UDICo's TierBroker

Product Technical Overview

Universal Data Interface Corporation Technical Document

VERSION 2.0b, April 25, 2001

Copyright Notice and Non-Disclosure

Table of Contents

1. TierBroker and the Future of Middleware

TierBroker is a middleware product that lets you easily design simple or complex workflows between a wide choice of data resources and then lets you run those workflows at high speed, in real time, on a variety of platforms. This white paper describes trends in the middleware market and why TierBroker's architecture makes it particularly well-suited to provide the efficiency and flexibility that will be required of more and more information management systems in the near-term and long-term future.

1.1. *What Does Middleware Do?*

Middleware makes it possible for systems that couldn't otherwise exchange data to do so. A good middleware system must make it easy to design, execute and maintain the exchange of data between systems, but it's not always so simple:

- As the number of systems that must be connected goes up, the number of potential links between them increases geometrically. A good middleware system must be able to scale up with the systems it integrates.
- Different systems usually expect data in different formats. A good middleware system must make the specification and integration of conversion specifications simple and straightforward.
- Data to convert and rout may suddenly appear in a specified directory as a multi-gigabyte disk file or it may be sent in a high-speed stream to a communications port. A good middleware system must be able to handle batch and real-time data equally well.
- As companies acquire new systems and change old ones, the ability to change the routing of data must be as simple as flicking a switch. In fact, it should be easier than that—a middleware system should enable content-based routing, which sends data to one destination if the value in a data field meets a certain condition and to another destination if not.
- Flexibility and larger capacity often come at a cost to the speed of execution, but a good middleware system must be able to handle high volumes at high speeds.
- A good middleware system should be able to handle data in any format. Common popular formats such as EDI, SWIFT, XML, HTML, and CSV should be easier to set up and use than unusual custom formats.
- Messages can sometimes come from or be sent to unreliable communications pipes. A good middleware system has the kind of transaction control that can add reliability to such communication.

For modern businesses, the ability to replace key systems is a technological shield that protects institutions in an environment where change occurs too quickly for anyone to predict long-term winners and losers. Successful businesses maximize the value of tactical solutions by incorporating the capacity to adapt their system architecture to evolving models. They can take advantage of short-term enhancements that would otherwise be too risky to consider.

A middleware product that successfully meets all of the above requirements makes this flexibility possible. A good middleware system plays a strategic role in enabling the kind of plug and play approach to enterprise-wide systems that lets an institution react quickly, discarding outmoded technology and taking advantage of new opportunities whenever necessary.

1.2. No More Specialized Middleware

Today, companies use different kinds of middleware to serve different needs. For example, Enterprise Application Integration (EAI) products such as WebMethods and Vitria focus on near real-time messaging, and Data Warehousing products such as Informatica and Mercator focus on extraction and transformation. Most companies that purchase middleware need to implement the same interfaces with multiple packages to satisfy both batch processing and real time requirements; the increased cost, effort, and complexity of this redundancy usually hurts the very efficiency that middleware is supposed to bring to an enterprise.

Until now, the only kind of general-purpose middleware that could meet all of these needs cost hundreds of thousands of dollars, putting it beyond the budget of small and mid-sized companies trying to grow to the next level. Today, the increasing availability of robust tools and the emergence of standard protocols let developers do a lot of interfacing without great expense. Java-based technologies such as J2EE and XML-based technologies such as XSLT can be used to assemble flexible systems, but with performance and maintenance issues that prevent such systems from scaling up to even a medium-sized organization.

There is a growing need for fast, scalable, affordable, general-purpose middleware today, and TierBroker fills that need. Its history of deployment at major financial institutions for global systems has demonstrated its ability to handle high-speed, high-volume requirements, and its ability to handle batch and real-time processing, extraction and transformation, and transaction management show that it can fill the need for reasonably-priced, robust general purpose middleware at enterprises of all sizes.

TierBroker's scalability, portability to multiple platforms, support for multiple languages in specifying business logic, XML (and in particular, SOAP) support add up to a middleware system that lets you easily connect legacy systems to serve a range of needs, including the growing need of systems to function as both web service clients and web service providers.

1.3. The .NET Vision

Microsoft is currently promoting a vision of software development, deployment, and use via XML-based web services that they call ".NET." Developers working within the .NET Framework will be able to select from over 20 programming languages and compile these programs to run in a virtual machine called "Common Language Runtime" (CLR). By taking advantage of Microsoft's Frameworks Class Library (FCL), different programs—even when written in different programming languages—will be able to work together to provide web services using the BizTalk Framework. (The .NET Framework development environment is still in beta release.)

The BizTalk Framework is a set of messaging specifications built around the XML, HTTP, MIME, and SOAP standards. For example, one part of the Framework is a specification for a BizTalk message as a SOAP envelope that includes specific BizTalk elements as children of the SOAP header to provide information expected by .NET-compliant software.

