# BEA Tuxedo® Mainframe Adapter for OSI TP

## User Guide

Version 9.1
Document Revised: September 30, 2006

# Copyright



# Restricted Rights Legend



# Trademarks and Service Marks

# Contents

## Introducing BEA Tuxedo Mainframe Adapter for OSI TP



## Managing Transactions and Buffers

## Understanding the DMCONFIG File



## Configuring BEA Tuxedo Mainframe Adapter for OSI TP

# Using the OSI TP Administration Utility

## Utilities Reference



## Manually Upgrading to BEA Tuxedo Mainframe Adapter for OSI TP 9.1



## Index

CHAPTER 1

# Introducing BEA Tuxedo Mainframe Adapter for OSI TP

This section covers the following topics:

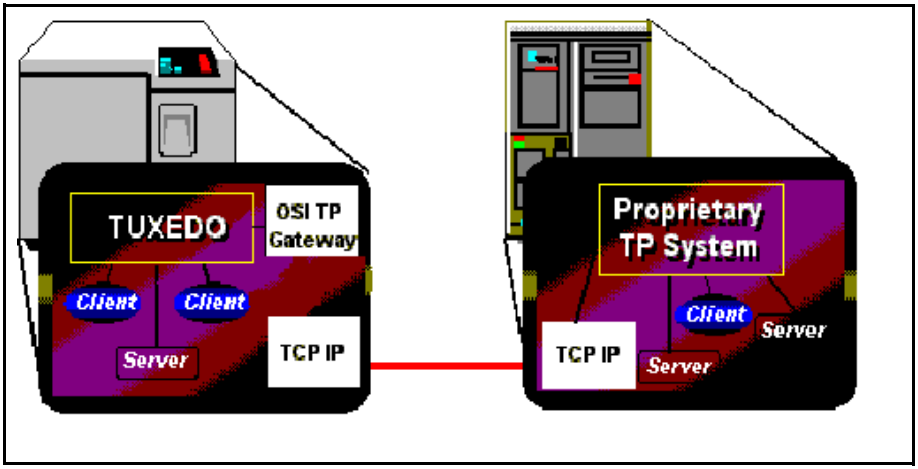


## BEA Tuxedo Mainframe Adapter for OSI TP Overview

BEA Tuxedo Mainframe Adapter for OSI TP is a gateway connectivity product that makes it possible for OLTP application programs on BEA Tuxedo systems to perform global transactions and various non-transactional tasks with application programs in the following environments:

- Other BEA Tuxedo applications. An application (or administrative domain) is a single computer or network of computers that share a single BEA Tuxedo configuration.
- Other systems that implement the Open Group XATMI standard interface and OSI-TP standard protocols. These include Unisys A Series enterprise servers or Unisys OS 2200 enterprise servers that support Open/OLTP software and OSI-TP, Microsoft MTS through Unisys OpenTI, and ICL Open VME.

**Figure 1-1 BEA BEA Tuxedo Mainframe Adapter for OSI TP Communicating with an OSI TP Partner**



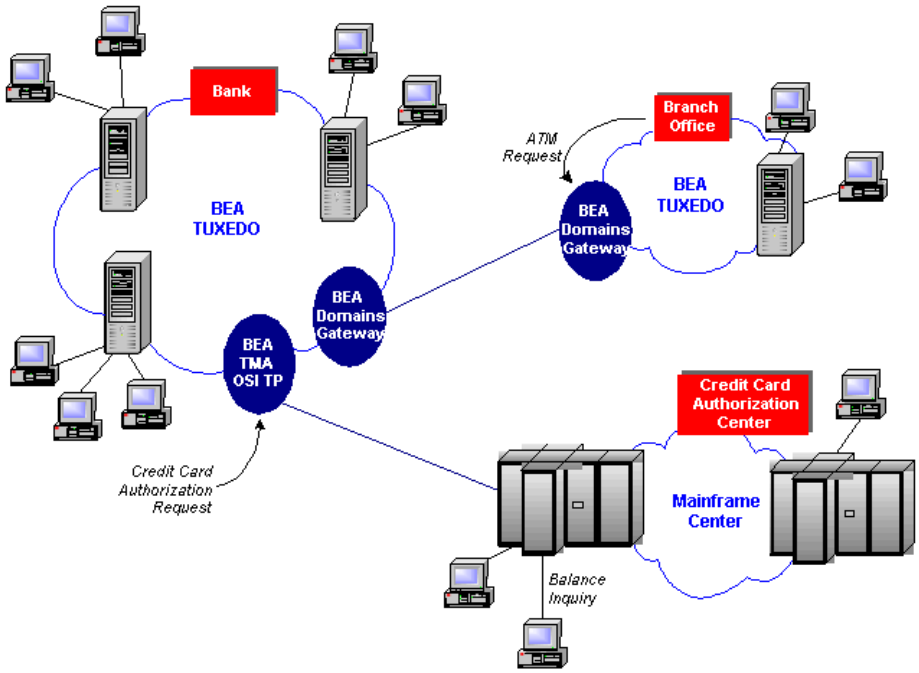

BEA Tuxedo Mainframe Adapter for OSI TP implements the OSI-TP standard, a set of protocols that is used to:

- Establish and support dialogs between application programs on different computers.
- Facilitate commitment and rollback of global transactions that span multiple computers.

The Open Group XATMI standard is an interface that application programs use to communicate with other application programs both inside and outside of global transactions. It supports conversational and request/reply communication styles and is fully implemented by TMA OSI TP.

Data mapping and transformation between the BEA Tuxedo and mainframe environments is easily automated in BEA Tuxedo-based applications with the BEA Tuxedo typed buffer mechanism. This mechanism allows system administrators to predefine how data should be conveyed to the remote application. Application programmers do not need to be aware of this translation; they can simply continue using the buffer types defined for the local application. The TMA OSI TP software is designed to provide transparent access to remote services that reside outside a BEA Tuxedo application. In addition, TMA OSI TP provides remote application programs with access to local services.

**Figure 1-2 BEA TMA OSI TP Sample Environment**




# BEA TMA OSI TP Features

The TMA OSI TP product supports the following features:

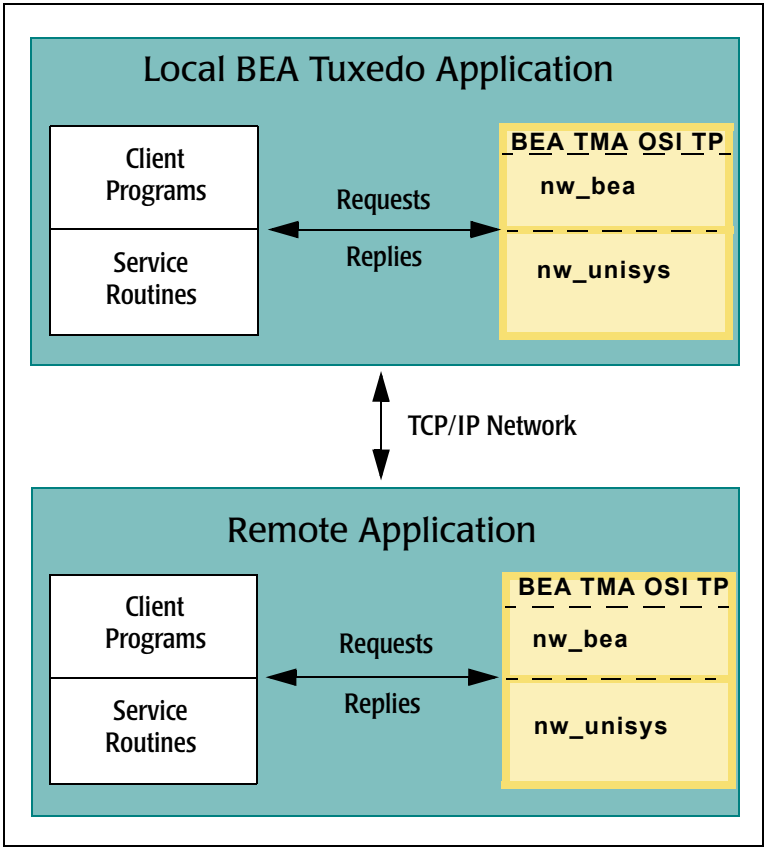
- Tightly-coupled and loosely-coupled transaction processing
- Full array of buffer translation options
- Optional `AUTOPREPARE` feature that allows requests to remote services to be automatically prepared
- Synchronous and asynchronous service requests and replies initiated from either the BEA Tuxedo or mainframe environments
- Conversational service requests and replies initiated from either the BEA Tuxedo or the mainframe environments

