Oracle® Tuxedo

Overview Guide

Release 22c (22.1.0.0.0)

F60931-03

Oracle Tuxedo Overview Guide, Release 22c (22.1.0.0.0)

F60931-03

Primary Author: Preeti Gandhe

Contributing Authors: Tulika Das

Contributors: Maggie Li

Contents

2 Oracle Tuxedo ATMI Core Components

Preface

This guide provides an overview of the Oracle Tuxedo Product Family.

Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc.

Access to Oracle Support

Oracle customers that have purchased support have access to electronic support through My Oracle Support. For information, visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info or visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs if you are hearing impaired.

1
Introducing Oracle Tuxedo

The following sections describe the architecture and major features of the Oracle Tuxedo product:

1.1 A Brief History of the Tuxedo System

Oracle Tuxedo is a proven, mature system spanning over 30 years of continuous development and enhancement:

- In 1983, the Tuxedo system began as an applied, forward-looking work project within the Bell Laboratories division of AT&T. The target applications for the Tuxedo system were UNIX-based operations support systems within AT&T.
- In 1989, the Tuxedo system was transferred to the UNIX System Laboratories (USL) division of AT&T, and its client/server framework was offered as a commercial product.
- In 1993, the Tuxedo system was transferred to Novell, Inc., when Novell acquired USL in 1993.
- In 1996, BEA Systems, Inc., entered into an exclusive agreement with Novell to distribute and continue development of the Tuxedo system on a variety of computer platforms, including Windows and most UNIX systems.
- In 2008, Tuxedo became Oracle product along with all other BEA Systems products.

1.1.1 Releases 1.0 Through 7.1

From release 1.0 in 1983 through release 7.1 in 2000, the Tuxedo system was extended and enhanced in a number of significant ways, always with the intent of making communication between client and server processes easier and more flexible. The Tuxedo system evolved into the de facto standard for open (open standard) online transaction processing (OLTP) solutions.

Release 4.0 introduced the *ATMI API* and *transaction processing*. Release 5.0 introduced the *Domains component*, which provided for the federation of Tuxedo applications and inter-application transaction processing. Release 7.1 introduced a *security plug-in architecture*, which allowed for the installation of third-party security systems.

Release 7.1 also introduced *multithreading* and *multicontexting*—ATMI functions that enabled programmers to write multithreaded and/or multicontexted application clients and servers—and *XML buffer support*—the ability to use extensible markup language (XML) typed buffers to exchange XML data within and between ATMI applications. In release 7.1, the Oracle Jolt product was bundled with Oracle Tuxedo for the first time.

For an overview of Oracle Tuxedo ATMI, see Oracle Tuxedo ATMI Core Components

1.1.2 Release 8.0

Release 8.0 introduced the Oracle CORBA API and CORBA Object Transaction Monitor (OTM) capability. The CORBA OTM combined the advantages of a COBRA-compliant

programming model with the proven power and reliability of the Oracle Tuxedo core technology infrastructure.

For an overview of Oracle Tuxedo CORBA, see Oracle Tuxedo CORBA Components

1.1.3 Release 8.1

Release 8.1 introduced the following features and enhancements:

- Localization enhancements
  Enables customers to install and interface with the Tuxedo system in English or Japanese.
- Multibyte character encoding
  Provides a new ATMI application typed buffer to handle multibyte character encoding.
- XML parser integration
  Incorporates the Apache Xerces C++ Version 1.7 parser into Tuxedo for use by customer applications to read and write XML data.
- Single point security administration
  Enables customers to use the Oracle WebLogic Server Administration Console to administer security for both Oracle Tuxedo and Oracle WebLogic Server.
- Domain gateway performance improvement
  Improves the performance of the Tuxedo domain gateway process without any changes in the user interface.
- Remote domain connection policy
  Changes the behavior of the `ON_STARTUP` type connection policy of the Tuxedo domain gateway process to allow customers to selectively establish connections on a per remote domain basis.
- Domains keepalive
  Keeps interdomain connections open through firewalls during extended periods of no application activity and enables the Tuxedo domain gateway process to quickly detect interdomain connection failures.
- Multithreaded bridge
  Allows users to configure the Tuxedo Bridge process for multithreaded execution (as opposed to single-threaded execution) to improve Bridge performance.
- Parameter length expansion
  Increases the maximum allowable length of certain Tuxedo configuration parameters from 64 or 78 characters to 256 characters.
- Global maximum transaction timeout
  Adds a global maximum transaction timeout parameter to cap ATMI transaction timeout values that are excessively long.
- Enhanced CORBA C++ client ORB
  Enables CORBA C++ clients to participate in global transactions with WebLogic Server application servers in the same way that WebLogic Server T3 clients do.

In addition, both the Oracle Jolt product and the Oracle SNMP Agent product are bundled with Oracle Tuxedo 8.1.

1.1.4 Release 9.0

Release 9.0 introduced the following features and enhancements:

- Enhanced Web Services
  Provides XML schema and transformation (XML to and from FML) support. Also provides a Tuxedo service metadata repository that provides access to Tuxedo service definitions. It is designed to process interactive queries by developers and administrators during application development or modification, and is not designed for the processing of high volumes of automated queries during the application production phase.
- Domain Gateway performance improvements
- Infrastructure improvements in the following areas:
  - Timeout controls
  - Domain connection policies
  - CORBA IIOP client failover.
- Security
  - Cert-C PKI Plug-in security for data protection and non-repudiation
  - Kerberos authentication support
- Tuxedo .NET Workstation Client support
  The Tuxedo .NET workstation client provides customers with access to the Tuxedo system using the .NET Framework environment. It is implemented as a set of APIs and development utilities for developers.

In addition, both the Oracle Jolt product and the Oracle SNMP Agent product are bundled with Oracle Tuxedo 9.0.

1.1.5 Release 10.0

Release 10.0 introduced the following new features and enhancements:

- Oracle Tuxedo System and Application Monitor (TSAM) Agent
  Oracle TSAM provides comprehensive monitoring and reporting for Oracle Tuxedo system and applications. It includes two components: Oracle TSAM Agent and Oracle TSAM Manager.

  The Oracle TSAM Agent enables collection of various performance metrics for applications, including XA and non-XA transactions, services, system servers. The Oracle TSAM Agent can be used in conjunction with the Oracle TSAM Manager.

  The Oracle TSAM Manager provides a graphical user interface to correlate and aggregate performance metrics collected from one or more Tuxedo domains. It displays the information in *real time*.
- SSL Support for ATMI applications
  This feature provides support for SSL encryption over all network links in Tuxedo where LLE encryption is available. For more in formation, see Introducing ATMI Security, in "*Using Security in ATMI Applications*."
- MQ Adapter
  The MQ Adapter provides bi-directional, transactional connectivity to and from WebSphere MQSeries. For more information, see Running the Tuxedo MQ Adapter.

- Generic AUTHSVR
  Generic AUTHSVR is a new Tuxedo system server (GAUTHSVR) that enables Tuxedo users to be authenticated with LDAP based directory servers without need to write custom code. For more information, see Implementing Single Point Security Administration, in "*Using Security in ATMI Applications*."
- DoS
  Provides Tuxedo TDomain domain gateway features used to defend against DoS attacks, and Tuxedo Domain improved password pair configuration flexibility. For more information, see Introducing ATMI Security in "*Using Security in ATMI Applications*."
- Integrating ACUCOBOL in buildclient/buildserver
  In Tuxedo 10.0, buildclient/buildserver can accept COBOL source files and generate C stub code automatically using ACUCOBOL compiler version 6.2.0 or above.
- OpenLDAP for X.509 Certificate Lookup.
  Tuxedo 10.0 PKI plug-in added support for OpenLDAP for X.509 certificate lookup. For more information, see Administering Security,Cert-C PKI Encryption Plug-In Configuration, and Configure Certificate Lookup in Using Security in ATMI Applications.

1.1.6 Release 10*g* Release 3 (10.3)

Release 10*g*R3 introduced the following new features and enhancements:

- IPv6 Support
  IPv6 is the next generation protocol designed by the IETF to replace the current version Internet Protocol, IP Version 4 (IPv4).

  The most obvious improvement in IPv6 over the IPv4 is that IP addresses are lengthened from 32 bits to 128 bits. It also adds many improvements to IPv4 in areas such as routing and network autoconfiguration.

  For more information on using IPv6 with Oracle Tuxedo 10*g*R3, see Enabling IPv6 in the *Oracle Tuxedo Programming Guide*.
- Application-Created Context in ATMI Server
  Two new APIs, tpappthrinit(3c) and tpappthrterm(3c) are provided for application-created thread in ATMI server to create and terminate separate Tuxedo context. In context created using tpappthrinit(3c), the application-created server thread can initiate service requests and define transactions.

  For more information, see Programming a Multithreaded and Multicontexted ATMI Application in the *Oracle Tuxedo Administration Guide*.
- Oracle Tuxedo Access Log
  Assists Tuxedo client administrators to monitor application validity at runtime. You can record application high water client count, current client count, and named users.
- Enhancements
  - CLOPT length
    Increased from 256 to 1024.
  - FML field length
    Increased from 30 to 254.
  - tlisten password encryption

tlisten.pw file is system-encrypted. To change password, you must use tlistpwd(1).

  - Dynamic DMIB Update
    Allows re-configuring the listening address of the remote domain gateway without shutting down the local domain.
  - Domain Gateway Persistent Disconnect
    Local domain with a PERSISTENT_DISCONNECT connection policy will neither connect to nor accept connect request from any remote domain.

1.1.7 Release 11*g* Release 1 (11.1.1.1.0)

Release 11*g*R1 introduced the following features and enhancements:

- Client/Server Affinity
  The Oracle Tuxedo Client/Server Affinity feature provides the flexibility to set up a simple session-aware application model. It creates a "virtual" request routing scope using the Oracle Tuxedo ATMI RPC infrastructure. When a session is established, all subsequent calls are impacted by the routing scope until the session is terminated (explicitly or implicitly). With Client/Server Affinity, you can retain session context resources inside the client/server affinity scope.
- Extended ATMI Service Name Length
  The maximum Oracle Tuxedo ATMI service name length is increased to 127 characters.
- Enhancements
  - Domain retries connection establishment after incorrect password failure
    When configured, `ON_STARTUP` domain gateway continuously re-tries to establish connection to remote domain when domain password pair validation fails.
  - Flexibility to run Oracle Tuxedo as user other than administrator on Microsoft Windows
    If configured in Microsoft Windows, processes booted in an Oracle Tuxedo domain are owned by the user who executed the `tmboot` command, instead of the user who starts the `TUXIPC` system service.

1.1.8 Release 11*g* Release 1 (11.1.1.2.0)

Release 11*g*R1 (11.1.1.2.0) introduced the following new features and enhancements:

- Oracle Tuxedo Group Multiple Resource Managers
  Oracle Tuxedo now supports multiple Resource Managers (RMs) in one group, so every group application server has the ability to communicate with multiple RMs in one global transaction. For more information, see the Oracle Tuxedo ATMI COBOL Function Reference, Oracle Tuxedo ATMI C Function Reference, Oracle Tuxedo File Formats, and Data Descriptions, MIBs, and System Processes Reference.
- Nested Views
  Previous Oracle Tuxedo releases have supported views. In Release 11.1.1.2.0 view functionality is extended to support nested views. For more information, see Managing Typed Buffers in Programming an Oracle Tuxedo ATMI Application Using C.

1.1.9 Release 11*g* Release 1 (11.1.1.3.0)

Release 11*g*R1 (11.1.1.3.0) introduced the following new features and enhancements:

- Oracle Tuxedo Exalogic Improvements

- Self-tuning Locking Mechanism
  Allows you to dynamically tune SPINCOUNT while taking the runtime environment into consideration, thus improve performance especially when there are heavy load on the system using XA without administrator configure SPINCOUNT with a static value. For more information, see Oracle Tuxedo/Oracle Exalogic Users Guide and File Formats, Data Descriptions, MIBs, and System Processes Reference in the Oracle Tuxedo Reference Guide.
- Direct Cross Node Communication Leveraging RDMA
  In Oracle Tuxedo applications, processes on separate machines communicate with each other through bridge processes using a socket. The communication between bridges can be considerably slow. This feature utilizes RMDA through which processes on separate Exalogic machines can communicate with each other directly. If Oracle Tuxedo application processes use RDMA instead of bridge processes overall performance is improved. For more information, see Oracle Tuxedo/Oracle Exalogic Users Guide
- SDP Support in Oracle Tuxedo
  This feature allows Oracle Tuxedo users to configure which protocol to use, either SDP or normal TCP, between Oracle Tuxedo components, including Domain gateware, bridge, work station client and WSH, Jolt client, and JSH t to leverage the advantages Exalogic provides such as high bandwidth, low latency as well as reduced CPU involvement. For more information, see File Formats, Data Descriptions, MIBs, and System Processes Reference, Command Reference, and ATMI C Function Reference in the Oracle Tuxedo Reference Guide. Also, Configuring the Oracle Jolt System in Using Oracle Jolt in the Oracle Tuxedo Users Guide.
- TLOG Information To Oracle Database
  Allows you the flexibility of using an Oracle database instead of file system to store the TLOG. You can also leverage various high availability Oracle database features in disaster recovery as needed. For more information, see About Transactions in Setting Up an Oracle Tuxedo Application in the Oracle Tuxedo Users Guide, File Formats, Data Descriptions, MIBs, and System Processes Reference ,and Command Reference in the Oracle Tuxedo Reference Guide.

1.1.10 Release 12*c* Release 1 (12.1.1)

Release 12c Release 1 (12.1.1) introduces the following new major features and enhancements:

- Optimizations for Oracle Exalogic
- Use of Shared Memory for Inter Process Communication
  Oracle Tuxedo 12c significantly enhances performance of Tuxedo applications on Exalogic with use of shared memory queues instead of IPC Message Queues for inter process communication on the same Tuxedo node. With the use of shared memory queues, the sender and receiver processes can exchange pre-allocated messages in shared memory, thus eliminating the need to copy messages several times before message reaches its intended target and resulting in much better throughput and lower latency.

  For more information, see How to Create TUXCONFIG File in Administering an Oracle Tuxedo Application at Run Time, ATMI C Function Reference, File Formats, Data Descriptions, MIBs, and System Processes Reference, and Command Reference.
- Shared Applications Staging

With Oracle Tuxedo 12c, one can share application directory (APPDIR) among many compute nodes of the storage appliance on an Exalogic system, making it easier to manage application deployment. For more information, see the configuration of `UBBCONFIG` in File Formats, Data Descriptions, MIBs, and System Processes Reference.

- Read-Only Optimization for XA
  Optimized Distributed Transaction processing within and across Tuxedo domains for read-only transaction, including global transaction across Tuxedo domain and WTC(in WLS 12.1.1 - Contact Oracle Support for a patch, or higher release of WLS). One of the typical scenarios is every branch of the global transaction access the same Oracle Database instance. In order to use this feature with WebLogic Server, minimum patch requirement for WebLogic Server must be met. This feature is not supported for CORBA applications. For more information, see the `RESOURCE` and `T_DOMAIN` sections in File Formats, Data Descriptions, MIBs, and System Processes Reference.
- Tightly Coupled Transaction Branches Crossing Domain
  Common global transaction identifier (GTRID) is introduced in default to make branches within a global transaction crossing domains using common GTRID. The branches would be tightly coupled if they are running on same database (if the database allows).For more information, see Oracle Tuxedo on Exalogic Users Guide.
- Application Packaging and Deployment
  Oracle Tuxedo 12c introduces a new concept of application packages. An application package is self-contained deployable unit. Application packages contain application binaries and required configuration artifacts, and can be automatically deployed/ undeployed to an already running Tuxedo application domain. Application packaging and deployment feature includes in this release provide infrastructure for private cloud applications.

  For more information, see Oracle Tuxedo Application Packaging and Deployment in Setting up an Oracle Tuxedo Application.
- Developing New Applications using Java
  With this feature, one can develop new Tuxedo services using Java programming language in order to extend existing C/C++/COBOL applications. Java services are deployed and coexist in the same container as C/C++/COBOL services, allowing one to manage and monitor applications written in different programming languages using same set of tools. Coexistence in the same container also optimizes transaction coordination across services written in different languages.

  Java services development includes following major features in the 12c release:

  – POJO programming model
  – JATMI based API
  – XA transactions
  – Monitoring and management - equivalent to C/C++/COBOL services

For more information, see Oracle Tuxedo Java Server and Java Programming.

- IBM WebSphere MQSeries Adapter Features
  Following new features are added to the IBM WebSphere MQSeries Adapter:

  – Reduced CPU usage, better throughput and higher scalability through multithreaded and event driven architecture of TM_MQI server
  – Automatic connection failure recovery for TM_MQI and TM_MQO servers
  – Support for clustered queues
  – Ability to connect to remote MQSeries Manager

  - Access to MQSeries message headers
  - Recoverable messages in case security failure

  For more information, see File Formats, Data Descriptions, MIBs, and System Processes Reference, Command Reference, and Tuxedo MQ Adapters.