.NET and the TierBroker

TierBroker can provide all the benefits of a .NET approach right now on a wider selection of platforms. TierBroker's interoperability, support for multiple development languages, robust transaction support, SOAP support and high-speed real-time server capabilities make it an excellent choice for the development and deployment of every type of web services made possible by .NET.

Industry analysts agree that, regardless of the success of the product line underlying Microsoft's new vision, these kinds of web services will play an increasingly important role in the future of business and consumer computing. Whether an enterprise commits to the .NET product line or not, TierBroker will make the development, deployment and maintenance of such web services simple and easy.

BizTalk Server

A key component in the use and deployment of .NET web services is Microsoft's BizTalk Server. This product, which must be installed on a machine running the Windows 2000 operating system, enables the creation and execution of message exchange between subsystems in the same intranet and between systems communicating via the public Internet. Because of this latter capability, Microsoft is pushing BizTalk Server as a solution for implementing XML-based B2B processes. BizTalk Server includes various graphical tools for specifying the structure and routing of your messages; for example, the "Orchestration Designer" (which requires an installed copy of Microsoft Visio to run) creates "schedules" showing message routing.

BizTalk reads information from other processes using two kinds of input channels: Microsoft Message Queue (MSMQ) messaging and directories that serve as drop-off points for files that it should process. The use of other messaging protocols such as IBM's popular MQ Series require third-party converters.

BizTalk can use its communications mechanisms to communicate with any database that supports them, but storage of workflow and routing information BizTalk requires Microsoft's SQL Server.

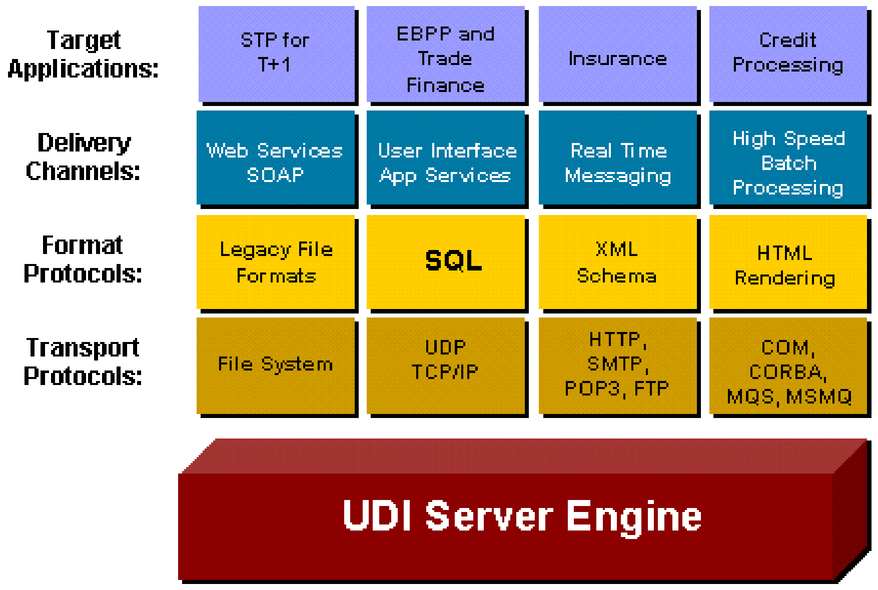
BizTalk Server's components, and any programming components you want to add to the mix, are connected by an extension to Microsoft's Component Object Model (COM) known as COM+, so close integration of an application with BizTalk Server requires a knowledge of COM+.

BizTalk and TierBroker

TierBroker could fill the same role in a B2B system that BizTalk Server was designed to fill before the .NET initiative was even announced. In addition to running on a variety of UNIX operating systems, TierBroker also runs on NT, Windows 98, Windows 95, and Windows 2000—three more Microsoft operating systems than Microsoft's BizTalk Server can run on.

This flexibility extends beyond the operating system level to include database management and messaging. In addition to TierBroker's support for ODBC communication with virtually every database product available, its native support for Oracle, Sybase, Informix, and SQL Server give it a big speed advantage over the many products whose database flexibility is based on the use of an ODBC layer for all database work. For communications, its native support of MQ Series, HTTP, FTP, SMTP, POP3, COM, and CORBA (as well as MSMQ) gives it the kind of versatility that a truly general-purpose middleware system needs to connect arbitrary components within an enterprise or between multiple enterprises.

2. TierBroker Architecture



TierBroker's easy adaptability to so many applications on so many platforms comes from the ease with which a developer can mix and match different choices at different levels of the application architecture. When configuring an input or output queue in a TierBroker workflow, you select the appropriate method of data transport (the transport protocol) and the format of the data (the format protocol) appropriate to the workflow source or destination specified as a requirement for your middleware application.

For example, many describe SOAP communication as XML being sent via HTTP, but they don't realize that the SOAP 1.1 specification also provides for sending the XML of a SOAP request or response over SMTP. TierBroker has built-in support for both HTTP and SMTP, making full SOAP compliance much easier for a web service built using TierBroker.