- Global Transactions with OSI TP partners
- Event monitoring and reporting
- Dynamic configuration support
- ASN.1 based data conversion
- Link-layer security and global user authentication to participating remote domains
- Multiplexed OSI TP connections with built-in keep alive functionality and enhanced flow control mechanism
- Multiple simultaneous request/response sessions and conversations
- Connections to remote domains
- Transaction recovery for failures occurring during the second phase of commitment
- Automatic suspension of remote services that have failed (through Tuxedo)
- NATIVE A Series COBOL Copybook Data Conversion
- XML Buffer Support
- Data Compression

# BEA Tuxedo and TMA OSI TP Architecture

A BEA Tuxedo application consists of client and server programs that operate across a network of BEA Tuxedo systems. Any client program can request services that are offered by any server program running on any computer in the application. The location of server programs is kept transparent because remote services are mapped to servers in a section of the configuration file. The TMA OSI TP architecture comprises two distinct internal components, `nw-bea` and `nw-unisys`. These two internal components extend the transparent access of the BEA Tuxedo system by sending requests to and receiving requests from remote systems through OSI TP and supporting network software. Figure 1-3 shows how this transparent access works.

**Figure 1-3 Routing Service Calls through BEA TMA OSI TP**




- When local BEA Tuxedo client programs send requests to remote systems, TMA OSI TP transforms those requests into OSI TP messages. When remote systems respond, TMA OSI TP transforms associated OSI TP messages into replies that local client programs can process.
- When remote client programs send OSI TP messages to local applications, TMA OSI TP transforms those messages into requests that local BEA Tuxedo service routines can process. When local service routines send replies, TMA OSI TP transforms those replies into OSI TP messages.

- When commitment protocol is sent for global transactions, TMA OSI TP transforms the protocol and manages the transaction commitment.

The TMA OSI TP software is implemented as an ordinary BEA Tuxedo server group that accepts standard BEA Tuxedo service requests and returns standard replies. The TMA OSI TP server group consists of the following components:

- A server program that includes the TMS service
- An administrative server

One TMA OSI TP server group acts as a gateway to multiple remote systems. The Tuxedo connection manager can either:

- Use multiple communications targets

  If you use multiple communication targets, you have a unique OSI TP endpoint called an *association*. When using unique communications targets, there is one association or connection to the remote system for each call to the remote system. After the call is complete, that association is reused by subsequent calls. There is a connection to the remote system for each simultaneous call made to the remote system. The association is released after a pre-set timer expires, turning unused resources back to the system. This pre-set timer is controlled by parameters specified in the OSI TP tailor file. Refer to "Tuning OSI TP-Specific Tables with the TAILOR File" for more information.

- Configure TMA OSI TP to multiplex all communications per remote system

  To multiplex all communications per remote system, you must have the version of TMA OSI TP that has this feature and your remote system must support OSI TP multiplexing. When using multiplexing, all data is streamed or collected and sent to the remote system. Data messages are sent and received in an efficient manner increasing scalability and performance. The XATMI data representation is still used, but the representation of internal OSI TP data exchanges is optimized.

Some remote targets, such as remote BEA Tuxedo applications, also support BEA TMA OSI TP. In this situation, TMA OSI TP servers associated with the local gateway communicate with TMA OSI TP servers associated with remote gateways through OSI TP.

Other products such as remote Unisys A Series Open/OLTP systems, Unisys OS2200 systems, Unisys OpenTI for MTS interoperability, and ICL TPMS for Open VME provide analogous functionality with which local TMA OSI TP servers can interact.

The TMA OSI TP software maintains its own control information in shared memory, in much the same way that BEA Tuxedo software itself maintains the Bulletin Board. Although TMA OSI TP

accesses the BEA Tuxedo Bulletin Board, BEA Tuxedo does not access TMA OSI TP control information.

To a remote system that supports the Open Group XATMI standard (more specifically, an XATMI application service element), the TMA OSI TP server group appears as a communications resource manager (CRM).

## Multiplexed Flow Control

When the multiplexed protocol is used, flow control is implemented in the following manner. When the amount of data buffered to a particular RDOM exceeds the threshold defined by `StartFlowControlThreshold`, GWOSITP sends a request to Tuxedo to unadvertised all services associated with the particular LDOM and RDOM involved. This allows all call and connect requests already accepted by GWOSITP to complete normally without being aborted. If one or more redundant RDOMs are configured with the same services advertised, then Tuxedo will automatically route new requests to these other RDOMs. If no other RDOM advertises the services, new `tpcall` or `tpconnect` requests will get a TPENOENT error.

When the amount of data buffered to a particular RDOM falls below the threshold defined by `StopFlowControlThreshold`, GWOSITP sends a request to Tuxedo to advertise all services associated with the LDOM and RDOM involved.

Because of the lag time between detecting flow control and unadvertising the services, it is possible for the amount of buffered data to exceed the value of `StartFlowControlThreshold` by a significant margin. Therefore, care should be taken when configuring `StartFlowControlThreshold` with a large value.

**Note:** For more information about the `StartFlowControlThreshold` and `StopFlowControlThreshold` parameters in the TAILOR file, refer to "Tuning OSI TP-Specific Tables with the TAILOR File."

# OSI TP Software Overview

Open Systems Interconnection (OSI) is a computer network architecture developed by the International Organization for Standardization (ISO) in cooperation with other standards Organizations. OSI standards define protocols for communication between diverse computer systems in a network.

# Benefits of OSI

By conforming to OSI standards, computer hardware and software vendors can enable applications on a local computer system to communicate with applications on remote computer systems that are manufactured by another vendor or have a different architecture from the local system.

# OSI Reference Model

The OSI architecture is based upon a framework that divides the networking tasks and requirements into seven layers. The layers are groups of related functions or tasks, intended to make their interfaces and functions easier to understand.

Each layer contains entities that perform specific functions in the communication process. Entities throughout the network that are in the same layer and perform the same functions on different systems are called peer entities. They communicate with each other in a standard way. This is called protocol.

