USING OPEN TRANSACTION INTEGRATOR IN THE WINDOWS ENVIRONMENT

White Paper

UNiSYS
CONTENTS

1 Executive Summary
2 About This Paper
3 Transaction Processing Basics
5 Microsoft Transaction Services
6 An Overview of COM+ Components
8 An Overview of Web Services
9 Open Transaction Integrator in the Open DTP Environment
16 An Overview of the Open Transaction Integrator Product Components
18 Summary
Executive Summary

In today’s data centers, heterogeneous platforms require integrated solutions for fully distributed transaction processing. The Open Transaction Integrator from Unisys is an advanced software tool for integrating distributed transaction processing services, compliant with The Open Group’s Distributed Transaction Processing model, within the Microsoft infrastructure. Open Transaction Integrator is one component of Distributed Transaction Integration, a transactional middleware product for ClearPath and ClearPath Plus servers.

Open Transaction Integrator enables you to:

➤ Integrate applications and data in your transaction processing systems with new or existing Microsoft COM and .NET applications, or make data available to Web services.
➤ Create or enhance applications in a convenient, common development environment
➤ Expand the scope of existing solutions with new application logic and Web presentation logic
➤ Take full advantage of the benefits of open transaction processing, such as bi-directional, two-phase committed transactions

Microsoft Windows 2000 is proving to be a capable and attractive platform for enterprise-level transaction processing. For Windows 2000 servers, Microsoft has established a paradigm for transaction processing based on the Microsoft Component Object Model (COM) and transactional extensions (COM+).

With Open Transaction Integrator, any application that is Tuxedo-based or that uses The Open Group specification, such as Unisys Open Distributed Transaction Processing (formerly called Open/OLTP), can participate in transactional relationships with Microsoft COM+ components.

About This Paper

This paper describes the role that Open Transaction Integrator plays in a Windows-based transaction-processing environment.

First, however, the paper offers some basics of distributed transaction processing (DTP). This provides a general background for understanding the technology underlying Open Transaction Integrator. You may want to skip this material if you are already familiar with these concepts.
Transaction Processing Basics

Transaction processing—often called online transaction processing or OLTP—is a computing model in which the prerequisite for a transaction to complete is the successful completion of all tasks within the transaction. A single transaction in a travel reservation system, for example, might involve updating airline, hotel, and car rental database files. All the updates must be completed for the overall transaction to take place; if any updates fail, the transaction itself will not be completed.

Transaction processing typically occurs in real-time with many end users concurrently accessing applications and updating common databases. Users can also queue transactional messages that are processed asynchronously. These OLTP applications are also often distributed across multiple servers. In distributed processing, the various tasks comprising the transaction are performed on different computer systems within the network. The success of this exacting environment depends on a number of processing “guarantees” such as the ACID properties and standards such as The Open Group’s Distributed Transaction Processing reference model—discussed below.

The ACID Properties

In order for transactions to behave as expected, they require the following ACID properties:

➤ Atomicity. A transaction can consist of multiple tasks, but all tasks must succeed for the transaction to take place. No individual task is performed unless all can be performed.

➤ Consistency. A transaction must leave the system in a correct state.

➤ Isolation. A transaction is independent; it is not affected by other transactions that may be executing concurrently.

➤ Durability. Once a transaction successfully completes, the result of the transaction on the system is permanent.

The Open Group’s DTP Reference Model

The Open Group’s Distributed Transaction Processing reference model defines the standard components of a transaction system and the interfaces among them. This globally accepted model defines an architecture that ensures that all computer systems adhering to the standard—regardless of manufacturer or processor type—can participate in distributed transactions.

In addition to the application programs themselves, the basic components of the model include:

➤ Resource managers that control shared resources such as databases and file systems

➤ Transaction managers that control and coordinate the various resource managers

➤ Communication resource managers that control communications among distributed applications
The Open Group model also defines the interfaces among the various components. One such interface that relates directly to Open Transaction Integrator is XATMI. This is an interface between applications and communication resource managers based on Tuxedo’s Application/Transaction Management Interface (ATMI). Figure 1 presents a simple diagram of The Open Group model.