TierBroker also separates the choice of delivery channel from your choice of format protocol and transport protocol. For input, it can read records joined from a multi-table database, watch a directory for the appearance of files, listen to a message queue, watch for real-time input from a data entry operator using web-based HTML forms, and more. For output, it can do all the same things in reverse—in fact, converting a configured input queue to an output queue is often as simple as changing one dialog box setting.

2.1. Development Architecture

The TierBroker IDE lets you assemble TierBroker middleware projects by dragging and dropping icons, writing simple or complex scripts, and setting configuration parameters for the various pre-built components of your middleware. While it is easy to use, it's not designed to turn receptionists into middleware developers; instead, it lets developers who know what they need specify those needs, set the appropriate parameters, and get their middleware up and running as quickly as possible.

When you assemble a middleware project with TierBroker, you're creating and configuring three kinds of things:

- *Object Classes* define the structure of the objects that TierBroker will read from its input queues and write to its output queues. You can enter these using TierBroker's scripting language for complex object definition, but in most cases you can add object class definitions to your project by telling the TierBroker IDE to read in existing data structure definitions such as an XML DTD, an SQL SELECT statement, or an MS Query file.
- *Workflows* tell the TierBroker Server where to get data objects and what to do with them. A typical workflow has at least one input queue, a process to manipulate the data from that input queue, and at least one output queue. If you've developed and tested a complex workflow that reads from and writes to CSV and XML files on your notebook computer's hard disk, small changes to the workflow queue configurations are all you need to run the same complex workflow on a high-end Sun workstation reading from a Sybase database and writing to an MQ Series message queue.
- *Maps* describe how to convert objects of one class into objects of another class. If your output objects have the same structure and field names as your input objects (for example, when converting the rows of an SQL table into XML elements whose subelements have the same names and ordering as the SQL columns), maps are unnecessary, but when you need, them, the TierBroker IDE makes it easy to rename, reorder, and manipulate the fields of one object to create a new one.

The configuration of workflow queues is where you identify the protocols necessary to get and deliver the data routed by TierBroker. By keeping this information completely separate from the map and object class configuration, TierBroker lets you define object classes and maps with no system-level dependencies, which makes them easier to re-use in multiple projects, which leads to faster application development.

UDICO's TierView, in addition to letting you monitor the progress of the TierBroker Server as it performs production workflows, serves as a powerful debugger for applications in development. You can set breakpoints and then, when execution is interrupted, examine data and perform commands at the debugging prompt before resuming execution. For even greater control over your middleware workflows, the TierBroker engine is available as a C++ class library and API.

2.2. *Scripting and Multiple Language Support*

Business logic associated with workflows is specified using scripts attached to workflows. Routing an order with an invalid customer code to an error queue instead of to its original target is as simple as writing a JavaScript "if" statement. The TierBroker IDE comes with a full-featured text editor that offers context-sensitive help, syntax completion, color-coded keywords, and the ability to define and run your own macros as you create your scripts.

The two default scripting languages supported by TierBroker are ECMAScript (also known by the names of its Netscape implementation, "Javascript," and its Microsoft implementation, "Jscript") for procedural logic and SQL for easier manipulation of table-oriented data.

This combination of JavaScript and SQL gives you a great deal of control over your logic and your data, but you can use other languages as well. TierBroker's "Schema for Language" technology makes it possible to support an arbitrary number of scripting languages for its scripting; a separate white paper describes this in more detail.