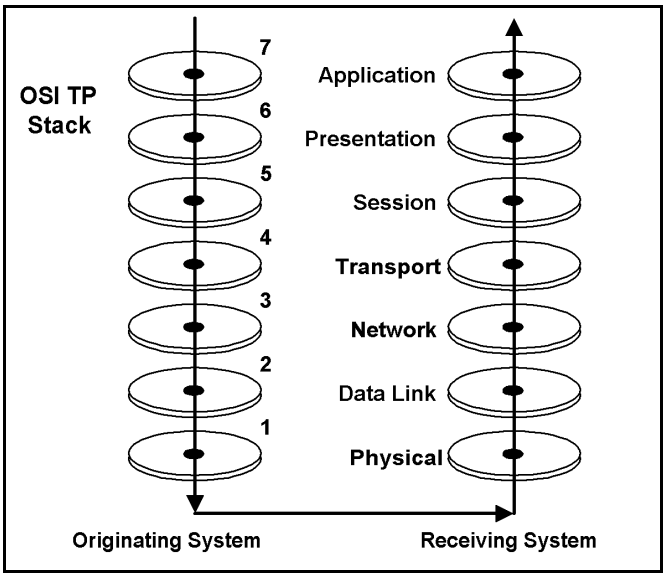
In the OSI Reference Model, peer entities cannot communicate directly. On the sending system, a layer entity uses peer protocols to attach a header containing routing and control information to the message being sent. It then passes this information down to the next layer. That layer adds its own header information and passes it to the next lower layer.

When the message reaches the receiving system, entities in each layer:

- Read and remove the information in the headers added by their peers on the sending system.
- Pass the remaining message up to the next layer.

The following figure illustrates the seven common protocol layers of the OSI Reference Model.

**Figure 1-4 OSI Reference Model**




## Transaction Processing Services

OSI TP works with Tuxedo to provide the following services:

- Marks the beginning of a distributed transaction.
- Coordinates the commitment and rollback of a distributed transaction.
- Performs automatic recovery of all communications paths in the event of a failure.

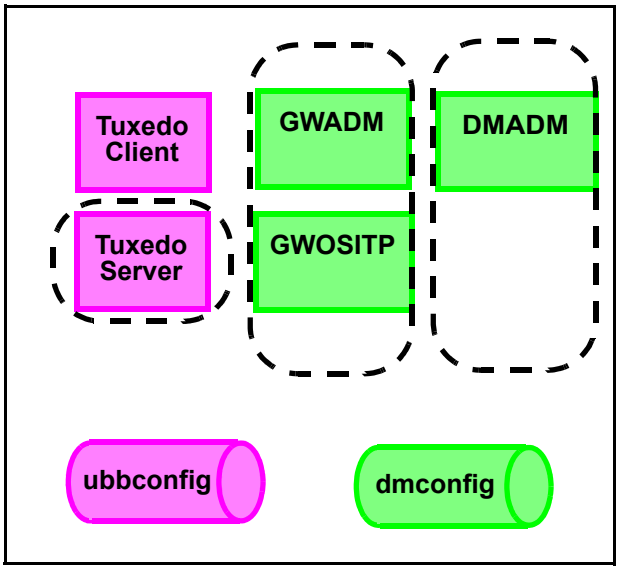
# OSI TP Domains Components

The TMA OSI TP software acts as gateways between BEA Tuxedo systems and other online transaction processing environments. Connections with remote systems are established by configuring TMA OSI TP as an ordinary BEA Tuxedo server group that identifies remote systems and available services.

The TMA OSI TP gateway is composed of several elements that can be configured to provide OSI TP solutions. For the most part, the OSI TP domain is much like the other domain gateways. It uses the `DMADM` and `GWADM` servers provided with BEA Tuxedo for administration.

The following diagram describes each component of the TMA OSI TP product.

**Figure 1-5 Domain Components**




BEA Tuxedo Mainframe Adapter for OSI TP uses the following administrative servers for domain and gateway configuration and administration:

- `DMADM` - Domain administration server
- `GWADM` - Gateway administration server
- `GWOSITP` - OSI TP Domain Gateway

**Note:** The gateway, `GWOSITP`, must be started **AFTER** the other servers.

Specific information on configuring the various sections of a domain are covered in the BEA Tuxedo document, BEA Tuxedo */Domain Guide.*

CHAPTER 2

# Managing Transactions and Buffers

This section covers the following topics:



## Transaction Management

Transaction management provides coordination for the completion of transactions, whether the transaction is successful or not. Application programmers can request the execution of remote services within a transaction, or users at remote domains can request the execution of local services within a transaction. Domains transaction management coordinates the mapping of remote transactions to local transactions, and the sane termination (commitment or rollback) of these transactions.

In the BEA Tuxedo system, a transaction tree is a two-level tree where the root is the group coordinating a global transaction and the branches are other groups on other machines that are involved in the transaction. Each group performs its part of the global transaction independently

from the parts done by other groups. Each group, therefore, implicitly defines a transaction branch. If this BEA Tuxedo transaction branch is controlled by the TMA OSI TP domain, it may contain multiple actual OSI TP transaction branches. The TMA OSI TP gateway controls the mapping between the single BEA Tuxedo transaction root and the many OSI TP transaction branches. The BEA Tuxedo system uses Transaction Manager Servers (TMS) to coordinate the completion of the global transaction and make sure each branch completes.

Domains transaction management can be summarized as follows:

- Gateways generate mappings from a BEA Tuxedo transaction to a network transaction. A new mapping is generated for each BEA Tuxedo transaction or for each incoming network transaction branch.
- Each instantiation handles its own representation of the network transaction tree. Instantiations observe the hierarchical nature of the inter-domain communication.

# Tightly-Coupled and Loosely-Coupled Transactions

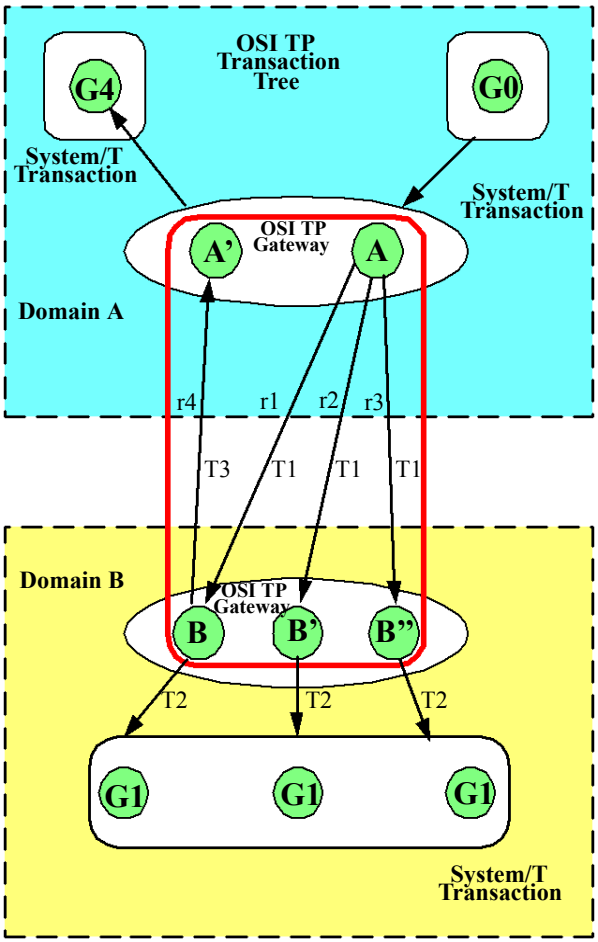
In the Open Group DTP Model, a Transaction Manager (TM) can construct transaction trees by defining either tightly-coupled or loosely-coupled relationships with a Resource Manager (RM). The coupling of the relationships are determined by the way the local service is defined in the `DMCONFIG` file.

A tightly-coupled relationship is one in which the same transaction identifier, XID, is used by all processes participating in the same global transaction and accessing the same RM. This relationship maximizes data sharing between processes; XA-compliant RMs expect to share locks for resources used by processes having the same XID.