- Service Versioning
  This feature provides a configuration-driven way to deploy different versions of Tuxedo services in an application domain or across domains without changing the existing code. By use of version, one can logically partition the existing Tuxedo applications into different virtual application domains, machines, and server groups based on current Tuxedo management hierarchy, so as to respond to several of special business access logics and on the other hand satisfy upgrade requirements in non-stop mode.

  For more information, see Applying Service Version to Tuxedo Applications in Setting up an Oracle Tuxedo Application.

- High Availability Configuration for Data Dependent Routing
  With this feature, multiple server groups can be configured for the same data range, thus allowing incoming requests to failover to an alternate group, if primary group is not available, increasing the availability of the application. This feature can be used either within a domain or across the domains.

  For more information, see ROUTING Section in File Formats, Data Descriptions, MIBs, and System Processes Reference.

- Use of XPath for XML based Data Dependent Routing
  With this feature, one can use XPath for much greater flexibility in specifying routing criteria if XML buffer type is used for data dependent routing.

  For more information, see the configuration of `UBBCONFIG` and `TM_MIB` in File Formats, Data Descriptions, MIBs, and System Processes Reference.

- Generic LDAP Authentication/Authorization Framework
  Oracle Tuxedo 12c provides a flexible authentication and authorization framework that can be used to store credentials and Access Control Lists (ACLs) in LDAP or another 3rd party framework.

  Generic LDAP authentication and authorization framework includes following major features:

  - LDAP based authentication and authorization
  - Flexible LDAP schema support
  - Nested group support for authorization

  For more information, see How to Enable Generic LDAP Based Security in Using Security in ATMI Applications.

- Expedited Diagnostics through ECID Propagation
  With this feature, an ECID (Execution Context ID) is propagated with each request within Tuxedo and across various products in Oracle stack. Propagation of ECID allows easy correlation of requests across Tuxedo domains and Oracle products, such as WebLogic Server, Database and so on, making it faster easier to diagnose application problems.

  For more information, see Configuring Tuxedo for Propagating ECID in Setting up an Oracle Tuxedo Application.

- Automatic Master Node and Server Group Migration

This feature enables automatic migration of master node to designated back up without any manual intervention, thus minimizing the application downtime and increasing the availability. Similarly this feature also enables automatic migration of server groups.

For more information, see Migrating Your Application in Administering an Oracle Tuxedo Application at Run Time.

- Millisecond Granularity for Timeouts
  This feature introduces millisecond granularity for various Tuxedo timeouts, and other configuration parameters, such as for SCANUNIT. Millisecond granularity allows faster cleanup, restart, and migration of failed servers and nodes as well as faster transaction timeouts, enabling enforcement of tighter service level agreements, such as in algorithmic trading applications. For more information, see ATMI C Function Reference, ATMI COBOL Function Reference, and File Formats, Data Descriptions, MIBs, and System Processes Reference.
- Cross Domain Event Broker
  This feature allows subscribe, unsubscribe, and post of brokered events across Tuxedo domains as is done in a local Tuxedo domain. For more information, see Subscribing to Events in Administering an Oracle Tuxedo Application at Run Time.
- Server Side Pseudo Code from Service Definition
  If a service definition is in Tuxedo Metadata Repository, this definition can be used to generate server pseudo code in 'C' programming language using `tmunloadrepos` command. Server side pseudo code is generated in addition to client pseudo code, which can be done in prior releases.

  For more information, see `tmunloadrepos` in File Formats, Data Descriptions, MIBs, and System Processes Reference.
- Nested Views for Jolt
  Nested Views for Jolt are now supported in the 12c release.

  For more information, see Creating the Oracle Tuxedo Service Metadata Repository in Setting up an Oracle Tuxedo Application, and Using Oracle Jolt
- New Programming Model
  Oracle Tuxedo 12c now includes a new programming model that makes it extremely simple to develop new Tuxedo applications in C++. The programming model, based on SCA and originally released in SALT 10gR3, is now part of Tuxedo installation and customers upgrading to Tuxedo 12c can use the new programming model without need for any additional product.

  For more information, see Service Component Architecture.
- Support for Dynamic Languages (PHP, Python, and Ruby)
  Oracle Tuxedo 12c now includes framework to develop Tuxedo services in PHP, Python, and Ruby dynamic languages. The framework originally released in SALT 10gR3 is now included in Tuxedo 12c installer and customers upgrading to Tuxedo 12c release can use this framework to develop Tuxedo services in these dynamic languages. The framework also includes client API for these languages and Apache Web server plugin to allow development of Web applications accessing Tuxedo services on the backend.

  For more information, see Service Component Architecture.

1.1.11 Release 12c Release 2 (12.1.3)

Release 12c Release 2 (12.1.3) introduces the following new major features and enhancements:

- Optimizations for Oracle Exalogic/SPARC SuperClusterj
  - XA Affinity for Transactions
  - Common XID
  - Single Group Multiple Branch (SGMB)
  - FAN Integration
  - Direct Cross Domain Communication Leveraging RDMA
  - Other Optimizations

  For more information, see Oracle Tuxedo/Oracle Exalogic Users Guide.
- XA Affinity for Transactions
  XA affinity provides the ability to route all database requests within one global transaction to the same Oracle RAC instance; no matter if the requests come from an Oracle Tuxedo application server or Oracle WebLogic Server. This feature can reduce the cost of redirecting database requests to a new Oracle RAC instance, and thus can improve overall application performance.

  For more information, see Using Tuxedo with Oracle Real Application Clusters (RAC).
- Common XID
  With the common XID (transaction branch identifier) feature in this release, Tuxedo shares the XID of the coordinator group with all other groups within the same global transaction. This is as opposed to each group having its own XID and thus requiring two-phase commit in earlier releases if multiple groups are participating.

  Common XID eliminates the need to XA commit operations for groups that connect to the same Oracle RAC instance through the same service by using the coordinator branch directly.

  In other cases, where all groups in a global transaction use the coordinator branch directly, one-phase commit protocol is used instead of two-phase commit protocol, and thus avoid writing to `TLOG`.

  For more information, see Using Tuxedo with Oracle Real Application Clusters (RAC).
- Single Group Multiple Branch (SGMB)
  This feature eliminates the need to use singleton RAC service when multiple servers in a server group participate in the same global transaction. If servers in the same server group and same global transaction happen to connect to different RAC instances, a different transaction branch is used. This enables such applications to perform load balancing across available RAC instances.

  For more information, see Using Tuxedo with Oracle Real Application Clusters (RAC).
- FAN Integration
  FAN (Fast Application Notifications) are events published by Oracle RAC to indicate configuration changes. A system server, `TMFAN`, is introduced to monitor FAN events and automatically reconfigure Tuxedo server connection to the appropriate Oracle RAC instance for planned DOWN events, UP events, LBA (Load Balancing Advisor) notifications, etc.

  For more information, see Using Tuxedo with Oracle Real Application Clusters (RAC).

- Direct Cross Domain Communication Leveraging RDMA
  This feature enables application in a Tuxedo domain to directly communicate with another application in another domain using RDMA technology on Exalogic. Applications can directly access IPC queues of remote applications removing `GWTDOMAIN` as a potential bottleneck, thus reducing the latency and improving throughput and scalability.

  For more information, see Direct Cross-Domain Communication Leveraging RDMA on Exalogic.

- Other Optimizations
  The following optimizations are supported on both Linux platforms on Oracle Exalogic and Solaris platforms on SPARC SuperCluster.

  – Self-Tuning Lock Mechanism
  – Oracle Tuxedo SDP Support
  – Use of Shared Memory for Inter Process Communication
  – Read-Only Optimization for XA
  – Shared Applications Staging
  – Tightly Coupled Transaction Branches Crossing Domain

- TLOG Information to Oracle Database
  This feature enables creation of TLOG in Oracle Database. This feature is now available on all platforms supported in Oracle Tuxedo 12c Release 2 (12.1.3).

  For more information about this feature, see About Transactions in Setting Up an Oracle Tuxedo Application in the Oracle Tuxedo Users Guide, File Formats, Data Descriptions, MIBs, and System Processes Reference and Command Reference in the Oracle Tuxedo Reference Guide.

- Diagnostic Tool
  Diagnostic tool provides a convenient way for system administrators to collect Oracle Tuxedo system runtime information and store it locally. You can choose to provide this information to Oracle support as this information can be very helpful to Oracle support engineers in analyzing Oracle Tuxedo system issues, especially for issues that can only be reproduced in production environments.

  For more information, see About Oracle Tuxedo Diagnostic Tool.

- Enhancements for `tpacall()`
  Oracle Tuxedo increases the maximum number of outstanding `tpacall()` to 2048, making it possible to invoke more `tpacall()` before `tpgetrply()`.

- Oracle Entitlements Server (OES) Integration
  Oracle Tuxedo integrates with Oracle Entitlements Server (OES), allowing organizations to protect resources by defining and managing policies that control access to/usage of these resources. For more information, see File Formats, Data Descriptions, MIBs, and System Processes Reference and ATMI C Function Reference in the Oracle Tuxedo Reference Guide, and How to Enable Security Service for OES.

- Oracle Tuxedo Plug-in for Oracle Virtual Assembly Builder
  Oracle Tuxedo Plug-in for Oracle Virtual Assembly Builder supports Oracle Tuxedo Application Runtime for IMS and the following Oracle VM Server in this release:

  – Oracle VM Server for x86
  – Oracle VM Server for Exalogic

  For more information, see Oracle Tuxedo Plug-in for Oracle Virtual Assembly Builder.

- Generating a Java Application with the eGen Application Generator
  The eGen utility maps a COBOL copybook into a Java class. The specified COBOL copybook is parsed a Java source file is generated. This utility can also create corresponding DTD and XML schema if XML support is needed. The generated artifacts can be used from a Java application, whether running within Tuxedo, in another application server or as a standalone application to access COBOL services hosted in Tuxedo or on mainframes within CICS/IMS application environments.

  For more information, see Generating a Java Application with the eGen Application Generator.

- Oracle Tuxedo Mainframe Transaction Publisher
  Oracle Tuxedo Mainframe Transaction Publisher greatly simplifies access to mainframe transaction from Oracle Service Bus (OSB) by providing an Oracle JDeveloper based graphical user interface. This GUI tool, imports COBOL copybooks of mainframe transactions and generates all the artifacts required, including Java Beans and OSB configurations, to access mainframe transactions through Oracle Tuxedo Mainframe Adapters.

  For more information, see Tuxedo Mainframe Transaction Publisher.

- Developing New Applications Using Java
  Oracle Tuxedo enhances the way of developing new applications using Java.
  - Oracle Tuxedo supports the following new Java APIs in this release.
    - `tpacall`
    - `tpgetrply`
    - `tpcancel`
    - `tpsubscribe`
    - `tpunsubscribe`
    - `tppost`
    - `tpnotify`
    - `tpbroadcast`
    - `tpenqueue`
    - `tpdequeue`
    - `tpsblktime`
    - `tpgblktime`
    - `tpforward`
    - `tpsuspend`
    - `tpresume`
    - `tpscmt`
    - `tpsprio`
    - `tpgprio`
    - `tpappthrinit`
    - `tpappthrterm`

    * tpsetctxt
    * tpgetctxt
    * tpxmltofml
    * tpxmltofml32
    * tpfmltoxml
    * tpfml32toxml
    * Fvftos
    * Fvftos32
    * Fvstof
    * Fvstof32
  - Oracle Tuxedo introduces a new version of Java server configuration file. It is strongly recommended to use this new version.
  - Oracle Tuxedo expands standard Java server initialization and termination methods.
  - Oracle Tuxedo provides an additional running mode for more flexibility (Java server now can run in single-thread mode).

  For more information, see Oracle Tuxedo Java Server and Java Programming.

- Java Server Transaction Management Integrating with Spring Framework
  This release provides a JTA compliant transaction manager interface for Spring applications. Transaction manager interface can be specified in the Spring Framework application context configuration file and instantiated by Spring Framework. With the transaction manager, Spring applications can invoke Java server APIs to access existing Oracle Tuxedo application services, enqueue/dequeue message to/from Tuxedo persistent queue devices, execute database operations via the connection Java server creates, and perform event related operations in a distributed transaction environment.

  For more information, see Oracle Tuxedo Java Server.

- RECORD Features
  The RECORD facility is particularly useful when the data is transferred between COBOL language and C language. Under such condition, RECORD buffer type can be used in C language, and copybook is used in COBOL language. The RECORD facility has the following features:
  - The `cpy2record` tool is used to generate record descriptions (stored in binary format) that are interpreted by your application programs at run time.
  - At run time record descriptions are read into a record file cache on demand, and remain there until the cache is full. When the cache is full and a record description that is not in the cache is needed, the least recently accessed record description is removed from the cache to make room for the new one.
  - When transferring data between RECORD buffers and COBOL records, the source data is automatically converted to the type of the destination data. For instance, a string field may be converted between EBCDIC and ASCII formats.

  For more information, see `cpy2record(1)` in Oracle Tuxedo Command Reference, Programming An Oracle Tuxedo ATMI Application Using FML, Managing Typed Buffers in Programming An Oracle Tuxedo ATMI Application Using C, and Setting Up Data Translations in Oracle Tuxedo Mainframe Adapter for SNA User's Guide.

1.1.12 Release 12c Release 2 (12.2.2)

Release 12c Release 2 (12.2.2) introduced the following new major features and enhancements:

- Distributed Caching
  Distributed caching feature of Tuxedo provides access to a distributed cache to Tuxedo applications. This feature leverages Oracle Coherence as the distributed cache and provides new APIs for cache access. With this feature, one can take advantage of all the benefits that Oracle Coherence has to offer for distributed caching. This feature enabled the following use cases:
  - Data caching for Tuxedo applications
    When data caching is enabled, one can store Tuxedo typed buffer in the distributed cache. Tuxedo applications, running anywhere in Tuxedo domain, can now retrieve the data from the cache. This offers a new way of sharing data between clients and servers as well as a way to cache frequently accessed data, without having to go to the database every time. Most Tuxedo buffer types are supported.
  - Result caching for Tuxedo services
    When result caching is enabled, Tuxedo first checks the cache to see if response for the request buffer exists in the cache. This is done based on a criterion, which consists of configurable fields in the request buffer. If cache hit succeeds, response is returned from the cache and service is not invoked. Service is invoked if a current cache entry does not exist. Once service is invoked, results are placed in to the cache.

  For more information, see Using Oracle Tuxedo Distributed Caching (TDC).
- Faster and Flexible Startup of Applications
  With this feature, one can significantly improve the start-up time of large applications. Servers configured in an application can be started in parallel. The feature enables specifying startup dependencies of boot sequence at the group and server levels. For more information, see tmboot(1).
- Integrating Audit with Oracle Platform Security Services (OPSS)
  This feature enables auditing of Tuxedo services based on Oracle Platform Security Services (OPSS) audit component. Based on the configuration, specific events with relevant data are generated, which can eventually be stored on a file system or eventually in the Database to further analysis using BI tools. Straightforward XML based configuration makes it easy to add/change audit policies without any impact to the application. For more information, see Integrating Audit with Oracle Platform Security Services (OPSS) and Implementing Custom Auditing.
- Integration with Oracle Access Manager (OAM)
  This release allows Tuxedo applications to leverage rich security features of Oracle Access Manager. Tuxedo applications can authenticate and authorize using the credentials and resource authorization policies stored in OAM. This feature also allows single-sign-on of mobile and other applications when accessing Tuxedo services using SALT Web services. For more information, see, Setting up OAUTHSVR as the Authentication Server, and `OAUTHSVR (5)`
- Java Server Modules and Dynamic Configuration Reload
  Oracle Tuxedo Java server introduces a module entity to provide better isolation among different applications running within a Tuxedo Java server. Tuxedo Java

server also provides dynamic configuration reload capability to allow users to add, remove and update applications with no need to restart Tuxedo Java server. For more information, see Oracle Tuxedo Java Server.

- XA Transaction Enhancements
  Following XA related enhancements are included in this release:
  - Logging Last Resource: With Logging Last Resource feature, Tuxedo allows one non-XA resource to participate in an XA global transaction. For more information, see Logging Last Resource Transaction Optimization.
  - In the event of a Resource Manager failure, Tuxedo servers can be configured to suspend the impacted services, and keep try reconnecting to the RM, and resume the related services when the reconnection to RM succeeds. For more information, see `RM_ERR_THRESHOLD` `UBBCONFIG(5)` `SERVERS` section, and `TA_RM_ERR_INTERVAL` `TM_MIB(5)` `T_SERVER` Class Definition.
  - In the event a TMS process goes down, the XA transaction aborts immediately, rather than waiting for transaction time out. Reduces response time.
- SNMPv3 Support
  In this release, SNMP agent has been upgraded to support SNMPv3. As a result of this upgrade, two important security features are introduced:
  - USM (User-based Security Model)
    Provides authentication and privacy (encryption) functions and operates at the message level.
  - VACM (View-based Access Control Model)
    Determines whether a given principal is allowed access to a particular MIB object to perform specific functions and operates at the PDU level. For more information, see Using SNMPv3.
- New Service Management Console
  This release of Tuxedo introduces a new console for managing Tuxedo services. The easy-to-use and cool looking UI provides following functionality in this release:
  - Tuxedo metadata repository editor: allows to add/edit/delete Tuxedo service definitions to be used by Jolt, SALT or other Tuxedo components.
  - Web services (SOAP and REST) configuration: enables Tuxedo services to be accessed as SOAP or REST services and enables Tuxedo applications to access external SOAP/REST Web services.
  - Mainframe transaction integrator: enables mainframe transactions to be accessed as Web services (SOAP or REST) or enables mainframe transactions to access external Web services.