3. Case Study: Implementing ERP Systems with TierBroker

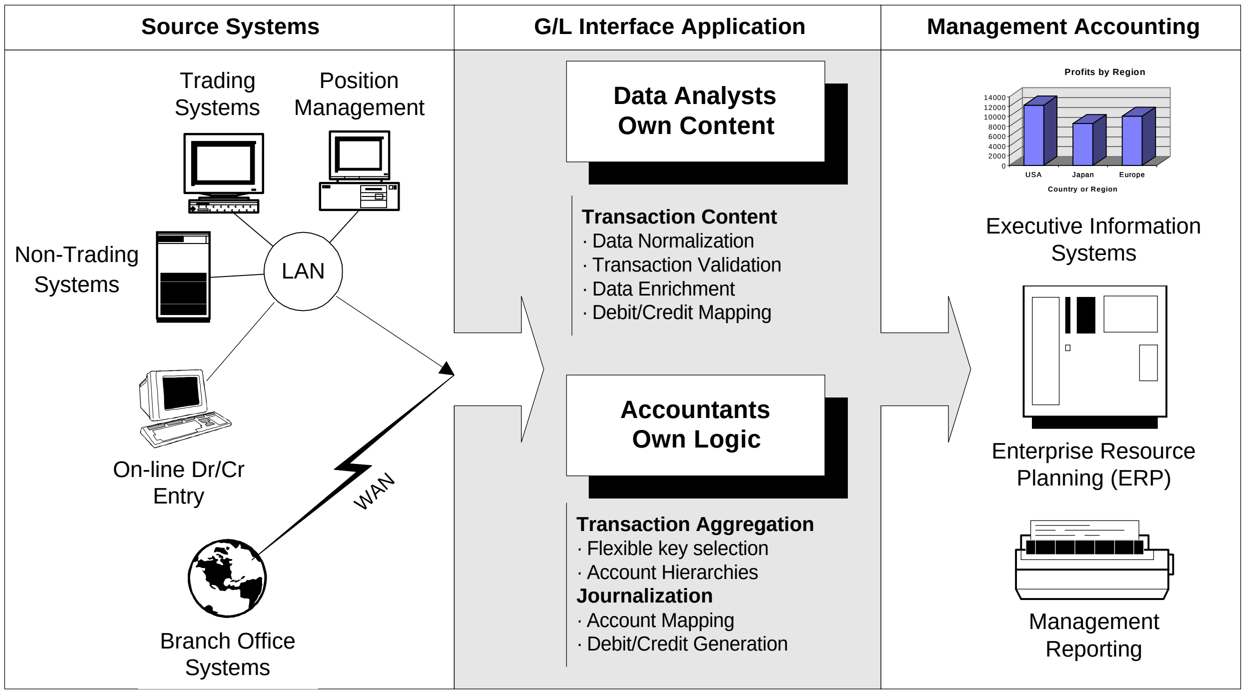
3.1. *Automating Accounting Logic*

TierBroker and the General Ledger (G/L)

TierBroker has been used to implement complex management and financial reporting applications for ERP packages such as PeopleSoft and SAP. Interfaces to the General Ledger (G/L) are complicated by the need to transform transactions into debits and credits; there is no one-to-one correspondence between a transaction type, such as a trade settlement, and a single debit or credit entry. TierBroker accepts these transactions from flat files, databases or real time source systems and translates them into the debit and credit records that represent the accounting activity for those transactions.

What is *Journalization*?

The word *Journalization* refers to the conversion of transactions into a sequence of debits and credits for use in a general ledger. Different fields on the source transaction are used to compute individual debit and credit records.



In many cases, the G/L becomes the first and only way an institution can look at a consolidated, aggregated and enterprise-level view of the entire company's accounting activity. The G/L becomes a type of data warehouse, or data repository, making it possible to reconcile activity from different branches and departments.

3.2. *Reconciliation and Data Warehousing*

The Transaction Data Warehouse

In order to reconcile the G/L back to the source systems, the original transactions are generally stored in a separate database known as an Operational Data Store (ODS). The transactions are *enriched* with data that allows them to be linked to the debits and credits that were generated.

Synchronizing the G/L and ODS

For reconciliation to work effectively, the G/L and ODS must be synchronized. If an insert operation with the G/L succeeds but the corresponding insert to the ODS fails, it will become impossible to reconcile the accounting activity back to the original source system transactions.

☞ TierBroker can synchronize the G/L to the ODS by performing multiple interface functions in the scope of a single database transaction.

TierBroker can load multiple systems simultaneously. Without this capability, complex programs must be written to create *compensating transactions,* which are used to correct errors when systems are not synchronized.

3.3. *Separation of Data Analysis and Accounting Logic*

Different Jobs, Different Roles

Creating automated accounting systems involves the collaboration of two very different types of resources. The **Data Analyst** develops the functional specifications for the transactions that will be mapped to the G/L. The **Accounting End User** specifies the business logic that will be used to generate sets of debits and credits.

☞ The Data Analyst owns the data definition and the data content, while the Accounting End User owns the business logic and accounting strategy.

Matching Technical Process to Business Process

TierBroker separates business event logic from the accounting strategy. Data Analysts maintain the Data Mapping and business logic for Data Transformation and Routing, while the Accountants maintain the Business Rules that select which sets of journal entries will be created for a given transaction type.

The Role of the Data Analyst

The Data Analyst uses TierBroker to define objects for Transactions, Journal Entries, Static Data and the Operational Data Store. The Data Analyst also takes requirements from the Accounting End User and creates Transformation Maps, which are templates for generating single Debit and Credit entries from specific transaction types. The Data Analyst specifies:

Data Definition – Objects which will be used to represent Transactions, Static Data and Debits and Credits.

Data Mapping – Transformation maps that will be used to compute individual debits and credits.

Process Workflow – The actual workflow that will be used to perform transformation and routing tasks such as computing of debits and credits, population of the transaction Operational Data Store, and posting of foreign GAAP adjustments.

The Data Analyst uses TierBroker to create a framework that the Accounting End Users will use to define the Accounting Strategy.

The Role of the Accounting End User

The Accounting End User does not use TierBroker directly. Instead, accounting strategies are specified using application panels built into the target G/L. It is not practical to teach accountants how to use TierBroker. Instead, PeopleSoft panels or other equivalent native G/L extensions can be constructed to implement the Accounting Strategy. The Accounting End User specifies:

Transaction Classification – Which features on a transaction will determine the types of journal entries required.

Journal Codes – Classes of Accounting Strategies, such as Trade Capture or Commission Sharing, which share a common set of Debit and Credit templates.