**Figure 1. The Open Group’s DTP Model**

**OSI TP**

The Open Systems Interconnection Transaction Processing standard (OSI TP) provides a standards-based network protocol for communicating between the domains of different transaction managers. Below the OSI TP message layer, communication resource managers can support industry-standard transport protocols such as TCP/IP and OSI.

An important part of the OSI transaction-processing standard is the definition of the two-phase commit protocol.

**Two-Phase Commit Protocol**

Two-phase commit is a technique for ensuring that transactions complete successfully across different systems. A successful transaction comprises two phases:

- **First Phase:** A transaction coordinator alerts all the involved computing nodes to the pending outcome of a transaction and asks the nodes to prepare to commit. All tasks are completed and, if successful, the nodes report that they are ready to commit.

- **Second Phase:** When the ready-to-commit condition is unanimous, the transaction coordinator directs the nodes to make their part of the transaction durable. The nodes then report back confirmations.

  If a single task fails prior to the second phase, then the transaction itself is aborted. All the nodes must perform a rollback—bringing their systems back to the state before the transaction was received. If a failure occurs in the second phase, then recovery procedures ensure that the transaction is successfully committed.
Client/Server Architectures
There are many client/server architectures. They include multiple variations of two-tier models that divide application code between the user interface on the client and the database on the server.

The three-tier client/server paradigm, however, is of particular significance to Open Transaction Integrator. In this model, the application code containing the business logic exists in a middle tier—separate from the user interface and data server. The advantages of this approach include:

➤ Improved scalability—connections can be pooled and managed by resource dispensers
➤ Greater flexibility—applications can integrate data from multiple sources

Transaction Processing (TP) Monitors
TP Monitors are software systems that fit in the middle tier of a three-tier client/server model. These control programs oversee transaction processing by performing functions such as:

➤ Starting server processes
➤ Ensuring transaction integrity by guaranteeing the ACID properties
➤ Controlling resources so that server applications can scale to many simultaneous users
➤ Balancing loads between clients and servers
➤ Restarting processes after failures

Open Transaction Integrator Optimizations
Open Transaction Integrator takes advantage of patented Unisys high performance OSI TP algorithms. This offers advantages over the standard OSI TP implementations by optimizing data movement throughout the OSI TP protocol stack using TM algorithms.

In addition, the high-performance OSI TP supports NATIVE-A encoding for data sent to and received by Unisys MCP systems. This eliminates XATMI field-by-field encoding and provides better performance for data translation.

Also optimized is the sending and receiving of data to other Unisys implementations, including MCP OSI TP and BEA eLink OSI TP. In these cases, Open Transaction Integrator connection manager uses multiplexed TCP-IP connections to eliminate the need for OSI protocol layers. At the core of all Open Transaction Integrator interoperability remains the OSI TP main engine. This provides the ACID properties for all of distributed transaction processing.
Microsoft Transaction Services

An important development in the maturation of the Windows operating system has been the evolution of its transaction services from Microsoft Transaction Server, to COM+ and .NET enterprise services. As a result, the Windows platform is a first-choice transaction-processing environment. Results from the Transaction Processing Performance Council show that overall performance of Windows-based solutions is among the highest and within the top ten for performance results.

Transaction Integration

Mergers, acquisitions, and departmental consolidations are realities in today’s business climate. As a result, a variety of systems exist in most data centers. For example, a single company may have a legacy payroll system running on an IBM mainframe, an Oracle-based accounting application on a Unix server, and a customer support system running on a Unisys ClearPath server. Integrating such a heterogeneous environment can be a daunting task. Enhancing it might seem impossible.

ClearPath Plus middleware provides a unique integration environment, one that makes it possible to integrate diverse services from legacy systems to form new, more powerful, applications. Products such as Open Transaction Integrator from Unisys, MQSeries, and COMTI integrate applications by using components to represent the various services. Clients running on a wide variety of platforms can then access these components through the Internet/intranet or by submitting requests in the form of messages. As a result, you can create distributed transactions that incorporate a combination of components and access different legacy systems.