  A new Tuxedo system server, `TMADMSVR`, is configured in `UBBCONFIG` in order to use this console. For more information, see TMADMSVR and `MTP`.
- Enhancements to RECORD Buffer Type
  This release includes the following enhancements to the RECORD buffer type:
  - Support for conversion from RECORD to VIEW/32 and FML/32 and vice-versa
  - Support for `REDEFINES` in the RECORD buffer
  - Support for RECORD buffer type in Jolt and SALT

  For more information, see RECORD Features, RECORD Functions, File Formats, Data Descriptions, MIBs, And System Processes Reference, and Command Reference.
- MSSQ Notification

With this feature enabled, a Tuxedo request can be sent to a specific MSSQ server instance. This is made possible via a new API `tpadvertisex(3c)`, which allows to advertise a singleton service, a service which can be advertised by one and only one server instance. With this feature enabled, XA affinity feature can work in MSSQ configurations. For more information, see Advertising and Unadvertising Services and `tpadvertisex(3c)`.

- Installer Enhancements
  - Console mode installation: Tuxedo installer now supports console install/deinstall mode.
  - Clone mode installation: Tuxedo installer also supports clone mode to copy an existing Oracle Tuxedo installation to a different location and update the copied bits to work in the new environment.

  For more information, see Preparing to Install the Oracle Tuxedo System.
- Statistical Trace in GWTDOMAIN
  This feature gathers statistics for the remote Oracle Tuxedo domain service calls, and flushes the statistics trace to the audit log file in a specific interval time. This feature makes it easier for you to track cross domain activities. For more information, see `tmtrace(5)` and `tmadmin(1)`.
- TMTRACE Enhancement
  This release of Tuxedo allows `tmtrace` to be enabled at the service level. You can now trace one particular service in a server, which may have many services. For more information, see `tmtrace(5)` and tmadmin(1).
- Jolt Enhancements
  This release of Tuxedo Jolt includes tmtrace for Jolt clients, providing workstation client equivalent tracing. This release also includes ECID propagation via Jolt clients, if running in WebLogic Server. For more information, see `tmtrace(5)` and `tmadmin(1)`.

Release 12c Release 2 (12.2.2) deprecated the following functionality:

- Jolt Repository
  Jolt Repository is deprecated and removed in this release. All service definitions stored in Jolt repository can be loaded in the Tuxedo metadata repository using the bulk loader tool. The original Jolt repository server JREPSVR is also deprecated, and all services that `JREPSVR` provided are now provided by `TMMETADATA`. If `tmloadcf` detects presence of `JREPSR` in `UBBCONFIG`, it automatically removes `JREPSVR` and adds `TMETADATA` if not already configured.

  Using one repository (Tuxedo metadata repository), and one server (`TMMETADATA`), improve operational effectiveness and reduces the risk of service definitions getting out of sync.

  For more information, see Using Oracle Jolt and Managing The Oracle Tuxedo Service Metadata Repository.
- Oracle Tuxedo Administration Console
  Oracle Tuxedo Administration Console is deprecated and removed in this release.

1.1.13 Release 22c Release 1 (22.1.0.0.0)

Oracle Tuxedo 22c Release 1 (22.1.0.0.0) introduces the following new major features and enhancements:

- Easy integration with Oracle Database Application Continuity
  Using Oracle Database Application Continuity with Tuxedo is easier than previous versions.

  > **See Also:**
  >
  > Tuxedo Application Leverage Oracle Database Application Continuity.

- Security Enhancements
  In Oracle Tuxedo 22c Release 1 (22.1.0.0.0) following are the updates to ensure that the default configuration for Tuxedo and its application is secure:

  1. TLS 1.2 is the default protocol for connection between Workstation client and Workstation handler/listener, Jolt client and Jolt handler/listener, CORBA client and CORBA handler/listener, between bridges,and between Domain Gateways.
  2. Default SSL cipher suite and key length is updated.
  3. We disable SNMP v1 and v2 protocols for the SNMP agent.
  4. XAUTHSVR connects to LDAP servers using TLS 1.2 by default.

  > **See Also:**
  >
  > Security Enforcement.

- Open JDK certification for Tuxedo Java Server
- Availability of Tuxedo 22c docker images in the Oracle Container Registry
- Availability of a sample Helm chart for Tuxedo deployment in Oracle Cloud Infrastructure (OCI) Container Engine for Kubernetes (OKE).

Release 22c Release 1 (22.1.0.0.0) deprecated the following functionality:

- Oracle SNMP Agent

  This release deprecates the Oracle SNMP agent, and its use is not recommended. For more information, see Oracle SNMP Agent.

Release 22c Release 1 (22.1.0.0.0) removed the following functionality:

- `GAUTHSVR`

  Use of `XAUTHSVR` is recommended in place of `GAUTHSVR`. For more information, see XAUTHSVR(5) and GAUTHSVR(5).

- Service Component Architecture (SCA)

  Starting from this release, Oracle desupports SCA. Oracle Tuxedo SCA Components. For more information, see Oracle Tuxedo SCA Components.

1.2 Support for Industry Standards

The Oracle Tuxedo system complies with the Open Group's X/Open standards, including support of the XA standard for two-phase commit processing, the X/Open ATMI API, and the X/Open Portability Guide (XPG) standards for language internationalization. Oracle Tuxedo also supports the CORBA specification for distributed application development, as well as

any relational database management system, object-oriented database management system, file manager, or queue manager.

The Oracle Tuxedo system and ATMI together implement the X/Open distributed transaction processing (DTP) model of online transaction processing (OLTP). The DTP model ensures that work being done throughout a client/server application is *atomically* completed, meaning that all involved databases are updated properly if the work is successful, or all involved databases are “rolled-back” to their original state if the work fails.

Other standards supported by the Oracle Tuxedo system include:

- Lightweight Directory Access Protocol (LDAP)—A set of protocols for accessing information directories. These directories can be physically distributed across multiple systems for access by many applications within an enterprise. LDAP is based on the standards contained within the X.500 standard, but is significantly simpler. And unlike X.500, LDAP supports TCP/IP, which is necessary for any type of Internet access. LDAP is an ideal way to publish certificates because it is closely coupled with the X.509 standard for certificates.
- X.509 Digital Certificates—A digital statement that associates a particular public key with a name or other attributes. The statement is digitally signed by a certificate authority. By trusting that authority to sign only true statements, you can trust that the public key belongs to the person named in the certificate. Oracle Tuxedo public key security recognizes certificates that comply with X.509 version 3.0.
- Public-Key Cryptography Standard 7 (PKCS-7)—One of a set of Public-Key Cryptography Standards developed by RSA Laboratories in cooperation with an informal consortium, originally including Apple, Microsoft, DEC, Lotus, Sun and MIT. PKCS-7 defines a general syntax for messages that include cryptographic enhancements such as digital signatures and encryption. Oracle Tuxedo public key security complies with the PKCS-7 standard.
- Secure Sockets Layer (SSL)—The standard protocol for establishing secure communications over the Internet (TCP/IP).
- Internet Protocol Version 6 (IPv6)—The next generation protocol designed by the IETF to replace the current version Internet Protocol, IP Version 4 (IPv4)

1.3 Support for Popular Platforms

A client/server application separates the calling (client) software and the called (server) software into separate programs. The advantage of a client/server application is that multiple client processes can interface with a single server process, where the processes do *not* need to run on the same host machine. Thus, clients and servers can run on hardware and software platforms suited to their particular functions. For example, clients can run on inexpensive platforms such as workstations or personal computers, and database management servers can run on platforms specially designed and configured to perform queries.

The Oracle Tuxedo system has been ported to most popular *client* platforms, including Microsoft Windows Server and XP, and a variety of UNIX workstations. The Oracle Tuxedo system has been ported to most popular server platforms, including Microsoft Windows Server, HP-UX, IBM AIX, and Sun Solaris.

For more information about the platform support for multiple Tuxedo releases, see Oracle® Tuxedo Family Platforms.

1.4 Support for Multiple Programming Models and Languages

Oracle Tuxedo supports two programming models and five languages. The supported programming models are ATMI and CORBA. The supported programming languages are:

- C and COBOL—supported for ATMI application clients and servers
- C++—supported for ATMI application clients and CORBA C++ application clients and servers
- Java—supported for CORBA Java application clients and Jolt application clients

> **Note:**
>
> The Oracle Tuxedo CORBA Java client and Oracle Tuxedo CORBA Java client ORB were deprecated in Tuxedo 8.1 and are no longer supported. All Oracle Tuxedo CORBA Java client and Oracle Tuxedo CORBA Java client ORB text references, associated code samples, should only be used to help implement/run third party Java ORB libraries, and for programmer reference only.
>
> Technical support for third party CORBA Java ORBs should be provided by their respective vendors. Oracle Tuxedo does not provide any technical support or documentation for third party CORBA Java ORBs.

1.5 Mission-Critical Software

ATMI and CORBA applications developed with Oracle Tuxedo are *mission-critical*, meaning that they are reliable, scalable, secure, and manageable. Applications can grow as the company grows, and they continue running when various parts of the network fail. Applications can expand and contract as the demand requires.

1.6 Distributed Transaction Management

Oracle Tuxedo specializes in managing transactions, on behalf of ATMI and CORBA applications, from their point of origin—typically on the client—across one or more server machines, and then back to the originating client. When a transaction ends, Tuxedo ensures that all the systems involved in the transaction are left in a consistent state. Tuxedo knows how to run transactions, route them across systems, load-balance their execution, and restart them after failures.

Oracle Tuxedo ensures the integrity of data accessed across several sites or managed by different database products. It tracks transaction participants and supervises a two-phase commit protocol, ensuring that transaction commit and rollback are properly handled at each site.

1.6.1 X/Open XA and TX Compliance

The Oracle Tuxedo system also coordinates the recovery of transactions in the event of site failure, network failure, or global resource deadlocks. The Oracle Tuxedo system uses the X/Open XA interface for communicating with the various resource managers. This interface,

proposed by Tuxedo developers and accepted by X/Open, is the standard interface for distributed transaction control between the transaction manager and resource managers.

Oracle Tuxedo uses the X/Open TX interface for transaction demarcation and its own ATMI transaction management functions (routines, verbs). The interface allows an application writer to bracket group operations within an application so that, either all of them are completed or none are completed. This is the process of committing or rolling back a transaction in a single atomic unit of work. This keeps all databases involved synchronized, even if one of the machines fails.

1.6.2 Transactions Documentation

For more information about transactions, see Introducing Oracle Tuxedo ATMI and Using CORBA Transactions.

1.7 Scalability and Performance

In an enterprise environment, applications supports hundreds of execution contexts (where the context can be a thread or a process), tens of thousands of client applications, and millions of objects at satisfactory performance levels. Subjecting an application to exponentially increasing demands quickly reveals any resource shortcomings and performance bottlenecks in the application. Scalability is therefore an essential characteristic of Oracle Tuxedo applications.

Oracle Tuxedo enables distributed applications to scale is response to changing transaction loads by dynamically spawning and terminating servers (ATMI) or by dynamically activating and deactivating objects (CORBA) to meet the workload demands. Oracle Tuxedo balances the workload among all the available services or objects to ensure that they are all evenly used.

Applications built on Oracle Tuxedo can support a single client on a single server, or they can support tens of thousands of clients and thousands of servers without changing application code. As an application scales, the Oracle Tuxedo system continues to provide end users with consistently high performance and good responsiveness.

For more information about scaling, see “Tuning an Oracle Tuxedo ATMI Application” in Administering an Oracle Tuxedo Application at Run Time and Scaling, Distributing, and Tuning CORBA Applications.

1.8 High Availability and Fault Management

In a distributed client/server environment, thousands of independent processors and processes must cooperate to run the application. Many malfunctions can happen. In spite of failures, Oracle Tuxedo keeps the application running in the following ways:

- Ensures no single point of failure by providing replicated server groups that can continue when something breaks.
- Restores the running application to good condition after failures occur.

Ensuring constant access to e-business applications is a key feature of Oracle Tuxedo. System components are constantly monitored for application, transaction, network, and hardware failures. When a failure occurs, Oracle Tuxedo logically removes that component from the system, manages any necessary recovery

procedures, and re-routes messages and transactions to surviving systems—all transparently to the end user and without disruption in service.

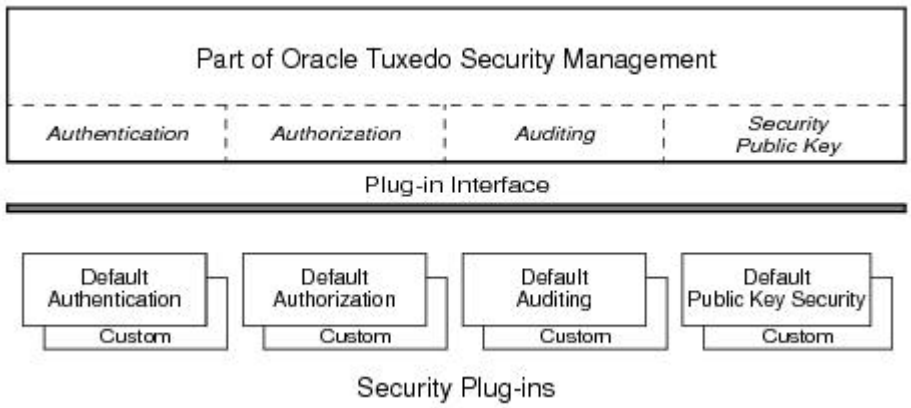
1.9 Security

Oracle Tuxedo security includes authentication, authorization, and encryption to ensure data privacy when deploying Oracle Tuxedo applications across networks. The following two levels of encryption are supported:

1. Encryption at the network level using SSL (Secure Sockets Layer)/TLS (Transport Layer Security) or LLE (Link-Level Encryption).
2. Encryption at the application level using the SSL protocol and public key encryption.

In order to integrate Oracle Tuxedo security with other security systems, Oracle Tuxedo provides the following security plug-in interface. The plug-in interface allows Tuxedo customers to independently define and dynamically add their own security plug-ins.

Figure 1-1 Oracle Tuxedo Plug-in Security Architecture



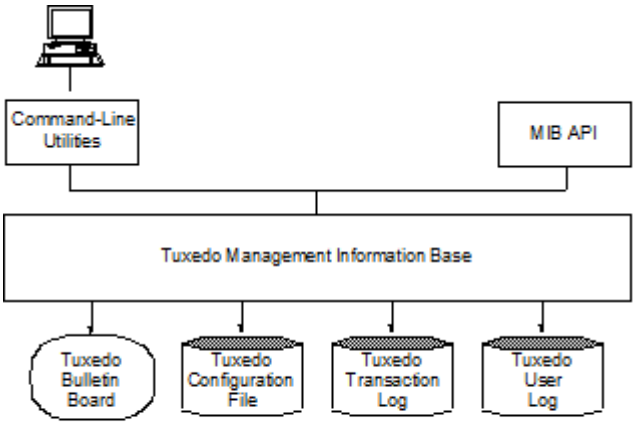
For more information on security in ATMI and CORBA applications, see Using Security in ATMI Applications and Using Security in CORBA Applications.

For information about security when interoperating with earlier releases of Oracle Tuxedo software or when interoperating with Oracle WebLogic Server, see Oracle Tuxedo Interoperability.

1.10 Management Tools

The Oracle Tuxedo system gives administrators a choice of several methods for performing the same set of administrative tasks for either Oracle Tuxedo ATMI or CORBA environments. The following figure illustrates the Oracle Tuxedo tools available to write an application's configuration file and dynamically administer an Oracle Tuxedo application during run time.

Figure 1-2 Simplified View of Administration Tools



In addition to using these tools to administrator Oracle Tuxedo applications, administrators use these tools to perform fault-isolation and recovery tasks when application failures occur. Oracle Tuxedo automatically recovers from many types of failures. However, some failures—often the most serious ones—require operator intervention to determine what has actually failed.

1.10.1 Command-Line Interface

Most of the functionality needed for dynamic modification of an Oracle Tuxedo application is provided by the `tmadmin` and `tmconfig` commands. Most of the functionality needed for dynamic modification of an Oracle Tuxedo Domains configuration is provided by the `dmadmin` command. Each of these commands is an interactive meta-command having many sub-commands for performing various administrative tasks, including the modification of configuration entries while the system is running. For details about these commands, see reference pages `tmadmin(1)`, `tmconfig`, `wtmconfig(1)`, and `dmadmin(1)` in Oracle Tuxedo Command Reference. Also, see "Oracle Tuxedo Management Tools in Introducing Oracle Tuxedo ATMI.