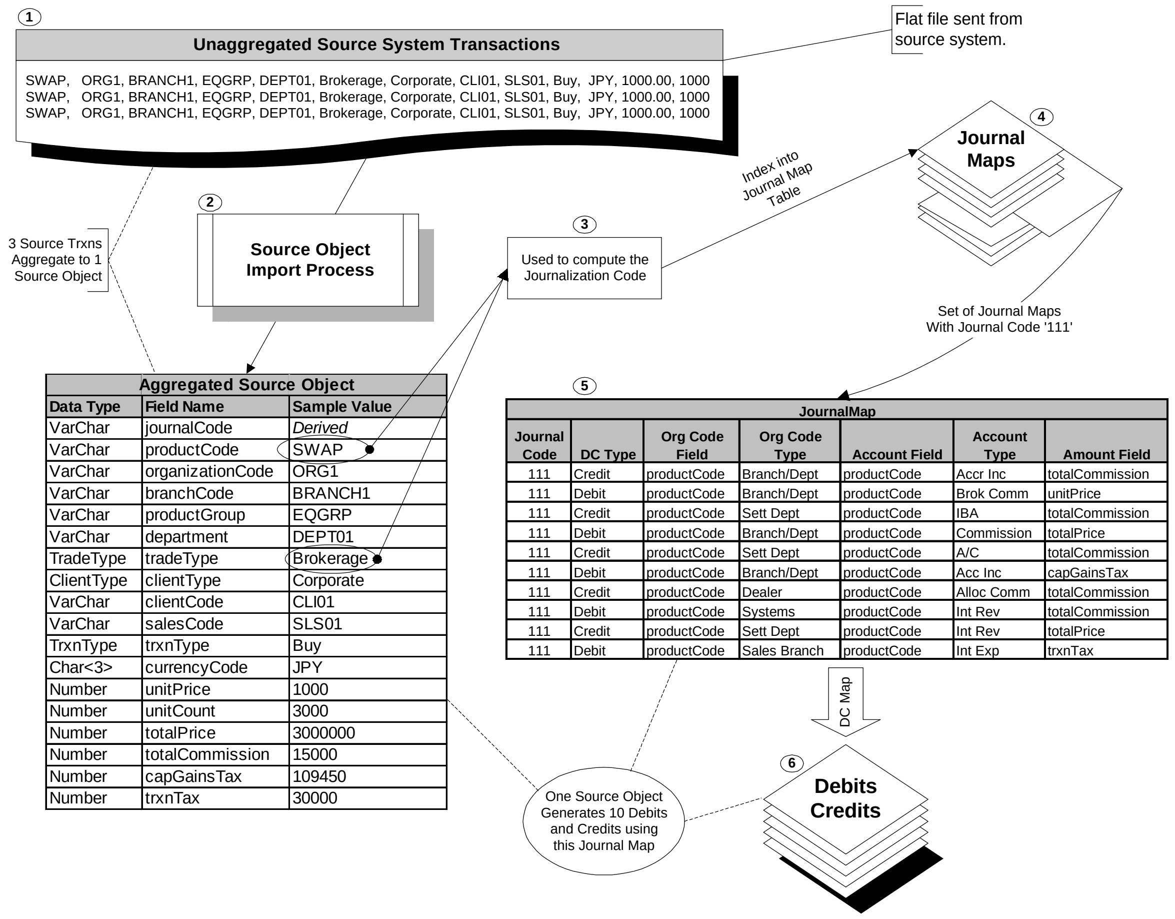
Journal Maps – Based on the Journal Code, which Debit and Credit transformation maps (created by the Data Analyst) are used to create a balanced journal entry.

The accountant uses tables that are external to TierBroker. These tables are maintained in the host ERP application in order to create a working environment and business process which does not require the participation of the Data Analyst. Through native connections to popular database packages as well as ODBC connections, TierBroker can use these tables to implement business logic at runtime.

3.4. Transaction Journalization Process Example

Swap Trade to Journal Entries

This simplified example reviews how TierBroker can be used in combination with external tables that define accounting strategies. This is not simply Content-Based Transformation, because the determination of the Journal Codes frequently involves complex functions or lookups to external data tables.


1
Unaggregated Source System Transactions
Flat file sent from source system.
2
Source Object Import Process
3 Source Trxns Aggregate to 1 Source Object
3
Used to compute the Journalization Code
Index into Journal Map Table
4
Journal Maps
Set of Journal Maps With Journal Code '111'
5
DC Map
6
Debits Credits
One Source Object Generates 10 Debits and Credits using this Journal Map

```
SWAP,   ORG1, BRANCH1, EQGRP, DEPT01, Brokerage, Corporate, CLI01, SLS01, Buy,  JPY, 1000.00, 1000
SWAP,   ORG1, BRANCH1, EQGRP, DEPT01, Brokerage, Corporate, CLI01, SLS01, Buy,  JPY, 1000.00, 1000
SWAP,   ORG1, BRANCH1, EQGRP, DEPT01, Brokerage, Corporate, CLI01, SLS01, Buy,  JPY, 1000.00, 1000
```