By developing and incorporating new Microsoft COM-based applications, you can add new features to existing systems. You can also integrate features found in readily available off-the-shelf applications. Products such as MS BizTalk Server and Siebel all have COM interfaces and therefore can be easily integrated with Open Transaction Integrator applications. In addition, the .NET framework contains a facility for COM interoperability that allows the integration of .NET applications. Using the Simple Open Access Protocol (SOAP) toolkit from Microsoft, Open Transaction Integrator components can be used as Web services.

Component-Based Technology

COM+ is based on the Microsoft Component Object Model. This enables developers to build binary software components using a variety of popular development tools, such as Microsoft Visual Basic and Microsoft Visual C++. Once created, these single-user ActiveX Server components are installed in the runtime environment as part of COM+ applications.

This component-based approach enables programmers to create applications quickly while significantly reducing software development and maintenance costs. Specifically, programmers can:

➤ Reuse and share components among applications
➤ Update specific components to respond quickly to changes in business requirements
➤ Use a variety of components: those they create, those shared with other applications, or those from third parties
➤ Easily deploy applications from test beds to production machines
An Overview of COM+ Components

With COM+, business logic is divided into multiple object-based components. Components are combined to create applications that can:

➤ Update one or more databases
➤ Invoke mainframe applications
➤ Send or receive messages

The COM+ environment that performs these tasks consists of the following components:

➤ COM+ Runtime Environment
➤ COM+ Component Services
➤ Distributed Transaction Coordinator (DTC)

Figure 2 shows the COM+ environment.
**COM+ Runtime Environment**

This COM+ transaction-processing infrastructure does the following:

- Manages system resources (such as processes, threads, and database connections)
- Manages the creation and deletion of components
- Controls transactions
- Implements security
- Provides tools for configuring and managing the transaction processing environment

**COM+ Component Services**

This GUI administration tool helps manage the components under COM+ to:

- Organize components in the runtime environment
- Provide security by defining roles and then assigning user privileges
- Package components together that perform related application functions

**Distributed Transaction Coordinator (DTC)**

COM+ provides a Distributed Transaction Coordinator (DTC) to provide transaction coordination among multiple resource managers.
An Overview of Web Services

What are Web Services?
Web services are modules of code that can be reused in combination with other Web services to fit the needs of the end user. In some respects they are like components, such as J2EE or MS COM+ components, but they allow for greater platform independence. They also use Internet protocols that provide programmable access to back-end transactions or databases. Because they are standards-based, Web services can be used on virtually any platform.

What is the Business Value of Web Services?
The value of Web services is that they are easily extended and can be distributed to multiple platforms because they are standards-based. They can access databases regardless of platform. They are quick to implement and require fewer resources to implement than other integration methods.

When to Use Web Services
Web services work well for Intranet and Extranet situations. They are an ideal integration technology where you need to support long business processes that move data through departments or between partners, such as phased-in approval processes. They are also ideal for interoperability across heterogeneous platforms. Currently, the standards governing Web services do not offer security that is acceptable to all industries. Improvements in security are forthcoming, but until they arrive, Web services are best suited for internal or closed implementations where high levels of security are not needed.
Open Transaction Integrator in the Open DTP Environment

High-volume transaction processing environments are mission critical in today’s data centers. Transaction processing applications run on a variety of platforms: mainframes, Unix servers, and Windows servers. Databases, and the transactions that continually update them, seem to be isolated in their respective platforms. Many data centers could achieve greater efficiencies, and draw additional value from their applications and data, if their Windows and other transaction processing environments were better integrated. This is the specific problem that Open Transaction Integrator resolves.

What is Open Transaction Integrator?

Open Transaction Integrator from Unisys provides application runtime and development environments that integrate with those of COM+. As such, both Open Transaction Integrator and COM+ fit in the three-tier client/server model defined by the Microsoft Web Solution Platform architecture.

