requests only; the OS 2200 end can only reply, not initiate outbound requests.

OpenTI provides a run-time environment of bidirectional access between Microsoft COM+/.NET and implementations of Open Group DTP. The connection is symmetrical; either end can initiate a request. OpenTI wraps Open Group 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 Group DTP services to the Open Group DTP end. By using the Oracle Tuxedo Mainframe Adapter OSI-TP environment integrator, the interworking extends to Tuxedo.
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 Group DTP views.

Turning now to the Transaction Integrator part of DTI, the adapters are in the upper right part of figure DTI-3, 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 generalised gateway, handles data as name/value pairs. The other two gateways are XML gateways. DTI supports XML handling as shown in Figure DTI-4.

![Figure DTI-4: XML handling by DTI](image)

As shown in the figure, applications in OS 2200 and external environments may or may not be XML-aware. All four combinations of data handling are possible:

- XML may be passed between XML-aware applications in OS 2200 and external environments. This is shown by the dotted red line between the two XML-aware applications in the figure.
- An external XML-aware application may communicate with an OS 2200 application that is not XML-aware. DTI performs the mapping between XML and COBOL data structures or Open DTP views.
• An external application that is not XML-aware may communicate with an XML-aware OS 2200 application. Again, DTI performs the required conversions.
• Finally, applications that are not XML-aware may communicate with each other, as shown by the solid green line.

XGate, which is a generalised XML gateway, provides automatic conversion between COBOL data structures or Open DTP views to or from XML. Both XGate and GGate can access TIP/HVTIP, Open DTP and batch application programs.

CXGate is an enhanced version of XGate for Open DTP only, handling additional data types not just ASCII.

GGate and XGate can use three connectors to access the OS 2200 applications, as shown in Figure DTI-4.

• The TIP/HVTIP connector uses COMAPI. [techbrief: Sockets]
• The RWQueue connector provides access to MSMQ. Inbound requests and outbound replies are passed via an MSMQ queue. [techbrief: MQM]
• The Open DTP connector is used to access Open DTP. The connection is in fact 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 is a relatively simple interface, providing access for client input data in HTML. It maps the input to and from the required form, depending on the target application type.

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.

The remaining piece of DTI shown in figure DTI-4 is the OS 2200 outbound facility. This provides a listener in DTI, allowing programs in any of the application types to initiate requests as clients to DCOM applications. The connection is not transactional; that is, two-phase commit is not supported. The OS 2200 applications use a standard Open DTP API to call the components, although the applications do not have to be Open DTP services. An interface to Java EJB remains but, as noted previously, the Java resource adapters are the recommended way to connect to Java.

DTI is a powerful product, supporting as it does interconnectivity with a wide variety of external client applications. This includes connecting Web Services to OS 2200 applications as a service provider, and also as a client for other providers. [techbrief: Web Services]

Data access

Data contained in OS 2200-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 techbrief describes the ODBC options for accessing RDMS and DMS, and CIFS.

**ODBC**

The products concerned are UniAccess for RDMS from Applied Information Sciences (AIS27) and Data Access for DMS. Figure Data Access-1 illustrates the architecture. (Note that both DMS and RDMS can also be accessed using Java resource adapters.) See [techbrief: Java]

UniAccess provides SQL access from Windows clients to RDMS data. With UniAccess, RDMS data can be accessed with the same client tools and applications that are used to access data in SQL databases running on other platforms, for example Oracle.

It enables any ODBC-compliant Windows-based programs such as Microsoft Excel and Access, or any client using the Sybase Open Client or MS DBLib, to access RDMS databases and repository information. UniAccess does not require any change to an existing schema. It operates against the data currently in the database; there is no duplication.

Data Access provides SQL access from clients to non-relational databases, including DMS 2200. It permits the use of Windows-based tools and applications to access DMS data using widely available tools and techniques. An ODBC driver interface and Interactive SQL access are supported.

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

Data 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 provide full update capability. It allows user-level limits to be set on queries, allowing different limits to be set for each user.
CIFS

The Common Internet File System (CIFS) for ClearPath OS 2200 enables access to OS 2200 files through standard internet protocols. CIFS also includes a client capability, enabling OS 2200 programs to access files on other systems that support the SMB (Server Message Block) protocol, such as all Windows platforms and Linux systems running a Samba server.