The BEA Tuxedo system achieves the tightly-coupled relationship through the group concept; that is, work done by a group on behalf of a given global transaction belongs to the same transaction branch; all the processes are given the same XID. In a loosely-coupled relationship, the TM generates a transaction branch for each part of the work in support of the global transaction. The RM handles each transaction branch separately; there is no sharing of data or of locks between the transaction branches. Deadlocks between transaction branches can occur. A deadlock results in the rollback of the global transaction. In the BEA Tuxedo system, when different groups participate in the same global transaction each group defines a transaction branch; this results in a loosely-coupled relationship. The TMA OSI TP instantiation is user configurable and can provide a tightly-coupled integration that solves this deadlock problem by minimizing the number of transaction branches required in the interoperation between two domains.

Following are diagrams showing loosely-coupled and tightly-coupled integrations and an explanation of each diagram.

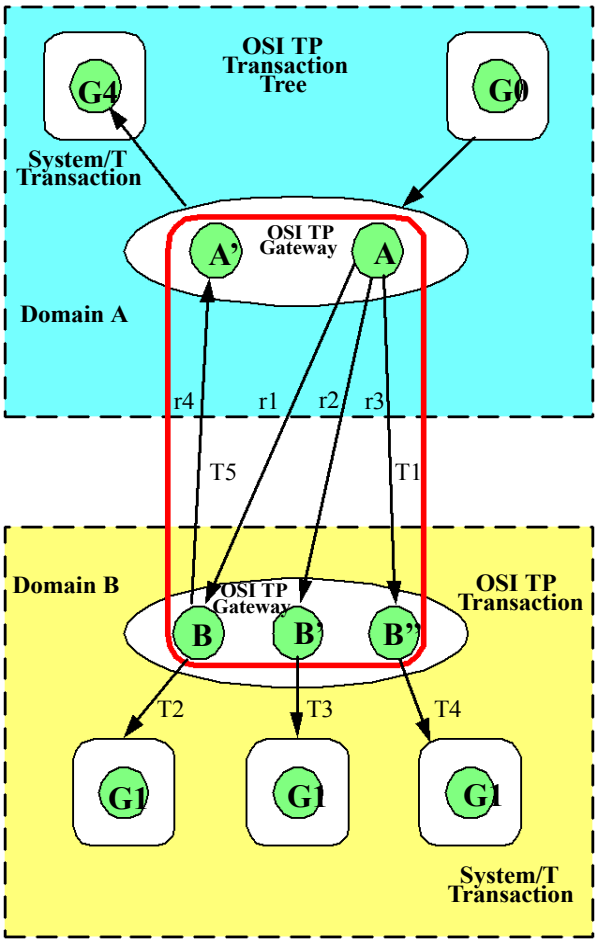
**Figure 2-1 Example of a Tightly-Coupled Integration**




The transaction tree for the tightly-coupled integration shown in Figure 2-1 eliminates the probability for intra-transaction deadlock by minimizing the number of transaction branches required in the interoperation between two domains. Application A makes three requests (r1, r2 and r3) to a remote Domain B. OSI TP sends all three requests mapped to the same OSI TP transaction, T1. On Domain B, the TMA OSI TP gateway checks the `COUPLING` flag for the

remote service and discovers that for service B the COUPLING=TIGHT. In this case all requests for service B belong to the same BEA Tuxedo system transaction. Each request for service B is added to the previous requests and all will have the same BEA Tuxedo XID indicated by T2. Resources in group G1 will not be isolated and changes made by any instantiation of service B for this transaction will be "seen" by the others. Request r4 is mapped to identifier T2 on Domain B, but the Tuxedo domain generates a new branch in its transaction tree (r4: B to A'). This is a new transaction branch on Domain A, and therefore, the gateway generates a new mapping T3, to a new BEA Tuxedo system transaction. The gateway group on Domain A also coordinates group G4, so the hierarchical nature of inter-domain communication is fully enforced with this mapping; group G4 cannot commit before group G1.

**Figure 2-2 Example of a Lossely-Coupled Integration**




The transaction tree for the loosely-coupled integration shown in Figure 2-2 shows group G0 in Domain A coordinating the global transaction started by the client. In this case application A sends three requests (r1, r2, and r3) to a remote Domain B. Like the tightly-coupled case, all three branches are represented by OSI TP transaction T1. On Domain B, the TMA OSI TP gateway checks the `COUPLING` flag for the remote service and sees that service B is `COUPLING=LOOSE`. In this case, the TMA OSI TP gateway generates three BEA Tuxedo system transactions: T2, T3 and T4. Any changes made to G1 are isolated. For example, any changes made by service B can not be “seen” by service B'. When B calls back the A', a new transaction, T5, is generated.

# Global Transactions Across Domains

A global transaction in a single BEA Tuxedo application follows a two-level transaction tree, but a global transaction across domains follows a more complex transaction tree. There are two reasons for this:

- A transaction across domains may involve more domains than can be known from the root domain (where the transaction is controlled), so the structure of the transaction tree cannot be fully known.
- The root domain of a transaction across domains may not be directly connected to all domains in the transaction as would be required for a two-level transaction tree.

The commitment protocol across domains must be hierarchical to handle the complex transaction tree structure. For example, a loop-back service request is made from one domain (Domain A) to another domain (Domain B) and then comes back to be processed in the original domain. The service in Domain B requests another service in Domain A. The transaction tree has two branches at the network level: a branch b1 from A to B and a branch b2 from B to A. Domain A cannot commit the work done on branch b2 before receiving commit instructions from B.

The TMA OSI TP instantiation optimizes `GTRID` mapping by optionally implementing a tightly-coupled relationship. In TMA OSI TP, service requests issued on behalf of the same global transaction are mapped to the same network transaction branch. Therefore, incoming service requests can be mapped to a single BEA Tuxedo transaction. However, the hierarchical structure of inter-domain communication and the inter-domain transaction tree must still be maintained. See Figure 2-1 for an example of a tightly-coupled relationship.

The optimization that TMA OSI TP introduces applies only to a single domain. When two or more domains are involved in the transaction, the network transaction tree contains at least one branch per domain interaction. Therefore, across domains, the network transaction tree remains loosely-coupled. There will be as many branches as there are domains involved in the transaction (even if all branches access the same resource manager instance). Domains gateway groups implement a loosely-coupled relationship because they generate different transaction branches for inter-domain transactions.

See Figure 2-2 for an example of a loosely-coupled relationship. Notice that the gateway still must perform mappings between a BEA Tuxedo transaction and a network transaction, and that the hierarchical nature of the communication between domains must be strictly enforced. Figure 2-2 shows that requests r1, r2, and r3 are mapped to a single TMA OSI TP transaction branch. Therefore, on Domain B only one BEA Tuxedo transaction needs to be generated; request r4 is mapped to an identifier on Domain B, but TMA OSI TP generates a new branch in its

transaction tree (r4: B to A'). This is a new transaction branch on Domain A, and therefore, the gateway generates a mapping to a new BEA Tuxedo transaction. The graph shows that gateway group GW on Domain A also coordinates group G4. Hence, the hierarchical nature of inter-domain communication is fully enforced with this mapping: group G4 cannot commit before group G1.