Open Transaction Integrator provides two distinct capabilities. First, it enables developers to integrate COM+ transactions with existing DTP services that are compliant with the Open Group’s DTP model. The DTP services can join COM+ transactions. This makes it possible to generate components that map to existing:

- Open Distributed Transaction Processing services on Unisys mainframes
- BEA Tuxedo services on a number of popular platforms
- TPMSX on Open VME mainframes

Second, Open Transaction Integrator makes it possible to enhance existing OLTP applications by creating COM+ components that are invoked using XATMI interfaces. Here COM+ components can join DTP transactions. This is a powerful tool for developing new features or integrating off-the-shelf software into existing applications. These new components integrate seamlessly because they are called using the application’s native interfaces.

Because Open Transaction Integrator creates standard COM components, it can be easily integrated with Windows applications from Siebel and others. This integration can be achieved through COM interfaces or through application integration tools such as Microsoft BizTalk Server 2000.

To protect unauthorized access to OLTP applications, Open Transaction Integrator provides three levels of security: static IP addresses, connection-level challenge/response, and user authentication.

In total, Open Transaction Integrator is a solution designed for enterprise-class clients who depend on high-volume transaction processing. During runtime, for example, Open Transaction Integrator supports multiple threads to maximize scalability and performance on multiprocessor Windows servers. It also offers a fully automated component builder, and ease-of-use features such as wizards, online help, tutorials, and sample programs in Visual Basic, Visual C++, and Active Server Pages.

Now, let’s take a closer look at the major Open Transaction Integrator capabilities.
Accessing OLTP Applications From COM+ Using Open Transaction Integrator Components

Figure 3 illustrates how client applications use COM+ and Open Transaction Integrator components within the Windows Web Solution Platform architecture to access OLTP services running on remote OLTP systems. In this architecture, tier 1 provides the user presentation and supports user interaction through many different devices—including Web browsers, wireless devices and Windows terminals. Through this support, Open Transaction Integrator lets you provide Windows client access to applications and Web access to OLTP services.

Tier 2 applies business logic to data from many different sources. A variety of tools can be used to create components for the COM+ component library. Open Transaction Integrator Builder from Unisys makes it possible to create components accessing XATMI-compatible remote services. Microsoft offers the COM Transaction Integrator (COMTI) for developing components from existing IBM CICS and IMS transaction programs. In addition, components can be created using a wide array of popular COM software development tools.
Tier 3 provides the data services—both data storage and access to mainframe services. Open Transaction Integrator provides runtime facilities in this tier that enable COM+ to treat legacy OLTP and Tuxedo transactions as a data store. This allows existing OLTP business logic to continue residing on the server.

The Open Transaction Integrator components you create are compatible with standard COM components and easily registered with COM+ as transactional, multithread objects. An Open Transaction Integrator component contains methods—one for each remote service—that describe the parameters corresponding to usable fields of the services. These fields may be STRING, CARRAY, XML or X-OCTET buffers, or fields of a VIEW buffer. By setting the parameters corresponding to the fields of the input VIEW buffer, and invoking the method, an application programmer can cause the Open Transaction Integrator Runtime to communicate with the remote service. The method returns parameters corresponding to the fields of the output VIEW or data of the other buffer types.

Accessing COM+ and Open Transaction Integrator Components From OLTP Applications

Figure 4 illustrates how OLTP programs—such as existing Unisys Open Distributed Transaction Processing, TPMSX for OpenVME, and BEA Tuxedo applications—can access COM+ server components. This access makes it possible to enhance applications by adding new functionality that you create yourself, or by “wrapping” features available in off-the-shelf applications.
To build a COM+ server component that uses Open Transaction Integrator, it is necessary to first use the Open Transaction Integrator Builder to create a server definition file. This file contains a type library that can be used in a Visual C/C++ ATL-based component or a Visual Basic component. The Open Transaction Integrator Builder will export stubs in Visual Basic and provide instructions to enable programmers to begin COM+ component development. These method stubs either contain the new features for the applications or can be used to wrap the methods from other COM+ components. Existing OLTP applications call these methods using the familiar “tpcall,” “tpacall,” and “tpconnect” functions of the XATMI programming interface. This allows programmers to work in a familiar programming paradigm.