Aggregated Source Object

Data Type	Field Name	Sample Value
VarChar	journalCode	*Derived*
VarChar	productCode	SWAP
VarChar	organizationCode	ORG1
VarChar	branchCode	BRANCH1
VarChar	productGroup	EQGRP
VarChar	department	DEPT01
TradeType	tradeType	Brokerage
ClientType	clientType	Corporate
VarChar	clientCode	CLI01
VarChar	salesCode	SLS01
TrxnType	trxnType	Buy
Char<3>	currencyCode	JPY
Number	unitPrice	1000
Number	unitCount	3000
Number	totalPrice	3000000
Number	totalCommission	15000
Number	capGainsTax	109450
Number	trxnTax	30000

JournalMap

Journal Code	DC Type	Org Code Field	Org Code Type	Account Field	Account Type	Amount Field
111	Credit	productCode	Branch/Dept	productCode	Accr Inc	totalCommission
111	Debit	productCode	Branch/Dept	productCode	Brok Comm	unitPrice
111	Credit	productCode	Sett Dept	productCode	IBA	totalCommission
111	Debit	productCode	Branch/Dept	productCode	Commission	totalPrice
111	Credit	productCode	Sett Dept	productCode	A/C	totalCommission
111	Debit	productCode	Branch/Dept	productCode	Acc Inc	capGainsTax
111	Credit	productCode	Dealer	productCode	Alloc Comm	totalCommission
111	Debit	productCode	Systems	productCode	Int Rev	totalCommission
111	Credit	productCode	Sett Dept	productCode	Int Rev	totalPrice
111	Debit	productCode	Sales Branch	productCode	Int Exp	trxnTax

1. **Unaggregated Source System Transactions** are sent from various branches and departments. These transactions can be submitted in a variety of formats and can include trading, non-trading, cost accounting and other types of information necessary for creating the consolidated general ledger.
2. **The Source Object Import Process** converts these transactions into TierBroker objects. Different transaction types, record layouts and data formats are resolved in the import process. In the case given above, three source system transactions have aggregated to a single `SourceObject`.
3. **Fields in the Source Object are used to compute a *Journalization Code*.** Each Source Object determines its own *accounting strategy*, or journal code. Common journal codes include Revenue Allocation and Commission Sharing. The journal code is determined dynamically based on the contents of certain fields in that object.

4. **The Journal Map Table** is queried to determine the `JournalMap` entries for this `JournalCode`. Each row in the `JournalMap` table represents a single debit or credit.
5. **This `SourceObject` generates ten debits and credits** using the rules encoded in each row of the `JournalMap` table that shares this transaction's `JournalCode`.
6. **Debits and Credits** are generated and sent to the G/L.

Other G/L Requirements

This particular example does not review requirements such as balancing, post to suspense, reconciliation, and multiple GAAP reporting; it shows how TierBroker uses a data-driven approach to transaction journalization. To fulfill other G/L requirements using TierBroker, application logic can be added using Business Rules or callbacks to external languages.

4. Proprietary and Competitive Advantages

All statements regarding transaction processing measured in records per second were tested using an IBM 600 notebook computer running at 233Mhz with 64Mb of RAM.

4.1. XML and E-Business Leadership

The principal author of TierBroker, Adam Greissman, was a co-founder of **Financial products Markup Language (FpML™)** initiative. FpML is an XML markup language for automating the exchange of currency and interest rate derivatives between financial institutions. This work, proposed jointly with JP Morgan, was honored by Risk Magazine as the most important technological development of 1999. TierBroker will extend its deployment in the wholesale capital markets by integrating applications that can process FpML with core banking applications in the front, middle and back office.

4.2. High Speed and High Volume Capacity

TierBroker has a significant speed advantage compared to its competition. In most cases TierBroker is ten to one hundred times faster than competing products. In cases where TierBroker has been selected in global financial institutions, a key factor was the inability of alternative technologies to manage the required data volumes and throughput.

TierBroker typically achieves database read/write throughput in the range of several thousand transactions per second. Most file operations, such as Comma Separated Value (CSV) and other forms of flat file parsing, occur at rates above 5,000-10,000 records per second.

4.3. Dynamic Message Grouping

One problem with real time environments is the overhead of the data transport layer wrapped around each message. Network and message latency is one reason that products using messaging middleware such as IBM's MQ Series rarely achieve throughput above 100 records per second. TierBroker can perform dynamic message grouping, which can increase transaction throughput using MQ Series above 20,000 messages per second — 200 times faster than competing technologies.

TierBroker achieves this performance, because its effective throughput is not gated by the per-message speed of the Messaging Middleware. As individual messages become larger, the realized throughput of MQ Series approaches some fraction of the network bandwidth as its theoretical limit. In a BroadVision application built for PwC in 1999, message packets of approximately 100-300 bytes were used for a Dynamic Grouping test case. These messages were used to simulate eventing in a browser application. In this case, the observed throughput was approximately 20K message packets per second.