1.10.2 MIB Interface

The MIB interface is an application programming interface for directly accessing and manipulating system settings in the Oracle Tuxedo management information bases. The interface allows administrators to have total control over Tuxedo applications. The MIB interface is powerful because it is implemented with the same APIs that Tuxedo developers use to write business-critical client/server applications.

There are MIB interfaces to administer the access control list, disk-based queues, Domains, events, core Tuxedo, and the workstation extension. The following are the corresponding MIB component names: ACL_MIB, APPQ_MIB, DM_MIB, EVENT_MIB, TM_MIB, and WS_MIB. Through the MIB interface, administrators control the application by programmatically querying the Tuxedo bulletin board for the current

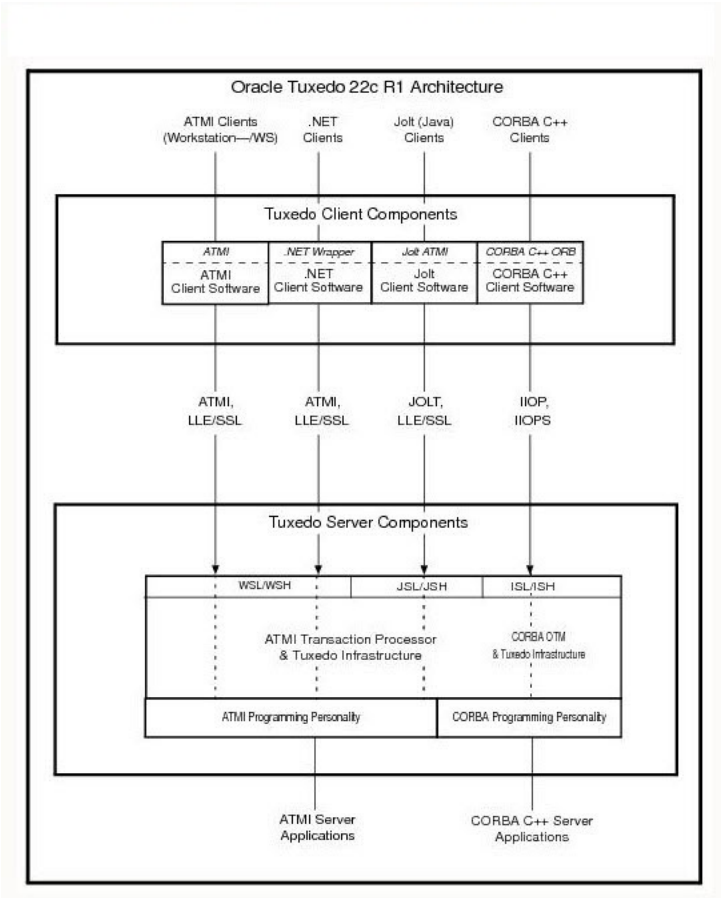
state of MIB objects, and then effecting administrative changes by either setting and resetting specific MIB values or creating new MIB objects.

The level of control available through the MIB interface really comes in handy in failover and fallback situations. The MIB programming interface is the only way to handle all the possible complications that can occur in a failover situation. During a failover, scripts can be used to execute client MIB programs that perform specifics task such as shutting down and migrating server groups, and verifying the state of the application. For details about the Oracle Tuxedo MIBs, see reference pages `ACL_MIB(5)`, `APPQ_MIB(5)`, `DM_MIB(5)`, `EVENT_MIB(5)`, `MIB(5)`, `TM_MIB(5)`, and `WS_MIB(5)` in Oracle Tuxedo Command Reference. Also, see Oracle Tuxedo Management Tools in Introducing Oracle Tuxedo ATMI.

1.11 Client and Server Components

The following figure identifies the Oracle Tuxedo client and server components and shows the connectivity between the clients and servers. The following figure illustrates the only remote Tuxedo clients:

Figure 1-3 Oracle Tuxedo Client and Server Components



A *remote* Tuxedo client—ATMI (/WS), Jolt, or CORBA C++—interfaces with a Tuxedo server via a network connection and a pair of Tuxedo gateway processes: Workstation Listener/Handler (WSL/WSH), Jolt Server Listener/Handler (JSL/JSH), or IIOP Listener/Handler (ISL/ISH). A remote Tuxedo client may run on a machine that is not part of the Tuxedo server

application (typically a workstation or personal computer), or the remote client may run on a machine that is part of the Tuxedo server application. For the latter case, the local operating system intercepts the messages destined for the network and redirects them to the destination process—the Tuxedo remote client or handler process—running locally.

A *native* Tuxedo client—a native ATMI client or a native CORBA C++ client—is co-located on a machine that is part of the Tuxedo server application and interfaces with a Tuxedo server via the Tuxedo infrastructure using interprocess communication. Native Jolt clients are not supported. These clients can only access a Tuxedo server via a pair of JSL/JSH gateway processes.

The following brief descriptions of other terms shown in the previous figure should prove helpful in understanding the connectivity between Oracle Tuxedo clients and servers:

IIOP
Internet Inter-ORB Protocol. A protocol used for communication between CORBA ORBs over the Internet (TCP/IP).

IIOPS
IIOP layered over the SSL protocol.

LLE
Link-Level Encryption. An Oracle Tuxedo protocol for establishing data privacy over network links between Oracle Tuxedo server machines.

SSL
Secure Sockets Layer. The standard protocol for establishing secure communications over the Internet (TCP/IP).

1.12 Invocation Capabilities

The following table lists the invocation capabilities for an application built on the Oracle system. An Oracle Tuxedo application may span multiple Oracle Tuxedo server machines and may provide ATMI services, CORBA objects, or both.

This component...	Can call a....	Through...
ATMI client *	ATMI service	WSL/WSH
Jolt client	ATMI service	JSL/JSH
CORBA C++ client *	CORBA C++ object	ISL/ISH
ATMI server	ATMI service	Tuxedo infrastructure
CORBA C++ object	CORBA C++ object	Tuxedo infrastructure
CORBA C++ object	ATMI service	Tuxedo infrastructure
* A native Tuxedo ATMI or CORBA C++ client does not use listener or handler gateway processes.		

Interoperability and coexistence between Tuxedo 22c and previous releases are as explained in the following list:

- Oracle Tuxedo 22c can coexist in the same domain with Oracle Tuxedo 22c,12.x, 11.x, 10.x, and 9.x.
- Oracle Tuxedo 22c supports interdomain interoperability with Oracle Tuxedo 22c,12.x, 11.x, 10.x, and 9.x.

- Oracle Tuxedo 22c ATMI server connects with Workstation client from 22c, 12.x, 11.x, 10.x, 9.x,6.5
- Oracle Tuxedo 22c Jolt server connects with Jolt client from 22c, 12.x, 11.x, 10.x, 9.x,6.5
- Oracle Tuxedo 22c workstation client connects with Oracle Tuxedo ATMI servers running on Tuxedo 22c, 12.x, 11.x, 10.x, and 9.x
- Oracle Tuxedo 22c Jolt client connects with Oracle Tuxedo Jolt server running on Oracle Tuxedo 22c, 12.x, 11.x, 10.x, and 9.x
- Oracle Tuxedo 22c CORBA client connects with Oracle Tuxedo 22c, 12.x, 11.x, 10.x, and 9.x
- Oracle Tuxedo 22c CORBA server connects with Oracle Tuxedo CORBA client running on Tuxedo 22c, 12.x, 11.x, 10.x, and 9.x

This is similar to Tuxedo 12.2.2, for information about the Oracle Tuxedo Interoperability, see Interoperability and Coexistence.

> **Note:**
>
> An Oracle Tuxedo clients cannot invoke each other.

1.12.1 Client-to-Server Invocation Capabilities

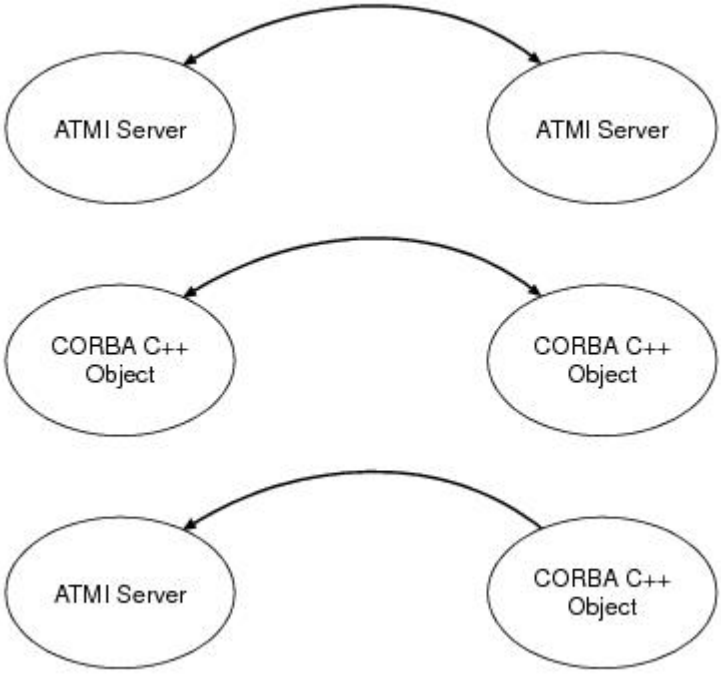
The following client-to-server invocations are supported by an Oracle Tuxedo application:

- An Oracle ATMI client invoking an Oracle Tuxedo service
  For more information about remote ATMI clients, see Using the Oracle Tuxedo ATMI Workstation Component.
- An Oracle Jolt client invoking an Oracle Tuxedo service
  For more information about Jolt, see Using Oracle Jolt and the Oracle Jolt API Javadoc reference information.
- An Oracle CORBA C++ client invoking an Oracle Tuxedo CORBA C++ object
  For details, see Creating CORBA Client Applications.

1.12.2 Server-to-Server Invocation Capabilities

The following figure shows the invocation capabilities between Oracle Tuxedo ATMI and CORBA C++ application servers.

Figure 1-4 Oracle Tuxedo ATMI and CORBA C++ Server Invocations



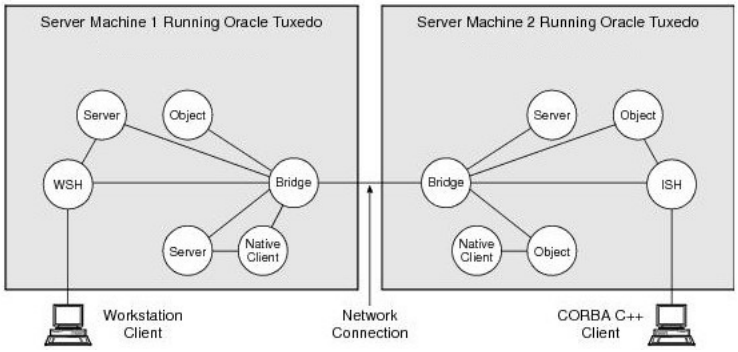
As shown in the figure, a CORBA C++ object can include ATMI calls to Oracle Tuxedo services. For an example, see the Wrapper University sample application, available in the Guide to the CORBA University Sample Applications.

1.13 Domains

An Oracle Tuxedo domain, or application, is defined and controlled by a single configuration file. A Tuxedo domain consists of many Tuxedo system processes, one or more application client processes, one or more application server processes, and one or more computer machines connected over a network. It is administered as a single unit.

An Oracle Tuxedo domain may provide ATMI services, CORBA objects, or both. The Tuxedo domain in the following example contains a mixture of ATMI services and CORBA objects.

Figure 1-5 Simplified View of an Oracle Tuxedo Domain



In Oracle Tuxedo terminology, a domain is the same as an application—a business application; both terms are used as synonyms throughout the Oracle Tuxedo user documentation. Examples of business applications currently running on Tuxedo are airline and hotel reservation systems, credit authorization systems, stock-brokerage systems, banking systems, and automatic teller machines.

For more information about Tuxedo domains, see Important Oracle Tuxedo Terms and Concepts. For information about interconnecting Tuxedo domains, see Tuxedo domains.

1.14 Oracle Tuxedo Product Family

The Oracle Tuxedo Family is comprised of the following:

- Oracle Tuxedo
- Oracle ART
- Oracle JCA
- Oracle Jolt
- Oracle SALT
- Oracle SNMP Agent
- Oracle TSAM
- Oracle Mainframe Adapter For TCP
- Oracle Mainframe Adapter For SNA

Excluding Oracle Tuxedo, this section provides a general introduction to the other Oracle Tuxedo family products.

- About Oracle ART
- About Oracle JCA
- About Oracle Jolt
- About Oracle SALT
- About Oracle SNMP Agent
- About Oracle TSAM
- About Oracle Mainframe Adapter For TCP

- About Oracle Mainframe Adapter For SNA

1.14.1 About Oracle ART

Oracle ART Runtime
In a z/OS environment, CICS is used to establish transactional communications between end-users and compiled programs via screens.

CICS is middleware that implements the control and integrity of shared resources, providing developers with APIs (EXEC CICS … END-EXEC statements) to dialog with CICS inside programs mainly developed on z/OS in COBOL, PL1, and Assembler languages.

Once all the components of z/OS CICS applications (COBOL programs and data) are migrated to a UNIX/Linux platform using Oracle Tuxedo Application Runtime Workbench, CICS Runtime enables them to be run unchanged using an API emulation on top of the native Oracle Tuxedo features.

Oracle ART Workbench
Oracle Tuxedo Application Rehosting Workbench is part of a packaged and comprehensive solution that enables its users:

- To perform a replatforming project with minimum risk and cost
- To run the replatformed applications in a standardized UNIX/Linux, Tuxedo, and Oracle environment

1.14.1.1 Oracle Art Runtime

In a z/OS environment, CICS is used to establish transactional communications between end-users and compiled programs via screens.CICS is middleware that implements the control and integrity of shared resources, providing developers with APIs (EXEC CICS … END-EXEC statements) to dialog with CICS inside programs mainly developed on z/OS in COBOL, PL1 and Assembler languages.Once all the components of z/OS CICS applications (COBOL programs and data) are migrated to a UNIX/linux platform using Oracle Tuxedo Application Runtime Workbench, CICS Runtime enables them to be run unchanged using an API emulation on top of the native Oracle Tuxedo features.

1.14.1.2 Oracle ART Workbench

Oracle Tuxedo Application Rehosting Workbench is part of a packaged and comprehensive solution that enables its users:To perform a replatforming project with minimum risk and cost;To run the replatformed applications in a standardized UNIX/Linux, Tuxedo, Oracle environment.

For more Oracle Application Workbench information, see the following documentation: Oracle Tuxedo Application Rehosting Workbench.

1.14.1.3 Oracle ART Documentation

For more Oracle runtime information, see the following documentation:Oracle Tuxedo Application Runtimes .

1.14.2 About Oracle Tuxedo JCA Adapter

The Oracle Tuxedo JCA Adapter is a JCA-based resource adapter that provides bi-directional service invocation between a JCA 1.5 compliant application servers and the Oracle Tuxedo system. Oracle Tuxedo JCA Adapter supports global and local transactions conforming to JCA transaction standards. It supports connection management, transaction infestation, identity propagation, and link-level security. Link-level Security uses industry standard SSL/TLS or a proprietary high performance algorithm.

1.14.2.1 Oracle Tuxedo JCA Adapter Documentation

For more Oracle Tuxedo JCA Adapter information, see the following documentation:Oracle Tuxedo JCA Adapter.

1.14.3 About Oracle Jolt

Oracle Jolt is a Java class library and API that enables remote Java clients to access existing Oracle Tuxedo ATMI services. It enables users to build client applets and applications that can remotely invoke Tuxedo ATMI services—such as application messaging, component management, and distributed transaction processing—using an ordinary Web browser.

Oracle Jolt extends the functionality of existing Tuxedo ATMI applications to include intranet- and Internet-wide availability. Oracle Jolt also enables Oracle WebLogic Server to invoke Tuxedo ATMI services. For clarification, see Making Tuxedo Services Web Accessible Through Oracle Jolt.

1.14.3.1 Oracle Jolt Documentation

For more information about Oracle Jolt, see the following documentation:

- Using Oracle Jolt
- Using Oracle Jolt with Oracle WebLogic Server

1.14.4 About Oracle SALT

Oracle Service Architecture Leveraging Tuxedo (SALT) is an add-on product option for Tuxedo, enabling Tuxedo applications to participate in SOA environments. Oracle SALT has two major components: native Web services stack and SCA container.

Oracle SALT allows external Web services applications to invoke Tuxedo services as Web services, and Tuxedo applications to invoke external Web services. Oracle SALT does not require any coding to achieve this. In addition, Oracle SALT includes SCA container, which allows you to develop new SOA applications focusing on business logic, while still taking advantage of Tuxedo infrastructure. SCA container also helps with effective reuse of existing application assets.

1.14.4.1 Oracle SALT Documentation

For more Oracle SALT information, see the following documentation: Oracle® Service Architecture Leveraging Tuxedo (SALT).