OS 2200 files can be manipulated from a workstation using the standard hierarchical directory structure in Windows Explorer. CIFS provides access to all OS 2200 files and elements through a special directory called /os2200. During CIFS initialization, all OS 2200 files (except certain system files) are automatically added to the /os2200 directory. CIFS monitors changes to the OS 2200 file environment and updates the /os2200 hierarchy accordingly.

This enables users to do the following:

- Move, copy, delete, and display properties of the OS 2200 files.
- Edit files with Windows Notepad, Microsoft Word, image processors and so on, and save changes to the files.
- Copy or move files, using the Windows drag-and-drop feature, between a user workstation and OS 2200.
- Create, update, view and extract zip files on the OS 2200 system.

CIFS for ClearPath OS 2200 is very secure. It strictly enforces every aspect of the OS 2200 security policy. CIFS for ClearPath OS 2200 incorporates programming practices that limit exposure to application security vulnerabilities. This reduces undetected buffer overflows and provides better handling of unanticipated requests.

CIFS for ClearPath OS 2200 includes the following components:

- CIFS Host Infrastructure (CIFSHI) This component includes the communications layers necessary for a remote client to use CIFS on the OS 2200, as well as the file system structure and host APIs. The communication capabilities of CIFSHI uses SMB protocols to provide access to existing OS 2200 files from most SMB clients, such as Windows Explorer.
- CIFS Library (CIFSLIB) This component consists of a UC- and UCOB-callable layer of interfaces to CIFSHI as well as I/O stream-handling functions.
- CIFS Utility (CIFSUT) This component provides a simple command-line interface to CIFS capabilities.
- ZIPUT Processor This command line utility is used to create, update, view and extract zip files on OS 2200 systems. CIFS interfaces with the Exec are invisible. CIFS utilities and application program interfaces are accessed by calling them as functions from a C program.

Sockets

Although sockets technology may not be regarded as middleware, it is widely used and is a platform on which other technologies build. This brief explains three products: COMAPI, CITA, and NTSI. All use the OS 2200 communications infrastructure, which includes the following components:

- CPComm (Communications Platform)
- The TSAM (Transport Service Access Method) subsystem
- The associated APIs

TSAMAPI provides access to both TSAM and CPComm; and CPCOMMAPI provides a very efficient access to CPComm.
COMAPI (Communications Application Program Interface)

COMAPI, which sits on top of the communications infrastructure, provides a sockets interface between any OS 2200 application program and some other program. See Figure Sockets-1.

The program must be written to use the API to send and receive messages. This is straightforward because the API is very well known. For example, C-language programs using a sockets interface on UNIX platforms have been recompiled for OS 2200 and have worked without change. COMAPI is also used as an infrastructure element in a number of other products within ClearPath OS 2200 middleware; NTSI and WebSphere MQ for ClearPath OS 2200 are examples.

An additional feature (not shown in figure Sockets-1) was developed to allow COMAPI to schedule TIP/HVTIP transactions via the Message Control Bank (MCB), although the programs must be written or adapted to use COMAPI directly for outputs. A better alternative subsequently developed – CITA – is discussed in the next section.

CITA (Communications Interface for Transaction Applications)

CITA replaced TCIS (TSAM Communications Interface Software), which was available as a special product (a reorderable custom engineering request) as a sockets connection specifically for TIP applications. CITA interfaces to COMAPI and interprets a byte stream input as having a header and an associated message. The byte stream is separated into input messages and passed to the TIP/HVTIP application programs via MCB.

CITA provides a number of options for use. Input and output can flow in both directions through all the various components, as shown in Figure Sockets-1. Another option for output, also shown in the figure, is that the application programs can directly interface to COMAPI, bypassing MCB and CITA. The result is a reduction in path length, although the application programs will lose the recovery capabilities provided by MCB. However, this may not matter, for example if a transaction simply displays information, making no database changes.
**NTSI (Message Integration Services for ClearPath OS 2200)**

NTSI was designed to communicate with Windows application programs running originally in an Intel® node within a ClearPath system. The purpose was to allow OS 2200 applications to invoke services in a Windows environment, for example mail applications. NTSI uses COMAPI to connect to a NTSI component in the Windows environment; an adapter then connects to the application programs. Figure Sockets-2 illustrates the NTSI architecture. NTSI also provides some of the infrastructure to support Kerberos and other secure logon technologies.