## Transaction Recovery

OSI TP can recover an entire transaction of individual dialogues if one or more failures occur during the second phase of the two-phase commit process. Failures that occur before the second phase of commitment cause the transaction to roll back automatically. Three types of failure can occur after the second phase of commitment begins. For these types of failures, the following transaction recovery actions can occur:

**Table 2-1 Transaction Recovery Actions for Failures**

| Type of Failure | Transaction Recovery Action |
|---|---|
| Communications Failure | TMA OSI TP automatically re-establishes communications so the transaction can be completed |
| Software Failure | TMA OSI TP maintains communications with the other hosts involved in the transactions until the application is restored. When the application is restored, OSI TP informs the application of the active transactions that need to be completed and the state of each transaction. The application must direct OSI TP to commit or roll back each transaction. |
| System Failure | The application informs TMA OSI TP of the active transactions that need to be completed and the state of each transaction from the secured data log. TMA OSI TP then establishes communications with the other hosts involved so that transactions can be completed. |

# Buffer and Data Translation

The TMA OSI TP software uses typed buffers to transmit and receive data. Full buffer translation is supported for the following buffer types:

- CARRAY

- X_OCTET

**Note:** Null X_OCTET buffers are *not* supported.

- FML and FML32
- STRING
- VIEW and VIEW32
- X_C_TYPE
- X_COMMON
- XML

The following sections introduce procedures that TMA OSI TP follows to process and convert data buffers.

## Layout Conversion for Buffer Types

XATMI (X/Open Application Transaction Manager Interface) mappings to OSI TP are defined in the XATMI ASE (Application Service Element). BEA TMA OSI TP supports this combination. Interoperability using TMA OSI TP requires that remote systems support XATMI ASE. Therefore, Tuxedo-specific buffer types, such as `STRING`, `VIEW`, `FML`, and `CARRAY` may need to be converted into XATMI standard types. BEA TMA OSI TP Gateways perform these layout conversions implicitly.

## ASN.1 Encoding

Abstract Syntax Notation 1 (ASN.1) is an international standard that provides a canonical representation to deal with data representation differences such as byte order, word length, and character sets. The local gateway (`GWOSITP`) encodes input from the local client program. It produces an ASN.1 encoded record that is sent to the remote service. When a reply is received, it is decoded before being returned to the client. Similarly, when remote requests for local services are received by the local gateway, they are decoded from the ASN.1 format. Replies are then encoded for return to the remote client.

## Buffers and Records

The following terms are used to describe input and output data:

**Buffer**

Input or output data as it exists inside the local BEA Tuxedo region. This includes all the buffer types that BEA Tuxedo software supports-both BEA Tuxedo ATMI buffer types and X/Open XATMI buffer types.

**Record**

Input or output data as it exists outside the local BEA Tuxedo region on different kinds of Open/OLTP systems.

These terms make it easier to understand how TMA OSI TP handles input and output data.

# Buffers Received from Local Programs

The TMA OSI TP gateway processes buffers from local programs in the following manner.

1. When TMA OSI TP receives a buffer from a local program, it automatically determines the buffer's type.

   The TMA OSI TP product automatically “types” input buffers that local client programs send to remote services.

   The TMA OSI TP product automatically “types” output buffers that local services return to remote client programs.

2. After TMA OSI TP determines a buffer's type, it refers to the configuration file (`DMCONFIG`) to determine whether the buffer needs to be converted to a different format.

   Client requests sent to remote services may need to be converted to record formats that are meaningful to those services.

   Server responses returned to remote client programs may need to be converted to record formats that are meaningful to those programs.

3. If the configuration indicates that conversion is required, TMA OSI TP transforms the buffer into the record format that is specified in the configuration.

# Records Received from Remote Programs

The TMA OSI TP gateway processes records from remote programs in the following manner.

1. When TMA OSI TP receives a record from a remote system, it refers to the domain configuration (`DMCONFIG`) to determine the record's type and whether the record needs to be converted to a different format.

Client requests from remote client programs may need to be converted to buffer formats that are acceptable to local service routines.

Server responses returned from remote services may need to be converted to buffer formats that are acceptable to local client programs.

2. If the configuration indicates that conversion is required, TMA OSI TP transforms the record into the buffer format that is specified in the configuration.

# Managing Parameters for Buffer and Record Conversion

The TMA OSI TP product provides four configuration parameters you can use to map buffers and records. These parameters are optional.

The following buffer configuration parameters are specified in the `DM_LOCAL_SERVICES` and/or the `DM_REMOTE_SERVICES` sections of the domain configuration file (`DMCONFIG`) as appropriate.

**INBUFTYPE**
: Identifies the type, and in some cases the format, of a buffer received from a Tuxedo client or server. This restricts the buffer type naming space of data types accepted by this service to a single buffer type.

**OUTBUFTYPE**
: Identifies the type, and in some cases the format, of a buffer to be sent to a Tuxedo client or server. Use this parameter to specify the type and format to translate the incoming message to.

**INRECTYPE**
: Identifies the type, and in some cases the format, of a record to be sent to a remote gateway. This parameter is used for buffer and record translation.

**OUTRECTYPE**
: Identifies the type, and in some cases the format, of a record received from a remote gateway.

The definitions of these four parameters depend on whether the service requests originate locally or remotely. The following sections describe these parameters in relation to where the service request originates.

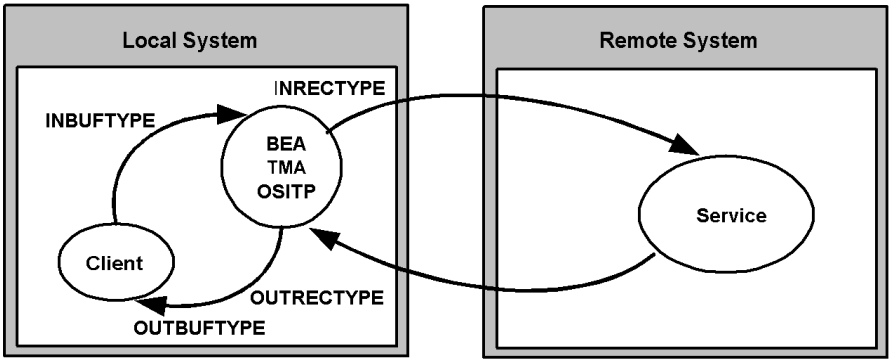
## Parameters for Locally Originated Calls

This section describes in more detail how TMA OSI TP handles service calls that originate locally, within the immediate BEA Tuxedo region. It also explains how the `INBUFTYPE`,

INRECTYPE, OUTRECTYPE, and OUTBUFTYPE parameters can be used to manage the conversion of buffers and records that flow between local client programs and remote services.

In the following figure, a local BEA Tuxedo client program issues a service call that a local TMA OSI TP gateway routes to a remote server through TMA OSI TP.

**Figure 2-3 How Parameters Are Mapped During Locally Originated Calls**




In this situation, the four configuration parameters that are shown in the figure have the following meanings:

- The INBUFTYPE parameter describes the BEA Tuxedo input buffer that the local client program provides to the TMA OSI TP gateway through BEA Tuxedo software. When this parameter is specified, the data type and subtype are verified.
- The INRECTYPE parameter describes the input record that is sent to the service on the remote system.
- The OUTRECTYPE parameter describes the output record that is received from the service on the remote system.
- The OUTBUFTYPE parameter describes the BEA Tuxedo output buffer that is returned to the local client program.

## Guidelines for Mapping Input Buffers to Input Records

The following sections provide detailed information explaining how to use the INBUFTYPE and INRECTYPE parameters for service calls that originate locally (where local client programs call remote services).

**INBUFTYPE**

The INBUFTYPE parameter is used to specify the request buffer type that is provided to a local TMA OSI TP gateway when a local client program issues a service request. Tuxedo uses this information to restrict the buffer type from the local client to only the types defined by the INBUFTYPE parameter.

**INRECTYPE**

The INRECTYPE parameter is used to specify the type, and in some cases the format, of the request record that a particular remote service requires. The TMA OSI TP gateway uses this information to convert BEA Tuxedo request buffers into records that remote services can process.

The INRECTYPE parameter may be omitted if the request buffer is identical, in type and structure, to the request record the remote service expects.