1.14.5 About Oracle SNMP Agent

Oracle SNMP Agent for Oracle Tuxedo enables SNMP-compliant network management frameworks to manage Oracle Tuxedo systems and Oracle Tuxedo applications. Oracle SNMP Agent complies with the Simple Network Management Protocol version 1 (SNMPv1) specification. Oracle SNMP Agent provides the SNMP links from Tuxedo applications to SNMP-based system-management consoles. It also allows multiple SNMP agents and subagents—from any vendor—to operate on the same machine.

1.14.5.1 Oracle SNMP Agent Documentation

For more information about Oracle SNMP Agent, see the following documentation:

- Oracle Tuxedo SNMP Agent Administration Guide
- Oracle Tuxedo SNMP Agent MIB Reference

1.14.6 About Oracle TSAM

Oracle Tuxedo System and Application Monitor (TSAM) provides comprehensive monitoring and reporting for Oracle Tuxedo system and applications. It includes two components: Oracle TSAM Agent and Oracle TSAM Manager. The Oracle TSAM Agent enables collection of various performance metrics for applications, including XA and non-XA transactions, services, system servers.

Oracle TSAM Manager provides graphical user interface to correlate and aggregate performance metrics collected from one or more Tuxedo domains and display it in real time.

The major features included in Oracle TSAM are:

- Call path monitoring and analysis
- Service monitoring and statistics
- System server monitoring and statistics
- Transaction monitoring
- SLA Event

1.14.6.1 Oracle TSAM Documentation

For more Oracle TSAM information, seeOracle Tuxedo System and Application Monitor (TSAM).

2
Oracle Tuxedo ATMI Core Components

The following sections describe the Oracle Tuxedo ATMI components and the Oracle Tuxedo infrastructure:

2.1 Important Oracle Tuxedo Terms and Concepts

The following terms and concepts are fundamental to understanding the Oracle Tuxedo system and applications built on the Oracle Tuxedo system:

Tuxedo domain
An Oracle Tuxedo domain (also known as an Oracle Tuxedo application), is a set of Tuxedo system, client, and server processes administered as a single unit from a single Tuxedo configuration file. A Tuxedo domain consists of many system processes, one or more application client processes, one or more application server processes, and one or more computer machines connected over a network. An Oracle Tuxedo domain may provide ATMI services, CORBA objects, or both.

> **Note:**
>
> A Tuxedo *domain* has the same meaning as a Tuxedo *application*.

Tuxedo configuration file
Each Oracle Tuxedo domain is controlled by a configuration file in which installation-dependent parameters are defined. The text version of the configuration file is referred to as `UBBCONFIG`, although the configuration file may have any name, as long as the content of the file conforms to the format described on reference page UBBCONFIG(5) in Oracle Tuxedo File Formats, Data Descriptions, MIBs, and System Processes Reference File Formats, Data Descriptions, MIBs, and System Processes Reference.

The binary version of the `UBBCONFIG` file is referred to as `TUXCONFIG`.As with `UBBCONFIG`, the `TUXCONFIG` file may be given any name; the actual name is the device or system filename specified in the `TUXCONFIG` environment variable.

Tuxedo master machine
The master machine, or master node, for an Oracle Tuxedo domain is a server machine containing the domain's `UBBCONFIG` file, and is designated as the master machine in the `RESOURCES` section of the `UBBCONFIG` file. Starting, stopping, and administering the one or more server machines in a Tuxedo domain is done through the master machine.

The master machine for a Tuxedo domain also contains the master copy of the `TUXCONFIG` file. Copies of the `TUXCONFIG` file are propagated to every other server machine—referred to as *non-master machines*—in a Tuxedo domain whenever the Tuxedo system is booted on the master machine.

Tuxedo bulletin board
The Oracle Tuxedo system uses the `TUXCONFIG` file to set up a *bulletin board* on each server machine in a Tuxedo domain. When a Tuxedo server process becomes active, it advertises the names of its services in the bulletin board. Some information in the bulletin board is global and is replicated on every server machine in the Tuxedo domain (for example, the names and locations of all servers offering a particular service). Other information is local and is visible only on the local bulletin board (for example, the actual number and type of client requests currently waiting on a local server request queue).

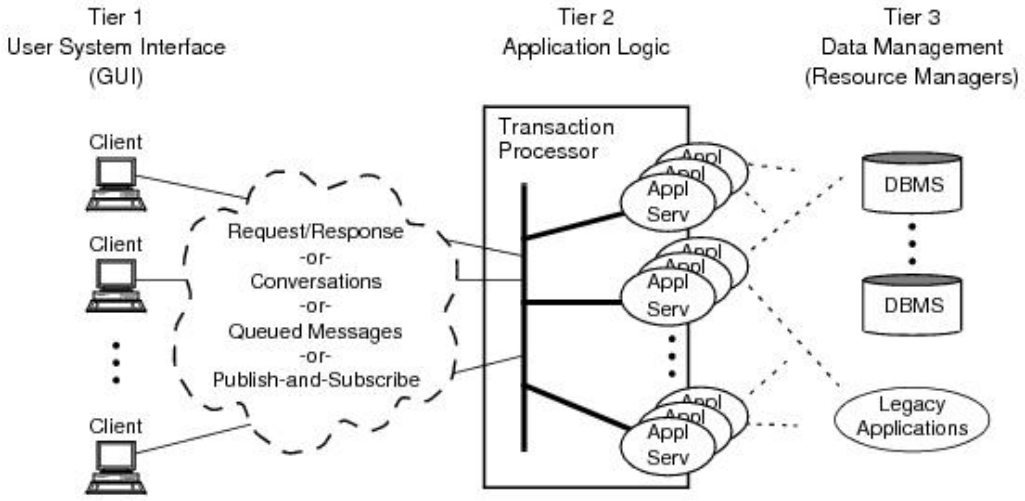
The bulletin board provides location and namespace transparency within a Tuxedo domain. Location transparency means that Tuxedo client and server processes do not have to be aware of the location of a resource (ATMI service, CORBA C++ object) within the Tuxedo domain. Namespace transparency means that Tuxedo client and server processes can use the same naming conventions (and namespace) to locate any resource in the Tuxedo domain.

2.2 Oracle Tuxedo ATMI Overview

Oracle Tuxedo ATMI is a set of core technologies that enables application designers to create ATMI applications that mix and match hardware platforms, databases, and operating systems. It provides all the features and benefits of a high-end online transaction processing (OLTP) system, including scalability, high-performance, mission-critical reliability, and open standards support.

At the foundation of Oracle Tuxedo ATMI is a proven, reliable transaction processor, also known as a transaction processing (TP) monitor. As shown in the following figure, a transaction processor is an example of a 3-tier client/server architecture, where the transaction processor supports the application logic between the GUI front-end and the back-end resource managers. Examples of resource managers are SQL databases, message queues, legacy applications, and other back-end services.

Figure 2-1 3-Tier Client/Server Architecture Using a Transaction Processor



By breaking the direct connection between the user interface front-end and the resource managers, a transaction processor controls all the traffic that links hundreds

or thousands or even tens of thousands of clients with application programs and the back-end resources. A transaction processor ensures that global (distributed) transactions are completed accurately, provides load balancing, and improves the overall system performance. More importantly, a transaction processor makes an application's server processes independent of the user interface front-end and any resource manager.

Oracle Tuxedo ATMI is a transaction application server that runs server-side applications and components. Besides managing an application's server processes and managing transactions, Oracle Tuxedo ATMI also manages client/server communications, that is, allows clients (and servers) to invoke an application service in a variety of ways, including:

Request/response
Request/response transactions usually involve people and thus require immediate attention; they run in high-priority mode. Oracle Tuxedo ATMI provides an ATMI request/response transactional communication interface.

Conversations
Conversational transactions also usually involve people and thus require immediate attention; they run in high-priority mode. Oracle Tuxedo ATMI provides an ATMI conversational transactional communication interface.

Queuing
Queued transactions can run as high-priority or low priority messages. Oracle Tuxedo ATMI includes its own bundled version of recoverable queues called */Q*.

Publish-and-subscribe
Publish-and-subscribe transactions usually run as high-priority messages. Oracle Tuxedo ATMI has a transactional publish-and-subscribe system called *EventBroker*.

Transactional communications use highly augmented versions of remote procedure calls, conversational peer-to-peer, queues, and publish-and-subscribe. However, most of the value-added elements are transparent to the programmer: The transactional client/server exchanges look like ordinary exchanges bracketed by start and end transaction calls. The distinguishing factor is that all resource managers and processes invoked through these calls become part of the transaction. A transaction processor, such as Oracle Tuxedo ATMI, orchestrates the actions of all the participants and makes them act as part of a transaction.

2.3 Oracle Tuxedo ATMI Architecture

Oracle Tuxedo ATMI consists of the following main components:

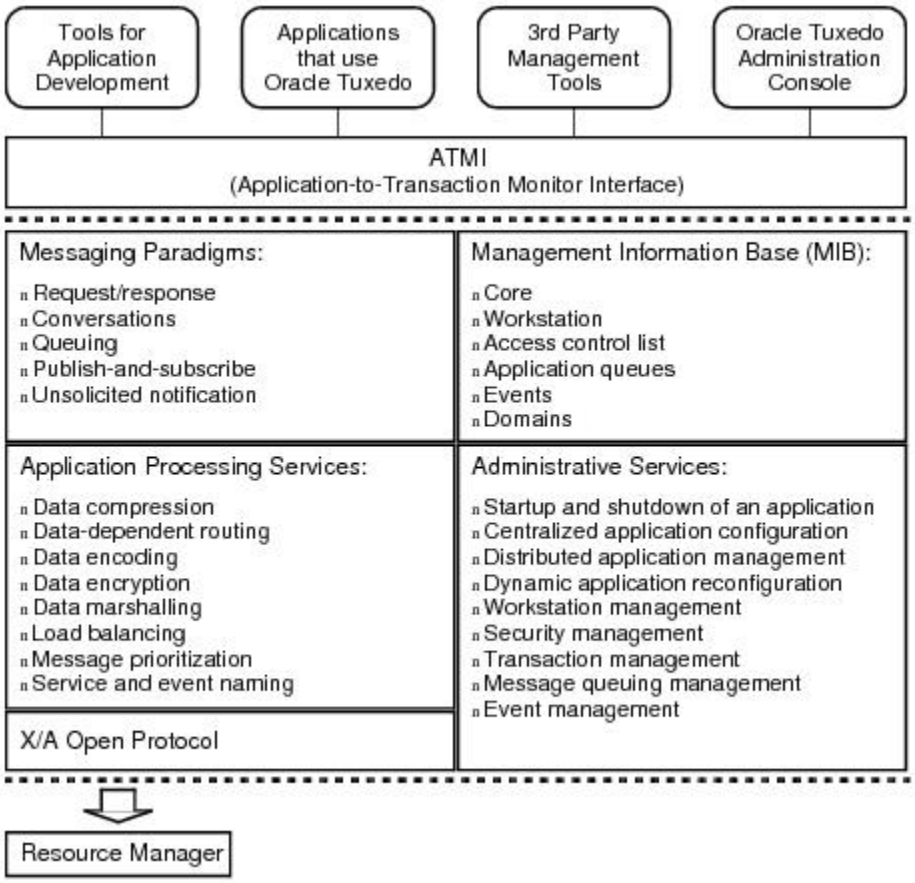
- Oracle Tuxedo transaction processor and infrastructure
  Provides the core services needed to run and administer a distributed ATMI application.
- Oracle Tuxedo Workstation
  Allows ATMI clients to reside on intelligent workstations and communicate over a network connection with an ATMI server application.
- Oracle Tuxedo /Q
  Provides a messaging and queuing capability to allow ATMI clients and servers to communicate across networks without being linked by a private, dedicated, logical connection.
- Oracle Tuxedo EventBroker
  Provides a publish-and-subscribe capability that brokers the distribution of application and system events between ATMI clients and servers.
- Oracle Tuxedo Domains

Offers the ability to connect ATMI applications that are logically and physically separate so that the combination appears to the user as a single application.

2.4 Oracle Tuxedo Transaction Processor and Infrastructure

The Oracle Tuxedo infrastructure provides the bedrock client/server architecture for both Oracle Tuxedo ATMI and Oracle Tuxedo CORBA. The transaction processor and infrastructure discussed here and illustrated in the following figure constitute the Oracle Tuxedo ATMI environment, which provides request/response and conversational communication interfaces, transaction support, and application-processing and administrative services for a distributed ATMI application.

Figure 2-2 Oracle Tuxedo ATMI Environment



2.4.1 System Management Interface

The Oracle Tuxedo system management interface, common to both Oracle Tuxedo ATMI and Oracle Tuxedo CORBA, can accommodate tools for administration, such as those described in Management Tools, and tools for application development, such as Simple Network Management Protocol (SNMP) agents. Oracle Tuxedo provides an open tool environment that is supported by many third-party tools.

2.4.2 ATMI Programming Interface

Oracle Tuxedo ATMI supports an ATMI programming interface that offers procedural library-based programming using a set of C or COBOL procedures. ATMI provides an interface for communications, transactions, and data-buffer management that works in all ATMI environments supported by the Oracle Tuxedo system. The ATMI interface and the Oracle Tuxedo system implement the X/Open distributed transaction processing (DTP) model for transaction processing.

The Oracle Tuxedo ATMI interface provides a foundation for request/response and conversational communications.

2.4.2.1 Request/Response Communications

Programmers use the ATMI request/response functions to send a single request from a requesting process, and to receive a single response from the called request/response server process. Request/response is a simple type of dialogue. The rules for communication during request/response are fixed: the client asks for a service and the server responds. The client never sends more than one message as part of its request, and the server never sends more than one response in its reply. For the requesting process, the execution of a request/response service can be synchronous or asynchronous.

2.4.2.2 Conversational Communications

Programmers use the ATMI conversational functions to establish and maintain state-preserving connections—context kept from message to message—between a requesting process and the called conversational server process. Specifically, programmers use the ATMI conversational functions to:

- Open a connection to a conversational server
- Begin and end a transaction during the conversation
- Have a conversation span multiple machines and resource managers
- Detect and provide notification of connection failures
- Terminate the connection when satisfied that the task has been completed

A conversational server is dedicated to the originating requester for the duration of the connection. The Oracle Tuxedo system automatically spawns a new copy of a server if one is not available when a conversational connection is requested. Thus, using the ATMI conversational programming interface, programmers can define transaction boundaries within their application so that the work performed can be treated as an atomic unit. What this statement means is that within a single Oracle Tuxedo transaction, the work performed is either committed or rolled back as a single unit of work, which keeps all the databases synchronized, even if there are machine failures.

2.4.2.3 ATMI Interface Documentation

For more information on the Oracle Tuxedo ATMI interface, see ATMI Introduction.

2.4.3 FML Programming Interface

In addition to the ATMI interface, Oracle Tuxedo ATMI supports a Field Manipulation Language (FML) programming interface, which is a set of C language functions for defining

and manipulating storage structures called fielded buffers. Fielded buffers contain attribute-value pairs in fields, where the attribute is the field's identifier, and the associated value represents the field's data content.

If the FML and its fielded buffer concept are specified by the application designers, application programmers have a rich array of functions for the definition and management of FML fields and buffers (See Managing Typed Buffers.) The selection includes functions to move data back and forth between a fielded buffer and a C structure or COBOL record (referred to as a VIEW), the members of which parallel the buffer's fields.

The FML function set has a companion function set, FML32, designed for use with larger records with more fields.

For more information on Oracle Tuxedo FML, see Programming an Oracle Tuxedo Application Using FML.

2.4.4 Typed Buffers

Oracle Tuxedo ATMI applications send and receive their data in typed buffers. Instead of allocating memory directly from the operating system, applications allocate typed buffers from the Oracle Tuxedo system in which to place their data.

Typed buffers are data structures defined by application programmers and made known to the Oracle Tuxedo system. Because the Oracle Tuxedo system knows about the application data buffers, it can optimally manipulate them during communication. Typed buffers contain information about themselves (metadata), which allows application programmers to transfer data without needing to know which data representation scheme is used by the machines on which the application's clients and servers are running. Typed buffers allow applications to maintain machine independence.

Each buffer type supported by an Oracle Tuxedo release has its own set of routines that can be called automatically to initialize; send and receive messages; and encode and decode data without programmer intervention. The set of routines is called a *typed buffer switch*.

Oracle Tuxedo provides different kinds of typed buffers, including FML and FML32, and allows application designers to define their own typed buffers. For more information about typed buffers, see Managing Typed Buffers in *Introducing Oracle Tuxedo ATMI*.

2.5 Oracle Tuxedo Workstation

The Oracle Tuxedo Workstation component allows ATMI clients to reside on a remote machine that does not have a full Oracle Tuxedo server-side installation, that is, a machine that does not support Oracle Tuxedo administration servers or a bulletin board. All communication between a remote ATMI client and the Oracle Tuxedo server application takes place over the network.