Open Transaction Integrator Compatibility with COM+

Open Transaction Integrator coexists with the Microsoft COMTI and COM+ runtime environments. This means that:

➤ Components created with either integrator and be combined. When registered with COM+, you can assemble any of these components to create multi-tiered applications for the enterprise.

➤ Open Transaction Integrator supports both network and user authentication with peer OLTP systems using OSI TP security extensions. Open Transaction Integrator user authentication can be combined with the declarative security through COM+ Component Services. Administrators can grant access to server packages using roles and Windows-based user and group accounts. Open Transaction Integrator allows both local and global access control policy to control access to components in the heterogeneous environment.

➤ Open Transaction Integrator has advanced deployment capabilities that build on the COM+ export utility. This allows the entire Open Transaction Integrator and COM+ environment to be deployed to a production machine from a test system.

The following example illustrates the value of Open Transaction Integrator in a customer environment. A large retail bank uses Open Transaction Integrator to integrate the insurance applications and banking applications that their employees and customers access using the Web. The insurance applications comprise an OS 2200 Business Information Server application, with some newer applications developed in Microsoft COM+. The banking applications use a J2EE environment in a Unix platform and IBM OS390.

The bank uses MQSeries as the interface method between the Windows and IBM systems that host the insurance and banking applications. MQSeries on the Windows server invokes Open Transaction Integrator for data updates in the insurance applications. Open Transaction Integrator connects into an Open Distributed Transaction Processing service running in the OS 2200 system. This service then passes the message to the Business Information Server application and returns its replies. Because Open Transaction Integrator is bi-directional, the Business Information Server application can also initiate requests to external environments, such as SQL Server databases on the Windows platform.
Open Transaction Integrator Compatibility with Siebel Applications

Siebel’s eScript language makes it possible to:

➤ Integrate OLTP services into Siebel applications

➤ Create an instance of an Open Transaction Integrator component and call a method corresponding to the OLTP service with code written in eScript

➤ Use Open Transaction Integrator to return a string containing the data returned from the service as an XML document. The Microsoft XML (MSXML) parser, or any XML parser, can then be used to access individual data fields. This allows Open Transaction Integrator to get around the eScript limitation of not allowing any output parameters returned on a method call

For example, a state human services organization uses Open Transaction Integrator to integrate data and transactions between Siebel and ClearPath applications. XML is the transport vehicle for the data. This functionality yields the following advantages for the organization’s statewide social workers:

➤ **Efficiency:** A client’s data in multiple databases is consistently and reliably updated with a single transaction

➤ **Productivity:** Thousands of transactions per day are completed with data integrity and efficiency

➤ **Improved Service:** Clients receive better service from social workers who are better informed and have more time for casework
Open Transaction Integrator Compatibility with Microsoft BizTalk Server 2000

Legacy OLTP services can be included as part of the overall Enterprise Application Integration (EAI) architecture using BizTalk Server 2000 along with Open Transaction Integrator.

With BizTalk messaging, for example, you can code an Application Integration component (AIC) that creates an instance of an Open Transaction Integrator component. This component can call a method that in turn calls an OLTP service. Add this AIC as a destination of a channel to enable post-mapped data (the document received by BizTalk after modifications have been made according to the channel configuration) to be sent to the OLTP system. Using the BizTalk scriptor AIC that comes with BizTalk, you can write just a few lines of Visual Basic Script (vbscript) code to call the OLTP service.