You must specify the INRECTYPE parameter when one of the cases described in the following table is true.

| Case | Explanation |
|---|---|
| The remote service uses an input record that is structured differently from the client program's request buffer | The remote service uses a record that is structured differently from the client program's VIEW, X_C_TYPE, or X_COMMON buffer. For example, the remote service may expect structure members to be sequenced differently. |
| The remote service uses a request record that differs from the client program's request buffer in both type and structure. | The client program uses a BEA Tuxedo FML buffer and the remote service expects a corresponding record with an appropriate structure. |

## Guidelines for Mapping Output Records to Output Buffers

The following sections provide detailed information explaining how to use the OUTRECTYPE and OUTBUFTYPE parameters for service calls that originate locally (where local client programs call remote services and receive output from those services).

**OUTBUFTYPE**

The OUTBUFTYPE parameter is used to specify the type, and in some cases the structure, of the reply buffer that a local client program expects. The TMA OSI TP gateway uses this information to map reply records from remote services to the appropriate kinds of reply

buffers.The TMA OSI TP maps the incoming record to the type and subtype defined by the OUTBUFTYPE parameter.

**OUTRECTYPE**

The OUTRECTYPE parameter is used to specify the type, and in some cases the format, of the reply record that a particular remote service returns to the local TMA OSI TP gateway. The TMA OSI TP maps the incoming record to the type and subtype defined by the OUTBUFTYPE parameter.

The OUTBUFTYPE parameter may be omitted if the remote service returns a reply record that is identical, in type and structure, to the reply buffer the local client program expects.

You must specify the OUTBUFTYPE parameter when one of the cases described in the following table is true.

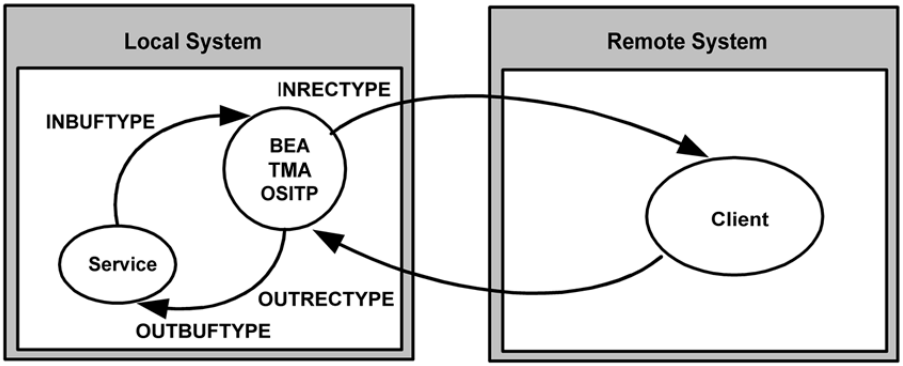
| Case | Explanation |
|---|---|
| The remote service returns a reply record that is structured differently from the reply buffer the local client program expects. | The remote service returns a record that is structured differently from the client program's VIEW, X_C_TYPE, or X_COMMON buffer. For example, the structure members of the output record may be sequenced differently than the structure members of the output buffer. |
| The remote service returns a reply record that differs in both type and structure from the reply buffer the client program expects. | The remote service returns a particular record and the local client program expects a corresponding BEA Tuxedo FML buffer. |

# Parameters for Remotely Originated Calls

This section describes how TMA OSI TP handles service calls that originate on remote computers, outside the local BEA Tuxedo region. It also explains how the INRECTYPE, INBUFTYPE, OUTBUFTYPE, and OUTRECTYPE parameters can be used to manage the conversion of buffers and records that flow between remote client programs and local services.

In the following figure, a remote client program issues a service request that a remote TMA OSI TP gateway routes to the local TMA OSI TP gateway. The gateway receives the request from the network and passes the request to a local BEA Tuxedo server.

**Figure 2-4 How Parameters Are Mapped During Remotely Originated Calls**




In this situation, the four configuration parameters that are shown in the figure have the following meanings:

- The `OUTRECTYPE` parameter describes the record that the remote client sends to the TMA OSI TP gateway.
- The `OUTBUFTYPE` parameter describes the BEA Tuxedo buffer that is provided to the local server.
- The `INBUFTYPE` parameter describes the BEA Tuxedo buffer that the local server returns to the TMA OSI TP gateway.
- The `INRECTYPE` parameter describes the record that the local TMA OSI TP gateway returns to the remote client program.

## Guidelines for Mapping Input Records to Input Buffers

This section provides detailed information explaining how to use the `INRECTYPE` and `INBUFTYPE` parameters for service calls that originate on remote systems (where remote client programs call local services).

**INBUFTYPE**

The `INBUFTYPE` parameter is used to specify the type, and in some cases the structure, of the reply buffer that the TMA OSI TP gateway expects from a local server. This restricts the buffer type naming space of data types accepted by this service to a single buffer type. Because the gateway determines the type of buffer automatically at runtime this parameter is described here for conceptual completeness only.

**INRECTYPE**

The INRECTYPE parameter is used to specify the type, and in some cases the format, of the reply record that the local TMA OSI TP gateway sends to the remote client. You can omit the INRECTYPE parameter if the local server program sends a reply buffer that is identical in type and structure to the reply record the remote client expects.

You must specify the INRECTYPE parameter when one of the cases described in the following table is true.

| Case | Explanation |
|---|---|
| The remote client program requires a reply record that is structured differently from the reply buffer the local service provides. | The remote client program sends a record that is structured differently than the local service's VIEW, X_C_TYPE, or X_COMMON buffer. For example, the structure members of the input record may be sequenced differently than the structure members of the input buffer. |
| The remote client program requires a reply record that differs in both type and structure from the reply buffer the local service provides. | The remote client program requires a particular record and the local service provides a corresponding BEA Tuxedo FML buffer. |

## Guidelines for Mapping Output Buffers to Output Records

This section provide detailed information explaining how to use the OUTBUFTYPE and OUTRECTYPE parameters for service calls that originate on remote computers (where remote client programs call local services and receive output from those services).

**OUTBUFTYPE**

The OUTBUFTYPE parameter specifies the request buffer type that the local TMA OSI TP gateway provides to the local server. The TMA OSI TP gateway uses this information to convert request records from remote clients into buffers that local server programs can process.

**OUTRECTYPE**

The OUTRECTYPE parameter is used to specify the type, and in some cases the format, of the request record a particular remote client program sends to the TMA OSI TP gateway. The TMA OSI TP maps the incoming record to the type and subtype defined by the OUTRECTYPE parameter.

The OUTBUFTYPE parameter may be omitted if the local service's request buffer is identical, in type and structure, to the request record the remote client program provides.