Advantages of the Oracle Tuxedo Workstation component include:

- Less administrative overhead
- Greater security—keeps clients off the Oracle Tuxedo server machines

- Off loads CPU cycles and decreases process context switches on Oracle Tuxedo server machines
- Smaller footprint

2.5.1 Workstation Communication

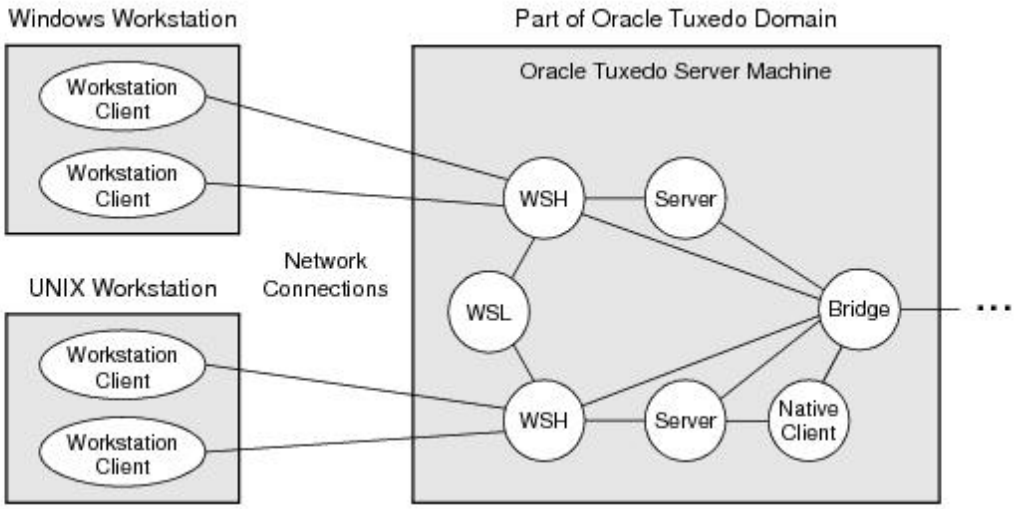
The Workstation component involves the following software processes:

Workstation Client
An ATMI client process that runs on a machine on which the Oracle Tuxedo Workstation client software is installed.

Workstation Listener (WSL)
An Oracle Tuxedo listening process, running on an Oracle Tuxedo server machine, that accepts connection requests from Workstation clients and assigns connections to a Workstation Handler also running on the server machine. It also manages the pool of Workstation Handler processes, starting them in response to load demands.

Workstation Handler (WSH)
An Oracle Tuxedo gateway process, running on the Oracle Tuxedo server machine, that handles communications between Workstation clients and the Oracle Tuxedo server application. A WSH process resides within the administrative domain of the application and is registered in the local Oracle Tuxedo bulletin board as a client. Each WSH process can manage multiple Workstation clients. A WSH multiplexes all requests and replies with a particular Workstation client over a single connection. The following figure shows how these processes are used to connect remote ATMI clients to the Oracle Tuxedo server application.

Figure 2-3 Connecting Remote ATMI Clients



2.5.2 Workstation Documentation

For more information about the Oracle Tuxedo Workstation component, see the following documents:

- "Oracle Tuxedo System Administration and Server Processes" in *Introducing Oracle Tuxedo ATMI*
- Using the Oracle Tuxedo ATMI Workstation Component
- "Administering Security" in *Using Security in ATMI Applications*
- UBBCONFIG(5), WS_MIB(5), and WSL(5) in Oracle Tuxedo File Formats, Data Descriptions, MIBs, and System Processes Reference

2.6 Oracle Tuxedo EventBroker

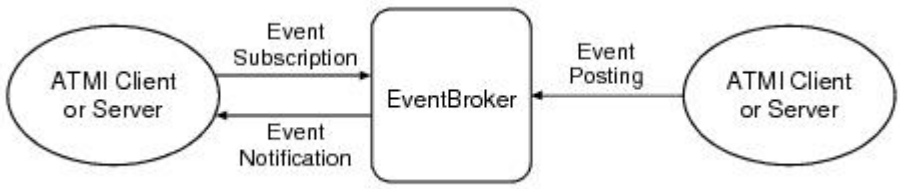
Oracle Tuxedo EventBroker is a transactionally enabled, XA compliant, application publish-and-subscribe system that provides asynchronous routing of application events among the processes running in an Oracle Tuxedo ATMI application. It also distributes system events to whichever application processes want to receive them.

An event is a state change or other occurrence in an application program or the Oracle Tuxedo system that may be of interest to an administrator, an operator, or the software. Examples of events are "a stock traded at or above a specified price" or "a network failure occurred."

2.6.1 Mediating Between Producers and Consumers of Events

There are producers of events, called *publishers* or *suppliers*, and consumers of events, called *subscribers*. EventBroker mediates between the producers and consumers about the distribution of events. The following figure shows how publish-and-subscribe communication works using EventBroker.

Figure 2-4 Event Subscription, Posting, and Notifications



When an event is posted in a global transaction, all work, including non-related work, is ensured to be complete if the transaction fails. In the event of any failures within the transaction, all work done within that transaction will be undone.

2.6.2 EventBroker Documentation

For more information about the Oracle Tuxedo EventBroker component, see the following documents:

- "Oracle Tuxedo System Administration and Server Processes" in *Introducing Oracle Tuxedo ATMI*
- "About the EventBroker" in *Administering an Oracle Tuxedo Application at Run Time*

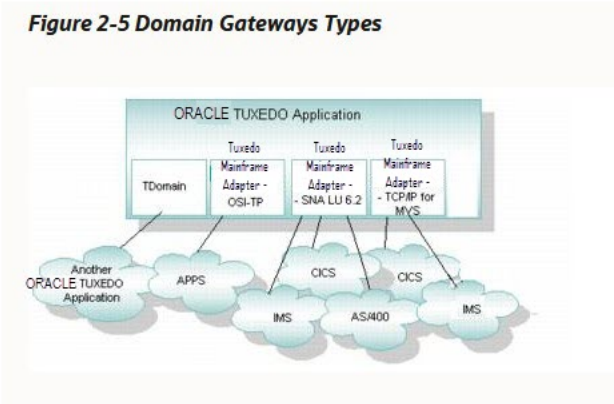
- tppost(3c), tpsubscribe(3c), and tpunsubscribe(3c) in *Oracle Tuxedo ATMI C Function Reference*.
- EVENTS(5), EVENT_MIB(5), TMSYSEVT(5), TMUSREVT(5), and UBBCONFIG(5) in *Oracle Tuxedo File Formats, Data Descriptions, MIBs, and System Processes Reference*.

2.7 Oracle Tuxedo Domains

The Oracle Tuxedo Domains component extends the Oracle Tuxedo system client/server model to provide transaction interoperability across TP domains—business applications. This extension preserves the model and the ATMI interface by making access to services on the remote domain (or accepting service requests from a remote domain) transparent to both the application programmer and the end-user. The Domains component makes this possible via a highly asynchronous multitasking *domain gateway* that handles outgoing and incoming service requests to or from remote domains.

The Oracle Tuxedo system offers the following types of domain gateways to allow an Oracle Tuxedo application to communicate with other Oracle Tuxedo applications or with applications running on other TP systems.

Figure 2-5 Domain Gateways Types



> **Note:**
>
> Oracle Tuxedo CORBA applications also use the Domains component to interoperate with one another and share resources. Only the *TDomain* gateway type—implemented by the `GWTDOMAIN` process—is applicable to Oracle Tuxedo CORBA applications.

2.7.1 Transparency Between Domains

In an Oracle Tuxedo Domains configuration, an administrator can configure which services of a domain are available to other domains in the configuration. The clients and the participating applications themselves do not need to know anything about the Domains configuration. All they need to know is what services or factory objects are available and how to access those services or objects. If applications were to include information about domains, changing configurations would require that the applications be rewritten as well.

2.7.2 Domains Documentation

For more information about the Oracle Tuxedo Domains component, see the following documents:

- "Oracle Tuxedo Administration Processes" in *Introducing Oracle Tuxedo ATMI.*
- Using the Oracle Tuxedo Domains Component Oracle Tuxedo Domains (Multiple-Domain) Servers
- DMADM(5),DMCONFIG(5), GWADM(5),GWTDOMAIN(5) , and UBBCONFIG(5) in File Formats, Data Descriptions, MIBs, and System Processes Reference.

3
Oracle Tuxedo CORBA Components

The following sections describe the Oracle Tuxedo CORBA components and built on the Oracle Tuxedo infrastructure:

> **Note:**
>
> The Oracle Tuxedo CORBA Java client and Oracle Tuxedo CORBA Java client ORB were deprecated in Tuxedo 8.1 and are no longer supported. All Oracle Tuxedo CORBA Java client and Oracle Tuxedo CORBA Java client ORB text references, associated code samples, should only be used to help implement/run third party Java ORB libraries, and for programmer reference only.

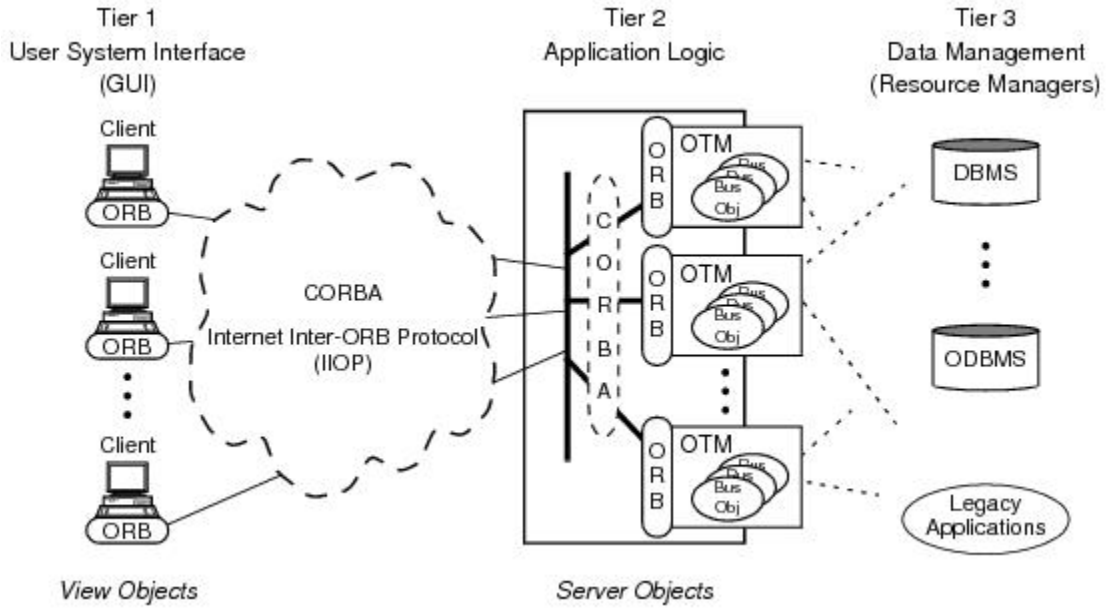
Technical support for third party CORBA Java ORBs should be provided by their respective vendors. Oracle Tuxedo does not provide any technical support or documentation for third party CORBA Java ORBs.

3.1 Oracle Tuxedo CORBA Overview

Oracle Tuxedo CORBA provides businesses and organizations that depend on mission-critical applications with the advantages of the CORBA-compliant programming model, combined with the power, robustness, and proven reliability of the Tuxedo transaction processing technology. Oracle Tuxedo CORBA also taps into the existing Tuxedo infrastructure for transaction management, security, message transport, administration and manageability, and XA-compliant database support. Oracle Tuxedo CORBA combines the ORB model with online transaction processing (OLTP) functions to create a top-of-the-line Object Transaction Monitor (OTM).

As shown in the following figure, an OTM is an example of a 3-tier client/server architecture, where the OTM supports the application logic between the GUI front-end and the back-end resource managers. Examples of resource managers are object-oriented databases, relational databases, message queues, legacy applications, and other back-end services.

Figure 3-1 Tier Client/Server Architecture Using an OTM



By breaking the direct connection between the user interface front-end and the resource managers, an OTM controls all the traffic that links hundreds or thousands or even tens of thousands of clients with run-time objects and the back-end resources. An OTM ensures that global (distributed) transactions are completed accurately, provides load balancing, and improves the overall system performance. An OTM also prestarts pools of objects and provides fault-tolerance. More importantly, an OTM makes an application's server processes independent of the user interface front-end and any resource manager.

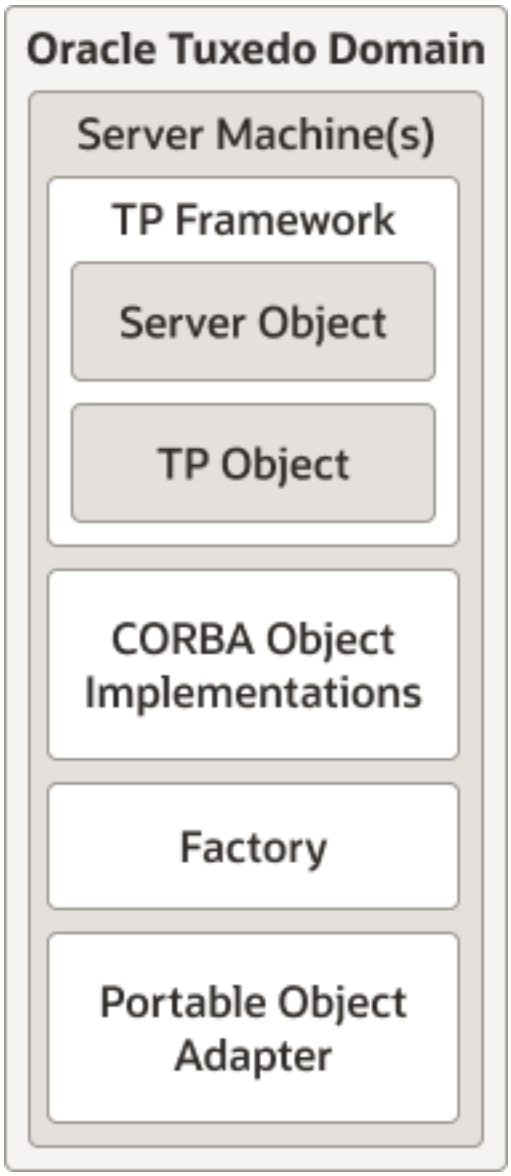
Oracle Tuxedo CORBA is an object application server that runs server-side distributed objects. Besides managing an application's server objects and managing transactions, Oracle Tuxedo CORBA also manages client/server communications.

Object-oriented transactional communications use a highly augmented version of ORB invocations. However, most of the value-added elements are transparent to the programmer: The transactional client/server exchanges look like ordinary exchanges bracketed by start and end transaction calls. The distinguishing factor is that all resource managers and processes invoked through these calls become part of the transaction. An OTM, such as Oracle Tuxedo CORBA, orchestrates the actions of all the participants and makes them act as part of a transaction.

3.2 Oracle Tuxedo TP Framework

The TP Framework component, shown in the following figure, provides a programming model that achieves high levels of performance while shielding the application programmer from the complexities of the CORBA interfaces.

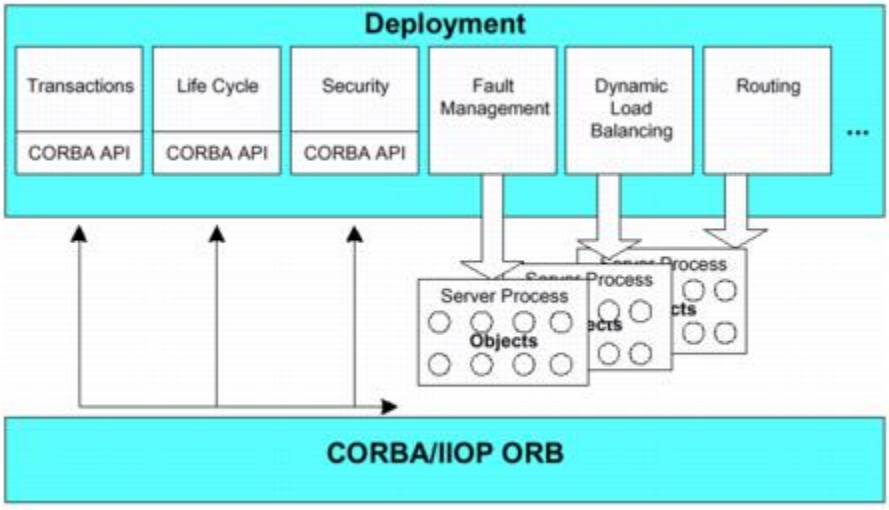
Figure 3-2 The TP Framework



The TP Framework API provides routines that perform many of the functions required in a standard CORBA application. Application programmers are responsible only for writing the business logic of the CORBA application and overriding default actions provided by the TP Framework.

The TP Framework and environmental objects help make development easy. The following figure illustrates the Oracle Tuxedo CORBA development environment:

Figure 3-3 Oracle Tuxedo CORBA Deployment Environment



3.3 Oracle Tuxedo CORBA Architecture

Oracle Tuxedo CORBA consists of the following main components:

Oracle Tuxedo OTM and infrastructure
Provides the services needed to run and administer a distributed CORBA application.