Observed Throughput in Other Client/Server Applications

In other client/server applications, TierBroker routinely delivers very high rates of throughput. For example, with SQL servers that allow array processed transactions (Oracle, Kdb), throughput as high as 10,000-15,000 records per second for Oracle and 150,000 records per second for Kdb are routinely observed for records with less than 20 columns.

With larger records in the 60 column range, database insert times greater than 1,000/rps and select times of 2,000/rps are observed with Oracle. The corresponding numbers in Kdb are 15,000 to 20,000 records per second. Performance with TierBroker is gated by a combination network bandwidth and the speed of the servers it is connecting to. The time required to perform a transformation or marshal data is almost never a gating factor, because that time is at a much lower level of granularity than the network overhead.

4.4. Small System Footprint

The TierBroker engine is less than 2MB in size. This can be scaled to fit in less than 1MB for deployment in embedded systems. The size of TierBroker also makes it feasible as a client side application deployed over the Internet.

4.5. Native XML Parsing and Validation

TierBroker is an XML-based tool. Unlike legacy middleware technologies that impose rigid adapters on the XML semantic model, TierBroker uses XML as the basis of its object definition and business rules processing engine. TierBroker's validating XML parser can parse and validate XML at speeds greater than 100,000 elements per second, yielding useful transaction throughput of approximately 5,000 records per second. TierBroker can write or stream XML at roughly the same rate of throughput. It can also process schema information contained in XML documents.

4.6. Extensive Connections and Adapters

SQL Databases

TierBroker has native drivers for SQL databases such as Sybase, Oracle, Informix and MS SQL Server. Under Windows NT, it can also use ODBC to access other RDBMS such as DB2 and MS Access.

Flat File Formats

TierBroker supports complex file formats that are character- or position-delimited, including those with records spread out over multiple lines. TierBroker is ideal for processing complex file structures with mixed record types such as SWIFT messages or SAP I-Docs.

Messaging Middleware

TierBroker has native drivers for raw TCP/IP, COM and MQ Series and has been deployed with other messaging middleware technologies such as CORBA. TierBroker also has the capability to work with scheduling, error reporting and transaction process monitor (TPM) applications.

Spreadsheets

TierBroker's ability to read and write spreadsheet formats gives it the capability to define test suites involving hundreds of trials for valid records and error handling. Testing approaches that require records to be tested one at a time (Neon) or through database tables (Informatica), are not as robust; in general, testing procedures using these competing products are more difficult to repeat and cannot be fully automated for the purposes of reconciliation.

Mainframes and Host Connections

TierBroker’s ability to read and write raw TCP/IP has allowed it to be deployed with mainframe applications that have proprietary host protocols.

4.7. General Ledger (G/L) Posting Engine

TierBroker has been used to implement complex accounting applications that take transactions and convert them into debits and credits. Accounting logic and business rules for journal posting, balancing, reconciliation and post to suspense have all been successfully developed for Enterprise Resource Planning (ERP) applications such as SAP and PeopleSoft.

5. Frequently Asked Questions

5.1. The Universal Data Interface Corporation (UDICo)

How old is UDICo?

UDICo was formed in March 2000 to commercialize middleware technology developed by the software author at PricewaterhouseCoopers from 1993-2000.

What is the relationship between PwC and UDICo?

UDICo is an independent software company. PricewaterhouseCoopers is an equity investor.

5.2. TierBroker

What is TierBroker?

TierBroker is middleware: computer software that allows different applications to exchange data in batch mode and in real time.

Where has TierBroker been used?

TierBroker was developed at PricewaterhouseCoopers over a seven year time period (1993-2000). Before it was available as a commercial software product from UDICo, PwC used it to implement some of the most complex systems in capital markets. Major financial institutions use TierBroker in production because it can process large volumes of complex information.

Why did PwC clients select TierBroker over third party products?

When in use at PricewaterhouseCoopers, TierBroker was marketed as leveraged software. Many of their clients selected TierBroker over many well-known global middleware brands because of its ability to process large volumes of data at high rates of throughput. In most of these cases, the clients did not have a choice between TierBroker and other applications, because existing middleware products could not satisfy the rigorous demands of global data processing for financial institutions.

They chose TierBroker because the only other choice was to build the entire middleware platform from the ground up. In other cases, including a sales force automation implementation for a major insurance company, TierBroker was selected because middleware adapters supplied by other vendors did not live up to the marketing claims made for features and performance.

How can TierBroker be as fast as you claim?