With BizTalk Orchestration, OLTP services can also be included as part of a BizTalk Orchestration XLANG schedule. Due to limitations in the Orchestration Designer, the Open Transaction Integrator components cannot be called directly, but must be called through a wrapper component created in Visual Basic or Visual C++. For maximum scalability, Visual C++ is recommended to create the wrapper component. The wrapper component contains the code that instantiates the Open Transaction Integrator component and calls the method for the OLTP service. The COM binding wizard in BizTalk Orchestration designer is then used to include the wrapper component on the schedule. If both the wrapper component and the Open Transaction Integrator component are set to “support transactions” in COM+, and the components are called from within a transaction on the XLANG schedule, the transaction will flow to the OLTP system. The work of the service will be undone if the transaction fails to complete.

Open Transaction Integrator Compatibility with MS .Net Web Services

Web services use a loosely coupled message-based model. The request and response data is described in XML using Simple Object Access Protocol (SOAP). Using Visual Studio .Net, programmers can focus on writing business logic rather than the details of building or processing a SOAP request buffer. Open Transaction Integrator has no knowledge that it is being called as a Web service, and does not need to know. To Open Transaction Integrator, the Web service code written in any .Net language is the Open Transaction Integration client.

The following is an example of using Web services with Open Transaction Integrator. The programmer uses one service from a previously created test program and makes it a Web service. The service uses all VIEW field data types, so the Web method had most of the .Net data types supported by OpenTI, including arrays. The Web service passes the input parameters to Open Transaction Integrator. Open Transaction Integrator returns the response as an XML string, which is the return value for the Web service.

The programmer then codes a client using VB .Net. Visual Studio .Net has excellent support for creating and consuming Web services. In the VB .Net project, the programmer adds a Web reference for the service. This causes VS .Net to read the disco file for the Web service, which contains the method definitions.

To call the Web service, you instantiate an instance of the class containing the service, and call the method. A Web service proxy on the client side builds the request into a SOAP-formatted XML document, and sends it to the server.

Note: Multidimensional arrays are not supported, so for the VIEW field type array of CARRAY, only an array of strings can be passed.
Open Transaction Integrator Security
Open Transaction Integrator supports three main types of security:

➤ Static IP Addresses. At the network level, all hosts participating in the OSI-TP network must be configured explicitly. The IP address must be known at configuration time. DHCP networks are not currently supported. To add new remote server nodes, each may be configured dynamically on all connecting systems.

➤ Link Layer Security. For participating peer systems, Open Transaction Integrator will use a challenge-response algorithm before allowing connections to be established.

➤ User Authentication. This allows user ID information to be passed between heterogeneous systems. The Open Transaction Integrator administrator selects an Access Control Policy that controls how incoming user IDs are handled. The User Authentication feature can be used in conjunction with the COM+ role-based security for Access Control.

Access control is provided by role-based security available to COM+ applications. This feature allows the administrator to set access permissions on a particular component or COM+ application. This is available to Open Transaction Integrator components produced by the Open Transaction Integrator Builder as well as Open Transaction Integrator server components. A role is a symbolic name that defines a logical group of users who have access to an application or component. For example, an online banking application might define roles for Manager and Teller.
An Overview of the Open Transaction Integrator Product Components

Open Transaction Integrator is a comprehensive product consisting of:
➤ Open Transaction Integrator Builder
➤ Open Transaction Integrator Runtime
➤ Open Transaction Integrator Management Console
➤ Open Transaction Integrator Component Remove Utility

Open Transaction Integrator Builder
The Open Transaction Integrator Builder provides the application developer with an easy-to-use, GUI tool for creating Windows-based component libraries and for exporting or importing the XATMI VIEW definitions used to communicate with OLTP services. Use the Open Transaction Integrator Builder to create libraries for both new client components and server definition files.