Oracle Tuxedo ORBs
Allows Tuxedo CORBA client and server objects to locate and communicate with one another.

Oracle Tuxedo IIOP Listener/Handler
Allows Tuxedo CORBA clients to reside on intelligent workstations and communicate over a network connection with a CORBA server application.

Oracle Tuxedo environmental objects
Provides a set of objects for helping Tuxedo CORBA clients and servers work with the Tuxedo CORBA environment.

Oracle Tuxedo CORBA object services
Provides object services to Tuxedo CORBA clients.

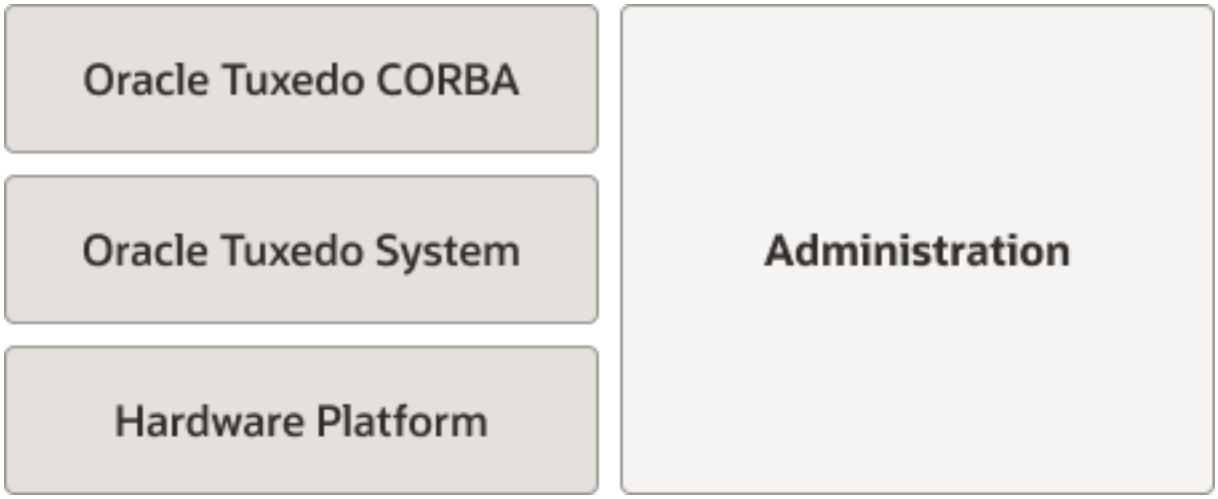
Oracle Tuxedo TP Framework
Provides a programming model for the rapid construction of Tuxedo CORBA server applications.

3.4 Oracle Tuxedo OTM and Infrastructure

The Oracle Tuxedo infrastructure provides the bedrock client/server architecture for both Oracle Tuxedo CORBA and Oracle Tuxedo ATMI. The OTM and infrastructure discussed here and illustrated in the following figure constitute the Oracle Tuxedo CORBA environment, which provides the communication interfaces, transaction

support, and application-processing and administrative services for a distributed CORBA application.

Figure 3-4 The ORB in a CORBA Client/Server Environment



3.4.1 Application Programming Interface

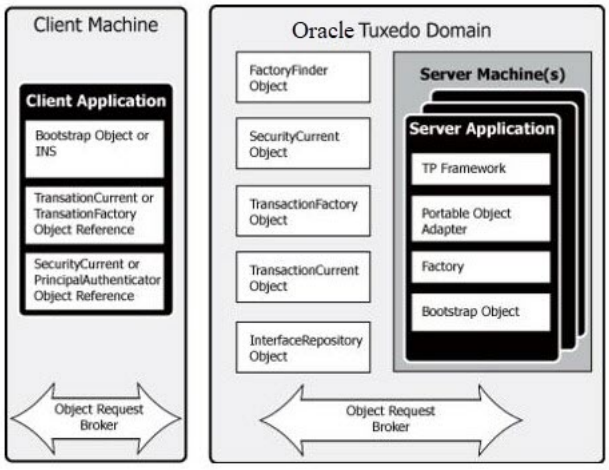
The Oracle Tuxedo CORBA programming interface consists of a C++ server ORB and a C++ client ORB. On the server side, instead of using the CORBA API directly, application programmers use an API that automates many of the functions required in a standard CORBA application. The Oracle Tuxedo CORBA server-side TP Framework component and client-side environmental objects enable programmers to use the deployment environment with a minimal amount of programming. For information about the TP Framework component, see Oracle Tuxedo TP Framework. For information about client-side environmental objects, see Oracle Tuxedo CORBA Environmental Objects.

3.4.2 Application Programming Environment

Application programmers develop a Tuxedo CORBA application as a set of CORBA objects, using the OMG Interface Definition Language (IDL) and, optionally, using standard, off-the-shelf programming tools. These objects communicate with other objects using the CORBA Internet Inter-ORB Protocol (IIOP). The following figure identifies the architectural components of the Oracle Tuxedo CORBA programming environment.

Figure 3-5 Components in an Oracle Tuxedo CORBA Application

Figure 3-5 Components in an Oracle Tuxedo CORBA Application



Oracle Tuxedo CORBA runs the objects in the server processes that it manages. Oracle Tuxedo CORBA can also manage server processes that run Tuxedo ATMI services, thereby allowing programmers to combine object-based and service-based components in the same Tuxedo application.

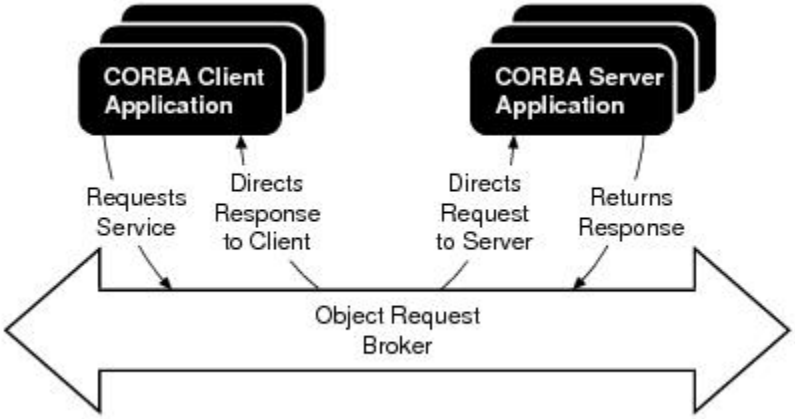
> **Note:**
>
> A Tuxedo *application* has the same meaning as a Tuxedo *domain*. For a definition of an Oracle Tuxedo domain, see Important Oracle Tuxedo Terms and Concepts.

3.5 Oracle Tuxedo ORB Software

The ORB, or Object Request Broker, is a library that enables clients to locate and communicate with servers independently of server location and network connections. The ORB is sometimes referred to as the *object* bus.

Programmers define their object's interface via OMG IDL, and the ORB takes care of the rest. The ORB serves as an intermediary for requests that CORBA clients send to CORBA server applications, so that the client and server do not need to contain information about each other. The following figure shows the relationship between an ORB, a CORBA application client, and a CORBA server application.

Figure 3-6 The ORB in a CORBA Client/Server Environment



Oracle Tuxedo CORBA includes C++ server ORB and C++ client ORB. The ORBs have built-in transaction support, meaning that the CORBA Object Transaction Service (OTS), on which a CORBA OTM is based, is patterned after the XA standard for two-phase commit processing.

The C++ server ORB is linked directly into the Tuxedo CORBA server processes. Other client ORBs can communicate with Oracle Tuxedo CORBA via CORBA's IIOP protocol.

3.6 Oracle Tuxedo IIOP Listener/Handler

The Oracle Tuxedo IIOP Listener/Handler allows a CORBA client residing on a remote machine that does not have a full Oracle Tuxedo server-side installation (that is, a machine that does not support Oracle Tuxedo administration servers or a bulletin board) to be able to communicate with an Oracle Tuxedo CORBA server application. All communication between a remote CORBA client and the CORBA server application takes place over a network connection using the IIOP protocol.

Advantages of remote CORBA clients include:

- Less administrative overhead
- Greater security—keeps clients off the Oracle Tuxedo server machines
- Off loads CPU cycles and decreases process context switches on Oracle Tuxedo server machines
- Smaller footprint

3.6.1 IIOP Listener/Handler Communication

The IIOP Listener/Handler communication architecture involves the following software processes:

CORBA client
A client process that runs on a machine on which the Oracle Tuxedo CORBA C++ client ORB software is installed.

IIOP Listener (ISL)
An Oracle Tuxedo listening process, running on an Oracle Tuxedo server machine, that accepts connection requests from CORBA clients and assigns connections to an IIOP Handler also running on the server machine. It balances client connections across handlers. In addition, an IIOP Listener manages the pool of IIOP Handler processes, starting them in response to load demands.

IIOP Handler (ISH)
An Oracle Tuxedo gateway process, running on the Oracle Tuxedo server machine, that handles IIOP communications between CORBA clients and the Oracle Tuxedo server application. An ISH process resides within the administrative domain of the application and is registered in the local Oracle Tuxedo bulletin board as a client.

Each ISH process can manage multiple CORBA clients. An ISH multiplexes all requests and replies with a particular CORBA client over a single connection.

3.6.2 IIOP Listener/Handler Documentation

For more information about the IIOP Listener/Handler, see the following documents:

- Setting Up an Oracle Tuxedo Application
- ISL(1) in Oracle Tuxedo Command Reference

3.7 Oracle Tuxedo CORBA Environmental Objects

Oracle Tuxedo CORBA provides a set of objects for helping the client work with the Tuxedo CORBA environment; the objects enable client applications to easily log on to a Tuxedo CORBA environment, invoke CORBA objects, and start and end transactions. Like the server-side TP Framework component, these objects interact with the Tuxedo CORBA services.

Here is what these objects do for application clients:

Bootstrap object
The Bootstrap object provides references to the Tuxedo CORBA objects in a Tuxedo CORBA application. An application client can connect to multiple Oracle Tuxedo CORBA applications using different Bootstrap objects. One of the first things that an application client does after startup is to create a Bootstrap object by supplying the host and port number of the IIOP Listener. After the application client contacts an IIOP Listener, the Listener assigns an IIOP Handler to the application client, and the Boostrap object establishes a communication link with the assigned IIOP Handler. The Bootstrap object also provides references to well-known objects that application clients use, such as TransactionCurrent, SecurityCurrent, InterfaceRepository, and FactoryFinder.

CORBA OTS TransactionCurrent object
The CORBA OTS TransactionCurrent object coordinates transaction demarcations with the transaction coordinator.

SecurityCurrent object
The SecurityCurrent object gets the application client's security credentials from the Security Service. The SecurityCurrent object registers the certificate with the IIOP Handler, which uses the certificate to permit or deny invocations.

3.8 Oracle Tuxedo CORBA Object Services

Oracle Tuxedo CORBA provides environmental objects for the C++ programming environment and for the Java programming environment. As of release 8.0, Oracle Tuxedo CORBA also supports the use of the OMG CORBA Interoperable Naming Service (INS) by third-party client ORBs, to obtain initial object references.

Each environmental object provides object services to application clients. Application clients access the environmental objects through a bootstrapping process that accesses the services in a particular Oracle Tuxedo server application. Oracle client ORBs use the Oracle Bootstrap object mechanism, and third-party client ORBs use the CORBA INS mechanism. For more information about bootstrapping an Oracle Tuxedo application, see *Oracle Tuxedo CORBA Programming Reference*.

The Oracle Tuxedo CORBA environmental objects provide the following services:

Object Life Cycle service
The Object Life Cycle service is provided through the FactoryFinder environmental object. The FactoryFinder object is a CORBA object that can be used to locate a factory, which in turn can create object references for CORBA objects. Factories and FactoryFinder objects are implementations of the CORBA Services Life Cycle Service. Oracle Tuxedo CORBA applications use the Object Life Cycle service to find object references.

Security service
The Security service is accessed through either the SecurityCurrent environmental object or the PrincipalAuthenticator object. The SecurityCurrent and PrincipalAuthenticator objects are used to authenticate an application client attempting to access an Oracle Tuxedo server application. The Oracle Tuxedo software provides an implementation of the CORBA Services Security Service.

Transaction service
The Transaction service is accessed through either the TransactionCurrent environmental object or the TransactionFactory object. The TransactionCurrent and TransactionFactory objects allow an application client to demarcate a transaction—that is, begin, suspend, resume, and commit a transaction. The Oracle Tuxedo software provides an implementation of the CORBA Services Object Transaction Service (OTS).

Interface Repository service
The Interface Repository service is accessed through the InterfaceRepository object. The InterfaceRepository object is a CORBA object that contains interface definitions for all the available CORBA interfaces and the factories used to create object references to the CORBA interfaces. The InterfaceRepository object is used with application clients that use Dynamic Invocation Interface (DII).

4

Oracle Tuxedo Product Support and Resources

The following sections describe the documentation and customer-support resources available to Oracle Tuxedo customers:

4.1 About the Oracle Tuxedo Documentation

The Oracle Tuxedo documentation is designed to provide you, the customer, with information at various levels about the Oracle Tuxedo system. Read the entire documentation and select the topics that are most relevant to your immediate requirements.

4.2 Learning Paths

To help you find the information you need, the following table lists user tasks and the documentation appropriate to each.

Table 4-1 Learning Paths

Task	Description	Online-help Topics
Evaluate the product	A high-level overview of the Oracle Tuxedo system.	• Product Overview • Interoperability • Getting Started with CORBA Applications
Install the software	Step-by-step procedures for installing and configuring each of the Oracle Tuxedo system components.	• Installation • Upgrade Information • Migration Information
Design or architect a system	To know (1) Oracle Tuxedo system capabilities, (2) the benefits these capabilities give you, (3) how to incorporate the benefits of the Oracle Tuxedo system into your design, and (4) how to integrate applications in an Oracle Tuxedo environment.	• Interoperability • ATMI Introduction • ATMI and CORBA Programming • System Administration • Reference • ATMI Tutorials • Messages
Write client or server applications	To know how to write, build, configure, and run applications.	The same topics as for design or architect.
Administer the system	To know how to configure, monitor, tune, migrate, and manage the Oracle Tuxedo system.	• System Administration • Migration Information • Interoperability

Table 4-1 (Cont.) Learning Paths

Task	Description	Online-help Topics
Learn about using Oracle Jolt with the Oracle Tuxedo system	To know (1) how to configure and integrate Oracle Jolt with Oracle Tuxedo applications so that Tuxedo services are available to customers on the Internet and (2) how to use, configure, and integrate Oracle Jolt to work with the Oracle Tuxedo system.	• Jolt Documentation

5

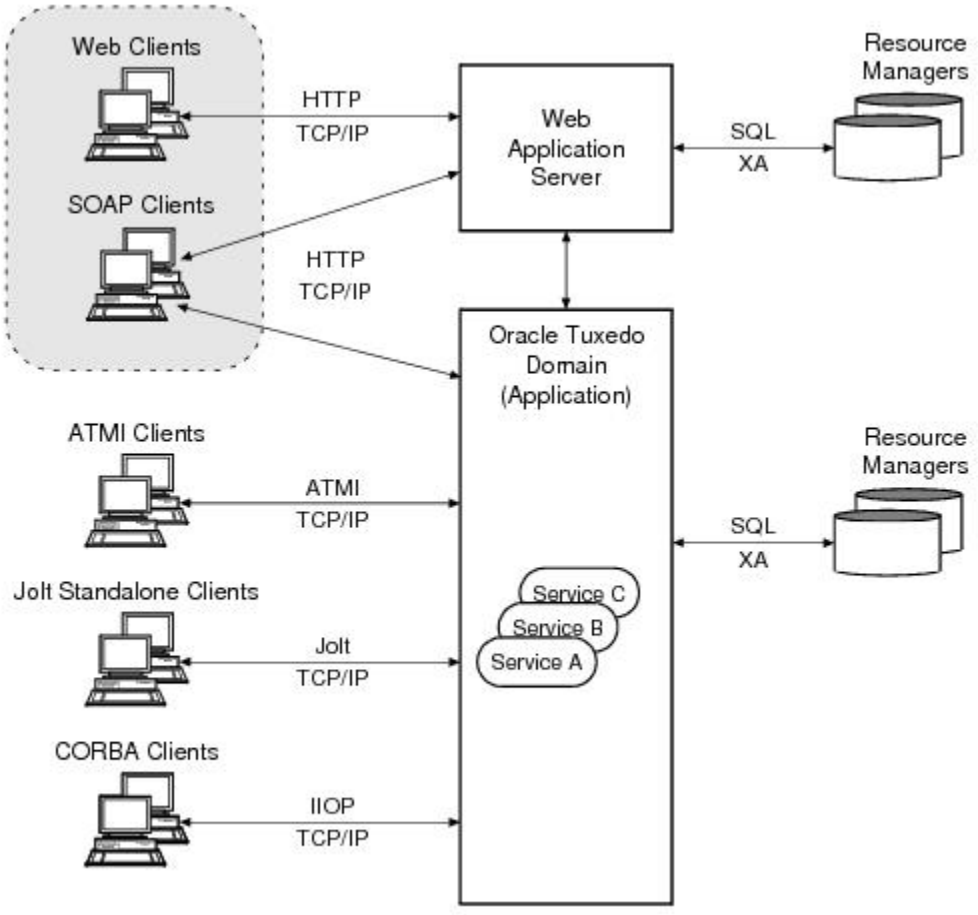
Oracle Tuxedo Web-Accessible Services

The following sections present the many ways of making Oracle Tuxedo services available to Web-based applications:

5.1 What Does Web Accessible Mean?

“Web accessible” means making Oracle Tuxedo application services available to Web-based applications. The following figure illustrates Web client access to Oracle Tuxedo application services.