![Figure Sockets-2: NTSI architecture](image)

**Advanced development environments (ADE)**

Enterprise Application Environment (EAE) and Business Information Server (BIS) are advanced development environments, which run under OS 2200 as well as a number of other environments including MCP and Windows. In addition to enabling rapid application development, EAE and BIS contain a number of infrastructure features that can be considered as middleware.

**EAE**

When running under OS 2200, EAE has two ways to generate the same application, which can coexist. It can generate HVTIP applications, which use RDMS as the database infrastructure, and it can generate Open DTP services, where each individual EAE Ispec (transaction) is generated as a service. With the Open DTP option, 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 Java resource adapters with other systems, such as Tuxedo, .NET, and Java EE. [*techbrief: ODTP*] [*techbrief: Java*] [*techbrief: DTI*]

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. It 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.

Client options for Component Enabler include:

- **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 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 that may reside on the same host or on a different host or even a different host type. It includes a two-phase commit capability to preserve the integrity of global transactions.
When running in an OS 2200 environment, BIS applications may be invoked as Open DTP services and may therefore be called from any other compatible system or through an environment integrator. However, BIS applications cannot function as clients in OS 2200 environments. BIS applications can also be accessed by a Java resource adapter. \[\text{techbrief: Java}\]

BIS also includes some middleware functions within its infrastructure. There are no specific Web interfaces that are unique to BIS; however, the Internet Commerce Enabler is particularly simple to connect to BIS applications, as well as many other applications and databases. Peer-to-peer networking among BIS systems is supported by the BIS Networking function. The systems invoke each other using a remote procedure call carried over a TCP/IP network.

BIS also supports interfaces to various relational databases using the BIS Relational Interface (formerly MRI). Figure ADE-2 illustrates the architecture of the Relational Interface and BIS Networking in an OS 2200 system. BIS can also use the WebSphere MQ MQI as Unisys has provided an interface for BIS through an optionally-installable basic-mode subsystem.

In the figure, workstations can connect into a BIS environment running under OS 2200. The application can invoke BIS in other systems. These applications can use the Relational Interface to access the major commercial database products, such as Oracle. The Relational Interface can be used to access RDMS in native mode and through ODBC with UniAccess (not shown in the figure). It can also access the non-relational DMS through ODBC and Data Access.

Similarly, workstations can be connected into the BIS environment running under Windows and other operating systems. The applications can invoke the services of BIS in the OS 2200 environment and of course use the Relational Interface to access databases.
Figure ADE-2: BIS network and the Relational Interface
Appendix: Glossary

This paper has used a number of technical terms, acronyms and product names.

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

Some examples of name changes are Unisys MAPPER to Business Information Server and RDMS to Enterprise Relational Database Server for ClearPath OS 2200. 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. A Unisys advanced development environment product, it 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.

**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 and the Red Hat JBoss Enterprise Application Platform are widely-used products.

**ASP**: Active Server Page.

**BIS**: Business Information Server. A Unisys advanced development environment product. It is the latest in the evolution of a product originally called MAPPER.

**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.

**CIFS**: Common Internet File System: see SMB.

**CITA**: Communications Interface for Transaction Applications (CITA). Enables remote devices to access OS 2200 TIP applications.

**COMAPI**: Communications Application Program Interface. Enables applications to access a communications network.
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.

Component Enabler: A Unisys environment integrator product connecting Microsoft .NET and Java environments into EAE and AB Suite applications. Full name is Enterprise Application Component Enabler

A standard, defined by the Object Management Group (OMG), for building distributed systems using object principles.

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

CPComm: Co-operative Processing Communications Platform, now called Communications Platform. The OS 2200 communications control component.


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.)

DCOM: See .Net

DMS 2200: Hierarchical database on ClearPath OS 2200 systems, now called Network Database Server for ClearPath OS 2200

DPS: Display Processing System. OS 2200 software providing a device-independent interface to applications.

DTI: Distributed Transaction Integration. A multipurpose middleware product which provides access to applications or transactions running on ClearPath systems from a Microsoft Windows environment. Also provides access to other transaction environments including Oracle Tuxedo. This product combines the capabilities previously provided by Transaction Integrator (formerly WebTx) and Open Transaction Integrator (OpenTI).

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 TIP/HVTP, 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 partition in ClearPath systems supporting s-Par®, or as an appliance in other ClearPath systems.