Client Components
The Open Transaction Integrator Builder can create Windows-based client component libraries in any of the following ways:
➤ Importing an existing XATMI VIEW buffer to convert to automation method definition
➤ Importing an existing Field Manipulation Language buffer (FML) to convert to an automation method definition and export an XATMI VIEW buffer
➤ Creating a method definition by adding and deleting parameters; then exporting an XATMI VIEW buffer
➤ Importing an existing COBOL data definition to convert to an automation method definition containing an XML string

Once the component library is constructed, the application programmer can develop the client application to access the data. The client application can be written in any language that supports automation, including Visual Basic, Java, Visual C/C++, and Active Server Pages. You can then drag and drop the newly created type library (the component .dll file) into Component Services for COM+. OLTP buffer types of VIEW, VIEW32, X-OCTET, STRING, CARRAY and XML are also supported.
**Server Definition Files**

The Open Transaction Integrator Builder can create Windows-based server definitions in either of the following ways:

- Importing an existing XATMI VIEW buffer to convert to an automation method definition
- Creating a method definition by adding and deleting data buffers. Non-VIEW data buffers will have one parameter, while VIEW data buffers will have one parameter for each field. VIEW definitions can be exported to a text file for use in an OLTP system

When creating server definition files, Open Transaction Integrator Builder can:

- Create method stubs to be included in the Visual Basic server component
- Use optional buffer types of STRING, CARRAY, XML or X-OCTET
- Import an existing COBOL data definition to convert to an automation method definition containing an XML string

**Open Transaction Integrator Runtime**

Open Transaction Integrator Runtime provides a generic automation server for all the components created by the Open Transaction Integrator Builder. When an application calls an Open Transaction Integrator component method, the Open Transaction Integrator Runtime translates the input parameters into a buffer that the service understands. When the service returns an output buffer, Open Transaction Integrator Runtime translates the contents into parameters or an XML document that the calling application understands.

Functionally, Open Transaction Integrator Runtime acts as both a resource manager and connection manager. As a resource manager, Open Transaction Integrator:

- Enables Open Transaction Integrator components to participate in DTC transactions
- Provides automatic enlistment of Open Transaction Integrator transactions with the inherent two-phase commit capabilities of DTC

As a connection manager, Open Transaction Integrator can:

- Control connections to both Unisys Open Distributed Transaction Processing and BEA Tuxedo services using OSI-TP. This includes preserving the ACID properties of such transactions, as well as supporting bi-directional, two-phase commit.
- Control connections to many simultaneous hosts. This includes pooling connections efficiently and releasing unused connections.
- Ensure durable connections for network or node failures. Through extensive recovery facilities, Open Transaction Integrator can maintain transaction states from lost connections, node crashes, and application or COM+ failure.
Open Transaction Integrator Management Console
The Open Transaction Integrator Management Console extends the Microsoft Management Console (MMC) environment. It consists of the following two parts:

➤ Open Transaction Integrator extension snap-in. This makes it possible to assign Open Transaction Integrator components to different remote hosts, to manage the remote host properties of these components from within the COM+ component services, and to set security attributes for a component.

➤ Open Transaction Integrator MMC snap-in. This is used to configure all the hosts in the Open Transaction Integrator network, to manage the Open Transaction Integrator server definition files that relate a particular OLTP service name with a COM+ component, and to manage Open Transaction Integrator security information of network and user authentication.

Once the application developer has created an Open Transaction Integrator component—and installed it in a COM+ application—the administrator can display and update the remote host information by accessing the properties of the component.

Open Transaction Integrator Component Remove Utility
The Open Transaction Integrator Component Remove Utility provides centralized removal of both Open Transaction Integrator and COM+ components and their registry entries. This ensures that all traces of a particular component are removed.

Summary
With the development of COM+, transaction processing based on Windows platforms offers many advantages for enterprise server clients. Unisys Open Transaction Integrator enhances this environment by providing a component builder and transactional runtime system that integrate COM+ with existing distributed transaction-processing systems that are compliant with The Open Group’s DTP model.

Unlike competitive offerings, Unisys Open Transaction Integrator is an extension of the Microsoft development environment. As a result, it offers a cost-effective means of creating standard COM components that can interact with legacy transaction systems such as Unisys Open Distributed Transaction Processing and BEA Tuxedo. And because Open Transaction Integrator creates standard COM components it can be integrated easily with Windows applications from Siebel and others. This integration can be achieved through COM interfaces or through application integration tools such as Microsoft BizTalk Server 2000.