Figure 5-1 Web Client Access to Oracle Tuxedo Application Services



A Web-based client application may be a simple Web client; for example, a Web browser displaying Hypertext Markup Language (HTML) pages to the end users or a Web client with SOAP engine that can initiating Web Service standard requests.

A Web application server may be a Web server or a cross between a Web server and an application server. The standard definition of a Web server is “a server software system that serves static content to a Web browser by loading a file from a disk and serving it across the network to a user’s Web browser. This entire exchange is mediated by the browser and server talking to each other using Hypertext Transfer Protocol (HTTP).” The standard definition of an application server is “a server software system that occupies a large chunk of computing territory between database servers and the end user, and often connects the two. An application server is sometimes referred to as a type of middleware.” The Oracle Tuxedo system, itself, is essentially two high-performance application servers, a transaction processing application server and an object application server.

A Web application server serves Web clients one of two types of pages (documents): Hypertext Markup Language (HTML) pages or Extensible Markup Language (XML) pages.

5.2 Exposing Tuxedo Services as Web Services

Exposing Tuxedo services as Web services opens the application to the outside world without any application code changes. The application can be broken down into smaller modular components, or shared services, that can be shared by and used as components of distributed Web-based applications. Oracle provides both Tuxedo native solution and Tuxedo-Other Oracle products integrated solution for exposure of Tuxedo Services as Web Services.

5.2.1 Web Services Standards at a Glance

The Web services technologies and programmatic interfaces are being developed by the World Wide Web Consortium (W3C). Web services are based on HTTP and XML as well as the following relatively new XML-based Internet technologies:

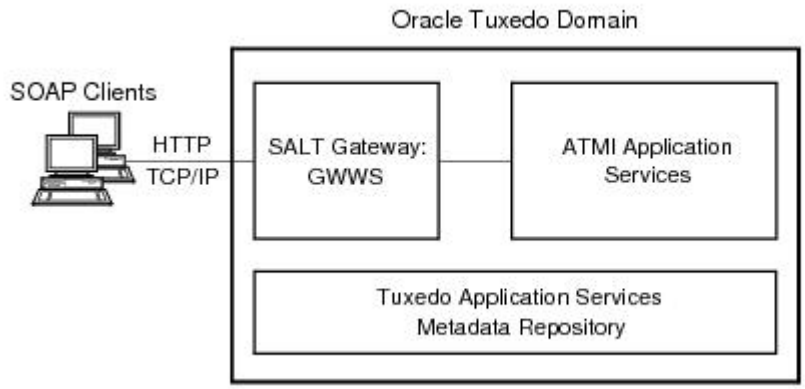
- Web Services Description Language (WSDL)
  The XML-based language for describing (1) the methods provided by a Web service, (2) the input and output parameters of the Web service, and (3) the instructions for connecting to the Web service. WSDL is the standardized way to describe a Web service to clients so that they can invoke it.
- Simple Object Access Protocol (SOAP)
  An XML/HTTP-based protocol for accessing services, objects, and servers in a platform-independent manner. SOAP is the standardized way to transmit data and Web service invocation calls between users and providers of a Web service.
- Universal Description, Discovery, and Integration (UDDI)
  A repository that stores descriptions, in a common XML format, about companies and the services they offer. UDDI is the standardized way for client applications to find a registered Web service and to register a Web service on an Internet server.

Web services communicate with clients, both end-user applications and other Web services, through XML messages transmitted by HTTP. Web services can reside on different computers and can be implemented by vastly different technologies, but they are packaged and transported using standard Internet protocols, thus making them easily accessible by any user on the Internet. For information about the Web services technologies, see W3C - Web Services Activity at http://www.w3.org/2002/ws.

5.3 Exposing Tuxedo Services as Web Services Through Oracle SALT

Oracle Service Architecture Leveraging Tuxedo (SALT) is an add-on product option for Tuxedo, enabling Tuxedo applications to participate in SOA environments. Oracle SALT has two major components: native Web services stack and SCA container. Oracle SALT allows external Web services applications to invoke Tuxedo services as Web services, and Tuxedo applications to invoke external Web services. Oracle SALT does not require any coding to achieve this. Oracle SALT is a native Tuxedo Web service integration solution. Oracle SALT complies with most primary Web Services standards: SOAP 1.1, SOAP 1.2, and WSDL 1.1. With Oracle SALT, Tuxedo applications can be easily exposed as Web Services. The following figure illustrates the principal software components comprising Oracle Tuxedo native Web Services solution to expose Tuxedo application services as Web services.

Figure 5-2 Exposing Tuxedo Application Services as Web Services Through Oracle SALT



Oracle SALT is the preferred product for exposing Tuxedo ATMI services as Web services. It reduces Tuxedo/Web Service integration costs and decreases conversion processes that may exist with other solutions for accessing Tuxedo services. It enables seamless connectivity between Tuxedo applications and external Web service applications.

5.3.1 SALT Gateway Server — GWWS

Oracle SALT provided Tuxedo system server (GWWS), connects with other Web service applications via SOAP over HTTP/S protocol. The GWWS server acts as a Tuxedo gateway process and is managed in the same manner as general Tuxedo system servers. Each GWWS server has bi-directional (inbound/outbound) capability. The GWWS server:

- accepts SOAP requests from Web service applications and issue Tuxedo native calls to Tuxedo services.
- accepts Tuxedo ATMI requests and issues SOAP calls to Web Service applications.

5.3.2 SALT Documentation

For more information about Oracle SALT, see Oracle SALT Documentation.

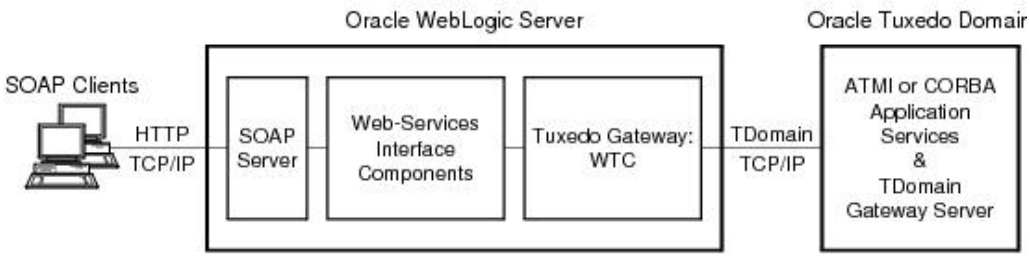
5.4 Exposing Tuxedo Services as Web Services Through Other Oracle Products

This section contains the following topics:

5.4.1 Through Oracle WebLogic Server

The following figure shows the principal software components comprising an Oracle Tuxedo-WebLogic integrated solution to expose Tuxedo application services as Web services.

Figure 5-3 Exposing Tuxedo Application Services as Web Services Through Oracle WebLogic Server



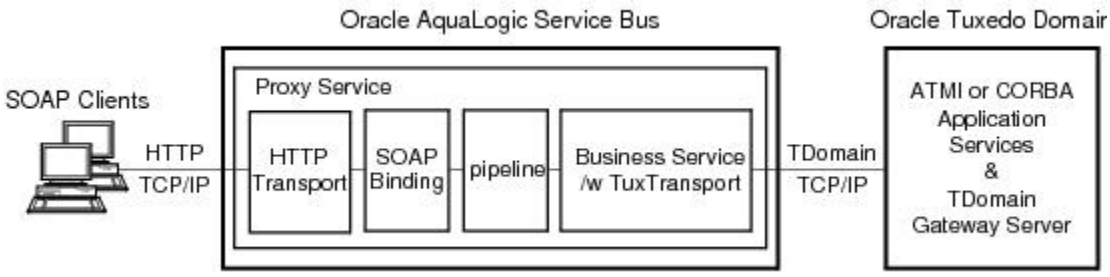
Both Java and non-Java client applications (such as Microsoft .Net Framework clients) can invoke Tuxedo services exposed as Web services through WebLogic Server. The client application assembles a SOAP message describing the Web service it wants to invoke and includes all the necessary data, either in the SOAP body or in a SOAP attachment. The client then sends the SOAP message over HTTP to WebLogic Server, which executes the Web service by performing the following tasks.

1. Calls the associated Tuxedo service via the WTC gateway.
2. Packages the Tuxedo response in a SOAP message.
3. Sends the SOAP message back to the client over HTTP.

5.4.2 Through Oracle AquaLogic Service Bus

The following figure shows the principal software components comprising an Oracle Tuxedo-AquaLogic Service Bus integrated solution to expose Tuxedo application services as Web services.

Figure 5-4 Exposing Tuxedo Application Services as Web Services Through Oracle AquaLogic Service Bus



AquaLogic Service Bus is an Enterprise-class service bus that connects, manages, and mediates interactions between heterogeneous services. To connect between SOAP client applications and Tuxedo ATMI services, you should deploy both SOAP connectivity components and Tuxedo domain connectivity components based on AquaLogic architecture.

5.5 Ceasing Tuxedo Services Using a Web Application Server

Besides being made available as Web Services, Oracle Tuxedo application services are also made available to Web client programs through a Web application server. Applications embedded in the Web Application Servers can access Tuxedo ATMI services through one of the following approaches:

- Making Tuxedo Services Web Accessible Through Oracle Jolt
- Making Tuxedo Services Web Accessible Through Oracle WebLogic Server

5.6 Making Tuxedo Services Web Accessible Through Oracle Jolt

Oracle Jolt provides Internet access to Tuxedo ATMI services for both Web-browser and standalone Java clients. Using Jolt, Java programmers can build client applets and applications that remotely invoke existing and new Tuxedo applications, allowing secure, scalable, intranet/Internet transactions between client and server. Using Jolt, Java programmers can also use HTTP servlets to perform server-side Java tasks in response to HTTP requests. This type of Jolt connectivity enables simple Web clients to access Tuxedo application services through any Web application server that supports generic servlets.

5.6.1 Jolt Class Library

The Jolt class library provides programmers with a set of object-oriented Java language classes for accessing Oracle Tuxedo ATMI services. The class library contains the class files that implement the Jolt API.

5.6.2 Jolt Client Personalities

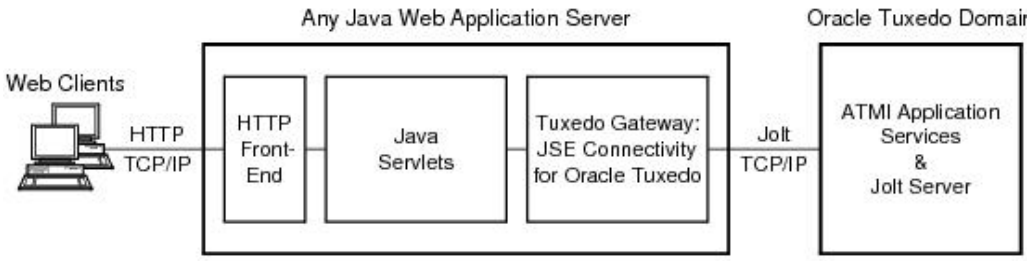
In addition to Jolt applets and Jolt standalone applications, Oracle Jolt supports the following three types of Jolt client personalities for simple Web clients:

- JSE Connectivity for Oracle Tuxedo
- WebLogic Connectivity for Oracle Tuxedo

5.6.2.1 JSE Connectivity for Oracle Tuxedo

This Jolt client personality is a Jolt HTTP servlet, running in a Java Web application server environment (for example, Oracle WebLogic Server), through which simple Web-browser clients can invoke Tuxedo ATMI services. Accessing Tuxedo ATMI services in this manner requires the installation of Jolt class packages jolt.jar and joltjse.jar on the machine running the Web application server.

Figure 5-5 Web Access to Tuxedo Using Jolt JSE Connectivity

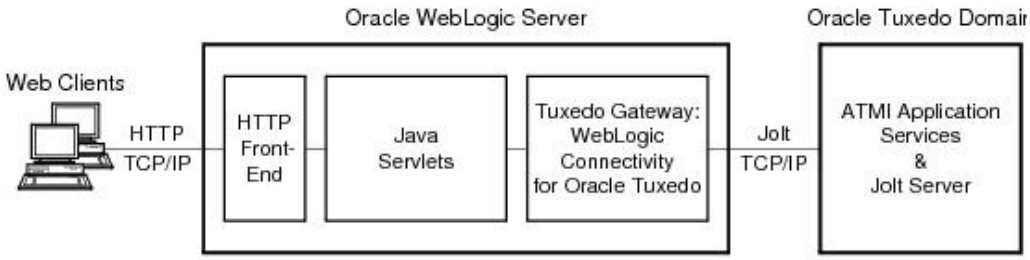


A Jolt HTTP servlet uses Jolt session pool classes to invoke Tuxedo services on behalf of simple browser clients. Thus, the servlet handles all Jolt transactions on the Web server, which enables simple browser clients to invoke Oracle Tuxedo services without directly connecting to the Jolt server and Oracle Tuxedo.

5.6.2.2 WebLogic Connectivity for Oracle Tuxedo

This Jolt client personality is a customized version of Jolt JSE Connectivity for the Oracle WebLogic Server. Accessing Tuxedo ATMI services in this manner requires the installation of Jolt class packages jolt.jar, joltjse.jar, and joltwls.jar on the machine running Oracle WebLogic Server.

Figure 5-6 Web Access to Tuxedo Using Jolt WebLogic Connectivity



> **Note:**
>
> The Jolt client personality “WebLogic Connectivity for Oracle Tuxedo” is also known as “Oracle Jolt for Oracle WebLogic Server.”

5.6.3 Jolt Servers

The Jolt server implementation acts as a proxy for the Jolt client, invoking the Oracle Tuxedo service on behalf of the client. The Jolt server accepts requests from Jolt clients and maps those requests into Oracle Tuxedo service requests.

5.6.4 Jolt Documentation

For information on configuring the Jolt server and the Oracle Tuxedo server to work with Jolt, see Using Servlet Connectivity for Oracle Tuxedo.

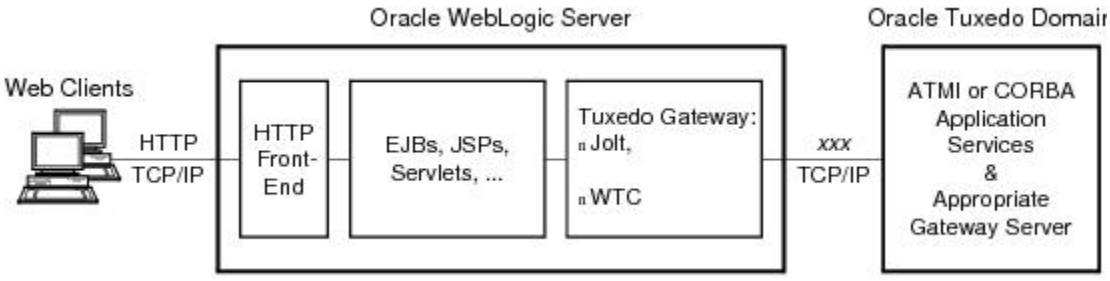
For common client and Web server deployment considerations, see Introducing Oracle Jolt.

5.7 Making Tuxedo Services Web Accessible Through Oracle WebLogic Server

Oracle Tuxedo services have been Web accessible through Oracle WebLogic Server ever since Oracle WebLogic Server release 5.1. The following Oracle Jolt software and Oracle WebLogic Server gateways are central to this accessibility:

- Oracle Jolt for Oracle WebLogic Server—also known as Jolt client personality “WebLogic Connectivity for Oracle Tuxedo” Enables WebLogic Server 5.1 or later EJBs, JavaServer Pages (JSPs), servlets, and other WebLogic Server application servers to call Tuxedo ATMI services on behalf of WebLogic Server Web-browser clients.
- WebLogic Tuxedo Connector (WTC) gateway Enables WebLogic Server 6.1 or later applications, such as servelets and other WebLogic Server applications, to call Tuxedo ATMI services or Tuxedo CORBA C++ objects on behalf of WebLogic Server Web-browser clients.

Figure 5-7 Web Access to Tuxedo Using Jolt or WTC



In addition to WTC, there is support for IIOP connections from WebLogic Server (WLS) via the WLS ORB and the Tuxedo ISL. For details about using Jolt or WTC to achieve

interoperability between Oracle Tuxedo and Oracle WebLogic Server, see Interoperability with Oracle WebLogic Server.