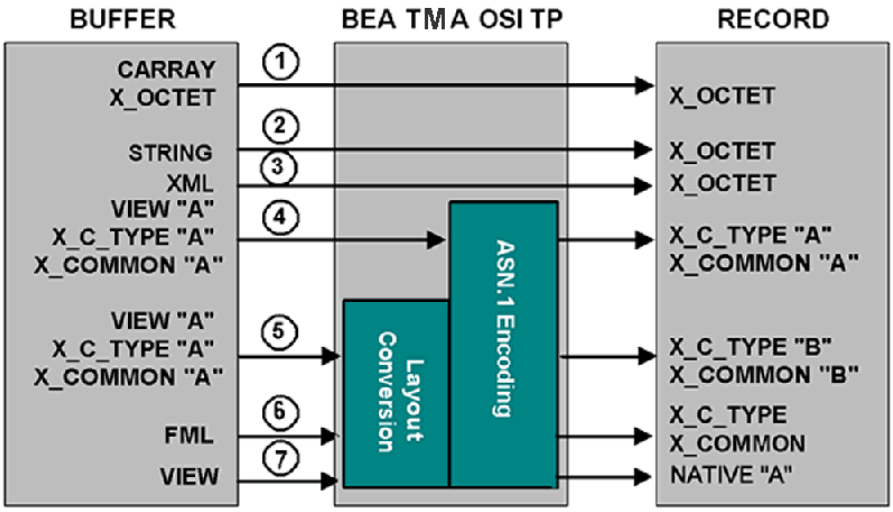
You must specify the OUTBUFTYPE parameter when one of the cases described in the following table is true.

| Case | Explanation |
|---|---|
| The remote client program provides a request record that is structured differently from the local service's request buffer. | The remote client program provides a record that is structured differently than the local service's VIEW, X_C_TYPE, or X_COMMON buffer. For example, the local server program may expect structure members to be sequenced differently. |
| The remote client program provides a request record that differs from the local service's request buffer in both type and structure. | The remote client outputs a record and the local service program expects a corresponding BEA Tuxedo FML buffer. |

# Mapping Buffers to Records

The following figure shows all the possibilities for mapping buffers to records. The TMA OSI TP gateway is responsible for mapping buffers to records based on information it finds in the TMA OSI TP configuration. This mapping occurs for Tuxedo client requests and Tuxedo server responses.

**Figure 2-5 Buffer to Record Mappings**




Following are explanations about the mapping possibilities shown in the figure above and some suggestions for setting the INRECTYPE parameter. The INBUFTYPE parameter is only used for verification purposes and is not discussed here.

1. BEA Tuxedo CARRAY input buffers can be copied to X_OCTET input records. A CARRAY buffer contains raw data that is not converted or translated. The TMA OSI TP gateway automatically sends the CARRAY Tuxedo buffer as X_OCTET to the remote system; there is no need to set the INRECTYPE parameter.

2. BEA Tuxedo STRING input buffers can be mapped to X_OCTET input records. No data conversion or translation is performed. The STRING buffer is copied left to right, up to and including the first NULL character encountered. However, before sending the data, the gateway removes the NULL terminator. The TMA OSI TP gateway automatically sends the STRING buffer as X_OCTET; there is no need to set the INRECTYPE parameter.

3. BEA Tuxedo XML input buffers can be mapped to X_OCTET input records. The XML buffer is copied to an X_OCTET buffer. There is no translation or conversion performed on the data. This can be useful when passing XML data to systems that do not support XML data types.

4. BEA Tuxedo VIEW input buffers can be mapped to X_COMMON or X_C_TYPE input records. Specify the desired record type and the name of this VIEW definition with the INRECTYPE parameter. The TMA OSI TP gateway translates the VIEW into the correct X_COMMON or X_C_TYPE input record.

5. BEA Tuxedo VIEW, X_COMMON or X_C_TYPE input buffers can be mapped to X_COMMON or X_C_TYPE input records-in any combination. However, in this situation, the data structure that the remote service expects (designated as X_COMMON `B' mapping possibilities in Figure 3-3) differs from the data structure the client program uses (designated as VIEW `A' in Figure 3-3). Consequently, you must

   a. Create a VIEW definition for the data structure that the remote service expects.

   b. Specify the desired record type and the name of this VIEW definition with the INRECTYPE parameter.

6. Before a BEA Tuxedo FML input buffer can be sent to a remote service that does not support FML, it must be mapped to one of the following input record types: X_C_TYPE or X_COMMON. Also, you must create a VIEW definition for the input data structure that the remote service expects. Set INRECTYPE to VIEW:viewname. Refer to the BEA Tuxedo online documentation for more detailed information about FML translation.

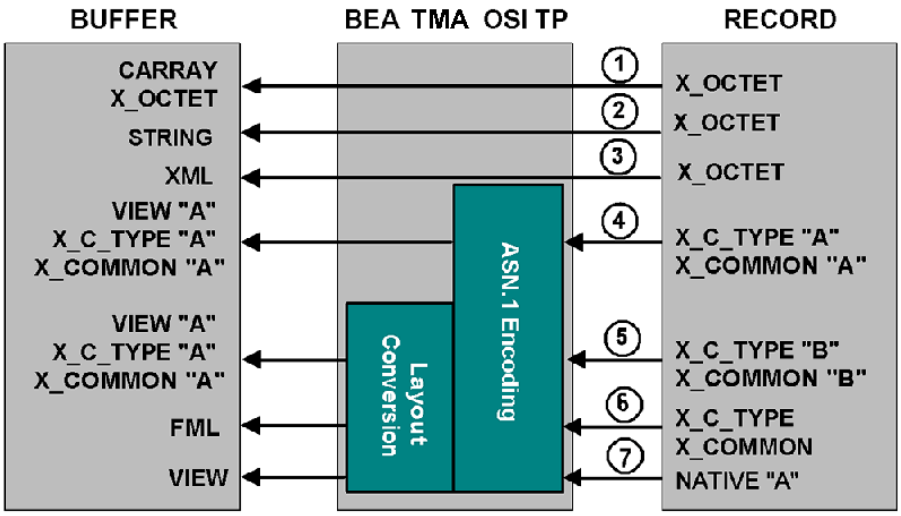
   **Note:** If the source and target VIEW names are different, FML fields must be specified for all VIEW to VIEW conversions that TMA OSI TP performs (For example, VIEW=V10.V --> X_C_TYPE=V10.V does not require FML mapping fields). In other words, any VIEW that is to be used in a VIEW to different VIEW, VIEW to FML, or FML to VIEW conversion must be defined with appropriate FML fields (no dashes in the FNAME column of the VIEW definition). In order for the FML fields to match, you must compile the VIEWs without the -n option specified.

7. The Native-A encoding feature in TMA OSI TP converts Tuxedo VIEW buffer types into NATIVE A record types. This feature moves most of the encode/decode processing from the A-Series to the Tuxedo system.

# Mapping Records to Buffers

The following figure shows all the possibilities for mapping records to buffers. The TMA OSI TP gateway is responsible for mapping records to buffers, based on information it finds in the TMA OSI TP configuration. This mapping occurs for remote client requests and remote server responses.

**Figure 2-6 Record to Buffer Mappings**




Following are explanations about the mapping possibilities shown in the figure above and some suggestions for setting the OUTBUFTYPE parameter (for service calls that originate locally. These suggestions use the OUTBUFTYPE parameter, which controls data translation.

1. Incoming X_OCTET output records can be copied to CARRAY or X_OCTET output buffers. A CARRAY buffer contains raw data that is not converted or translated. Set the OUTBUFTYPE to either CARRAY or X_OCTET; the OUTRECTYPE does not need to be set.

2. Incoming X_OCTET output records can also be copied to STRING output buffers. This creates a string that goes through no conversion and no translation. When going from X_OCTET to STRING, a NULL value is added at the end for the Tuxedo application. The resultant buffer is the length of the original X_OCTET buffer. Since all characters are copied, if the X_OCTET buffer contains null characters, it affects the buffer when later handled as a STRING. The OUTBUFTYPE should be set to STRING.

3. Incoming X_OCTET output records can be mapped to XML output buffers. No translation or conversion is performed on the data. This can be useful when passing XML buffers from systems that do not support XML buffer types to a Tuxedo domain that does support XML data types.

4. Incoming output records can be mapped to identical BEA Tuxedo VIEW output buffers. In this situation, the data structure that the remote service returns is identical to the data structure the

local client program expects. There is no need to create a new VIEW definition. The OUTRECTYPE parameter can be set to VIEW:viewname, for greater type checking, but it is not mandatory.