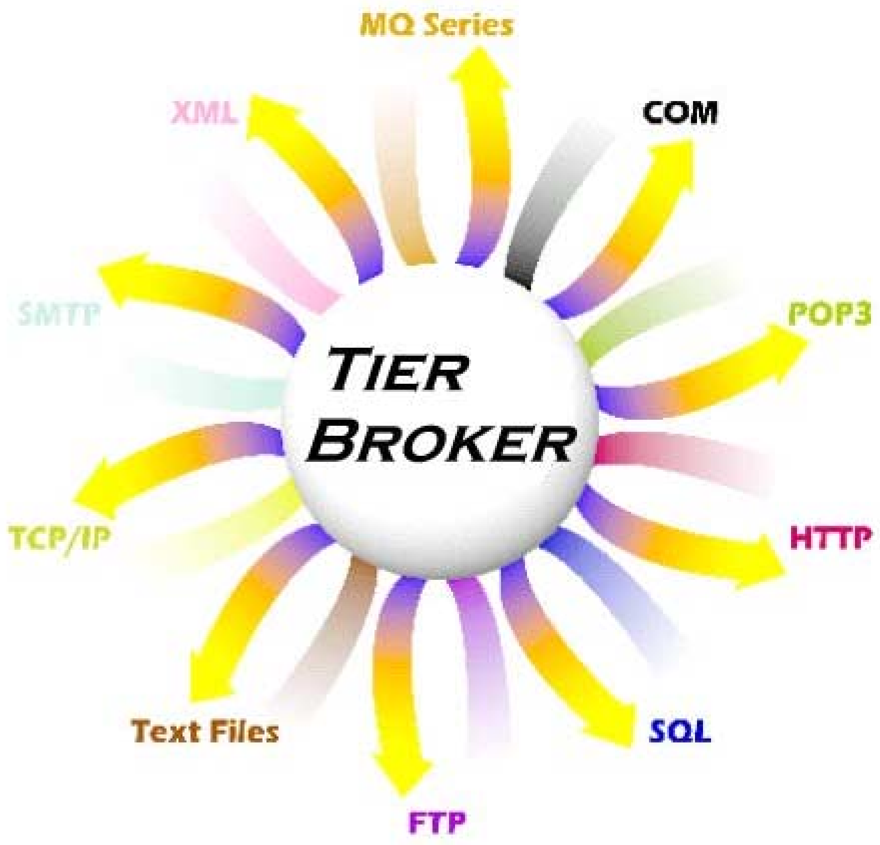
Several features contribute to TierBroker's speed:

- The TierBroker Runtime Engine is written in C++, making it much faster than any Java-based solution.
- This C++ code uses a combination of proprietary techniques and algorithms that make it even faster than typical C++ programs addressing the same problems.

- While TierBroker can easily use an existing messaging infrastructure, it has no built-in dependence on one. This removes a common source of processing time overhead.
- In addition to ODBC support, TierBroker has native support for Oracle, Sybase, Informix, and SQL Server, making communication with the most popular database management products much faster and more efficient than going through an ODBC layer to reach them.

It's "XML-based." Does it only work with XML data?

TierBroker can easily read and write character-delimited files, position-delimited files, SQL, COM, TCP/IP, HTTP, SMTP, POP3, and MQ Series, and of course, XML. The list is growing the time.



While the flexibility of many XML-based middleware solutions is based on their ease of converting XML to other formats, TierBroker makes it very easy to get other formats *into* XML, which is often 80% of the battle.

What about W3C Schema support?

The W3C's schema specification gives XML system developers many advantages that are unavailable when using XML 1.0 DTDs—especially the ability to specify data types for elements and attributes. Once this spec becomes a W3C Recommendation, TierBroker will support it as a data definition language. Until then, it's a moving target.

What's an adapter?

Any add-on to the basic TierBroker product. This may range from a set of object class definitions for a popular data format such as FpML or SOAP to custom DLLs written to accommodate specific client needs.

Can TierBroker work with any DTD or Schema? How are new adapters developed?

One of the TierBroker IDE’s key strengths is the ease with which it lets you create object class definitions for your own system's data structures. In many cases, you can just import these definitions from an existing DTD, SQL SELECT statement, Microsoft Query file, or database table definition; for more complicated cases, the TierBroker scripting language lets you specify more complex relationships with fine-grained control over data relationships and types.

For coding new features to incorporate into your middleware, TierBroker's support of COM and TCP/IP (with Java JNI/JMS support underway) let you incorporate your own routines in the language of your choice.

Does middleware simulation really work?

In 1995 Price Waterhouse implemented SAP for a major Japanese securities firm. All of the development and testing was performed on MS Windows 3.1, although the target operating system was HP/UX. The UNIX development was installed just two months prior to production. That system runs in production today and processes more than 1.25 million trades a day from more than 130 different source systems.

How can TierBroker use accounting logic in PeopleSoft to convert transactions into journal entries?

TierBroker separates business event logic from the accounting strategy. Data analysts maintain the data mapping and business logic for data transformation and routing, while the accountants maintain the business rules that select which sets of journal entries will be created for a given transaction type.

6. UDICo Contact Information

Questions about this document may be addressed to the following UDICo officers:

Adam Greissman
Chief Executive Officer
(212) 607-7621 – ALG@udico.com

Simon Smith
Chief Financial Officer
(212) 607-7626 – Simon.Smith@udico.com

Universal Data Interface Corporation
95 Wall Street, 21st Floor
New York, NY 10005
Fax: (212) 607-7620

info@udico.com