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.
**FCSS:** File Control Superstructure. Part of the OS 2200 file system.

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

**HAA:** Heritage Application Access. A component of Open DTP.

**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 Client for DPS:** Allows applications using DPS to be accessed from Web browsers. Current name is Web Enabler for Display Processing System.

**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.

**Java SE:** Java Platform, Standard Edition.

**JAX-WS:** Formerly JAX-RPC (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 OS 2200.

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

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

**MCP**: Master Control Program. Operating system for ClearPath Libra systems.

**MQI**: Message Queuing Interface.

**MQM**: Message Queueing Middleware.

**MSMQ**: Microsoft Message Queuing. Microsoft’s message queuing middleware.

**MSMQI**: Microsoft Message Queueing Interface.

**NTSI**: Message Integration Services for ClearPath OS 2200. An additional API over COMAPI providing a simple interface for communication between OS 2200 and Windows programs.

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

**OMG**: Object Management Group. Not for profit, open, industry consortium developing enterprise integration standards.

**Open DTP**: The 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

**PID**: Position Identifier. Position Identifier. Used in TIP/HVTIP to identify the origin of input requests.

**PMM**: Processor Memory Module. A part of a ClearPath fabric-based system containing the firmware to house the OS 2200 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.

**QProcessor**: Provides a runtime environment for WebSphere MQ. Implemented as a specialty partition in ClearPath Dorado systems supporting s-Par®, or as an appliance in other ClearPath Dorado systems.

**RA**: Resource Adapter. A name for a connector from Java environments to applications or databases.
RDMS 2200: Relational Data Management System for OS 2200, now called Relational Database Server for ClearPath OS 2200

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.

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

SMB: Server Message Block. The Server Message Block (SMB) Protocol is a network file sharing protocol, and as implemented in Microsoft Windows is known as Microsoft SMB Protocol. The set of message packets that defines a particular version of the protocol is called a dialect. The Common Internet File System (CIFS) Protocol is a dialect of SMB.

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.

SRDF: Symmetrix Remote Data Facility, a software product from EMC Corporation for replicating data on a remote storage subsystem. It can be synchronous or asynchronous (SRDF/A). Which is used depends on the business requirements, the volume of data and the distance between the two subsystems.

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.

TCIS: TSAM Communications Interface Software, a re-orderable customer engineering product. Replaced by CITA.

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.

TIP/HVTIP: Transaction Interface Processor/High Volume TIP. Native transaction processing environments on ClearPath OS 2200 systems.

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

TM2200: Implements the TM and CRM functions in Open DTP.

Tomcat: Open Source web application server from the Apache Software Foundation
Tuxedo: Distributed transaction application server. 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.

UCS: Universal Compiling System. A platform for applications development on extended mode systems, comprising language compilers, supporting software products, and interfaces to other Unisys products.

UDS: Universal Data System, a suite of data management software components and products which form an integrated environment for control, maintenance, and recovery of user databases. Includes DMS 2200, RDMS 2200 and FCSS.


 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.


UTS: Universal Terminal System. A family of green-screen terminals with defined link and display protocols. Emulators available for PCs.

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


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.

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. WebSphere® MQ Version 7 for ClearPath OS 2200 (WebSphere MQ) is the follow-on product to MQSeries for ClearPath OS 2200. Executes in the QProcessor.

WebTS: Web Transaction Server for ClearPath OS 2200. Provides a Web-compatible interface directly to the ClearPath OS 2200 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.

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 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.

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

**XML:** eXtensible Markup Language. A language defined by the W3C for encoding messages and protocols in a machine-independent way.

**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>October 1997</td>
<td>First version of paper</td>
</tr>
<tr>
<td>2</td>
<td>March 2001</td>
<td>New edition, with a new title, and major revision of content</td>
</tr>
<tr>
<td></td>
<td></td>
<td>and structure</td>
</tr>
<tr>
<td>3</td>
<td>April 2002</td>
<td>Updated for ClearPath Release 7.0</td>
</tr>
<tr>
<td>4</td>
<td>October 2007</td>
<td>Major revision to bring content up to ClearPath Release 11.1</td>
</tr>
<tr>
<td>5</td>
<td>October 2009</td>
<td>Updated for ClearPath Release 12.0, edited and reformatted</td>
</tr>
<tr>
<td>6</td>
<td>February 2015</td>
<td>Major revision to ClearPath release 15</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*