5. Incoming X_C_TYPE and X_COMMON output records can be mapped to VIEW output buffers-in any combination. However, in this situation, the data structure that the remote service returns (designated as X_C_TYPE `B' in Figure 2-6) differs from the data structure the client program expects (designated as VIEW `A' in Figure 2-6). To facilitate the conversion process, perform the following tasks.
   - Create a VIEW definition for the data structure that the remote service returns.
   - If the name given to the VIEW definition is different from the name that the remote service returns (that is, ATMI buffer subtype), specify the output record type and the name of X_C_TYPE (or X_COMMON)`B' with the OUTBUFTYPE parameter. (By doing this, you override the value the TMA OSI TP requester automatically detects.)
   - Specify the output buffer type and the name of an existing view (VIEW `A' in the figure) specified in the OUTBUFTYPE parameter.

   **Note:** FML Field definitions may be required to map VIEW 'B' to VIEW 'A'.

6. Incoming X_COMMON or X_C_TYPE output records can be mapped to FML output buffers. To facilitate the conversion process, you must perform the following tasks.
   - Create a VIEW definition that describes the data structure that the remote service returns.
   - If the name given to the VIEW definition is different from the name that the remote service returns (that is, the ATMI buffer subtype), specify the output record type and the name of your VIEW definition with the OUTBUFTYPE parameter. (By doing this, you override the value the TMA OSI TP requester automatically detects.)
   - If verification of the FML buffer is desired, set the OUTBUFTYPE to FML or FML32 in the DMCONFIG followed by a colon(:). (Example: OUTBUFTYPE="FML32:")

   **Note:** FML fields must be specified for all FML to VIEW conversions that TMA OSI TP performs. In other words, any VIEW that is to be used in an FML to VIEW conversion must be defined with appropriate FML fields (no dashes in the FBNAME column of the VIEW definition). In order for the FML fields to match, you must compile the VIEWs without the -n option specified.

7. Incoming NATIVE A output records can be mapped to VIEW output buffers.

# Special Cases and Examples of Buffer Conversion

Following are some examples of special cases and considerations for buffer conversion.

- If the incoming buffer (INBUFTYPE) or (OUTRECTYPE) = STRING, CARRAY, X-OCTET, or XML the conversion type must be STRING, CARRAY, X-OCTET, or XML
- Conversion must be between 16-bit buffers, or between 32-bit buffers. You can not convert a 16-bit buffer to a 32-bit buffer, or vice versa.

  Examples:

  ```
  a. OUTRECTYPE=”VIEW:v10”
     OUTBUFTYPE=”VIEW:v12”/* OK */

  b. OUTRECTYPE=”VIEW:v10”
     OUTBUFTYPE=”VIEW32:v13”/* ERROR */

  c. OUTRECTYPE=”VIEW32:v13”
     OUTBUFTYPE=”FML32:”/* OK */

  d. OUTRECTYPE=”VIEW32:v13”
     OUTBUFTYPE=”FML:”/* ERROR */
  ```

- If INBUFTYPE = "FML" or "FML32": or if the buffer being sent to a remote service is of type "FML" or "FML32", then...

  INRECTYPE, of the remote service, **must** be configured to "VIEW" or "VIEW32", respectively.

- INRECTYPE and OUTRECTYPE cannot be "FML" or "FML32".

  ```
  INRECTYPE=”FML:”/* ERROR */
  OUTRECTYPE=”FML32:”/* ERROR */
  ```

- For INBUFTYPE and OUTBUFTYPE, configure buffer type FML as follows:

  ```
  INBUFTYPE=”FML:”
  OUTBUFTYPE=”FML32:”
  ```

  **Note:** A colon is required at the end of keywords FML and FML32.

- INBUFTYPE and OUTRECTYPE must be configured to the same type/subtype as the incoming buffer type/subtype.
- Packed decimal type dec_t cannot be used in views participating in a conversion when either the source or the destination buffer is FML or FML32 or when the source and destination are two different views.

- Packed decimal type `dec_t` cannot be used in views participating in conversion to NATIVE A buffers.
- For views participating in conversion, substitute view fields of type `int` with fields of type `long` or `short`.
- If a View member contains system default values, that member will not be transferred during buffer conversion. To force the transfer, use `"NONE"` in the default Null column of the source View file.

# DMCONFIG Examples for Conversion to View32 and FML

In the following example, incoming buffer `X_COMMON:v10` gets converted to `VIEW:v12` before the request is sent to the service.

**Listing 2-1 Conversion to View32**

```
*DM_REMOTE_SERVICES
TOUPPER12
RDOM=DALNT2
LDOM=DALNT19220
PRIO=66
RNAME=”TOUPPER12”
INBUFTYPE=”X_COMMON:v10”
INRECTYPE=”VIEW:v12”
```

In the following example, incoming buffer `VIEW:v12` gets converted to FML, before the request is sent to the local service, and FML gets converted to `X_C_TYPE:v16` before the reply is returned to the remote client.

**Listing 2-2 Conversion to FML**

```
*DM_LOCAL_SERVICES
OUTRECTYPE=”VIEW:v12”
OUTBUFTYPE=”FML:”
INBUFTYPE=”FML:”
INRECTYPE=”X_C_TYPE:v16”
```

For more information about FML, refer to the BEA Tuxedo Online Documentation at **http://edocs.bea.com/Tuxedo.**

# XML Buffer Support

BEA TMA OSI TP supports Tuxedo XML buffer types. XML buffers are treated like X_OCTET buffers because they are only passed through to the network. No data conversion is performed on the data.

XML buffers are passed to the remote domain as an XML data type unless converted to `X_OCTET`. If the remote domain does not support the XML data type, a decoding error occurs on the remote domain.

CHAPTER 3

# Understanding the DMCONFIG File

Before you configure BEA Tuxedo Mainframe Adapter for OSI TP and set up the gateway configuration, it is helpful to understand the DMCONFIG file.

This section covers the following topics:

- "Overview of the DMCONFIG File"
- "DMCONFIG File Format"
- "DMCONFIG File Sections"
- "Methods for Modifying Configurations"

For detailed instructions on how to configure TMA OSI TP by modifying the DMCONFIG file, refer to "Configuring BEA Tuxedo Mainframe Adapter for OSI TP."

## Overview of the DMCONFIG File

The configuration specified in the DMCONFIG file controls much of the operation of the TMA OSI TP gateway. A sample of this file is provided in the installation directory of your TMA OSI TP product software.

DMCONFIG is the ASCII version of a Tuxedo System/Domain domain configuration file. The DMCONFIG file is parsed and loaded into a binary version by the dmloadcf utility. The binary configuration file, called the BDMCONFIG, contains information used by domain gateways to initialize the context required for communications with other domains. In its monitoring activity, dmadmin uses the binary file (or a copy of it). There is one BDMCONFIG file for each Tuxedo

System/Domain application that uses the /Domain feature. Refer to Processing a Configuration File with the dmloadcf Utility for more information about the binary configuration files.

A DMCONFIG file, and its binary BDMCONFIG counterpart, are analogous to the UBBCONFIG and TUXCONFIG files of a non-/Domain System/T application. The DMCONFIG file extends the definition of a non-/Domain System/T application so that the application becomes a domain.

# OSI TP Application Addresses Used in the DMCONFIG File

OSI TP application address information is used for several parameters in the DMCONFIG file. The address of a Tuxedo application using OSI TP consists of a collection of the names of each of the components described in Table 3-1. These names must coordinate with the remote domain OSI TP implementation.

Starting with eLink OSI TP 4.0, the OSI TP implementation is optimized to eliminate several layers of the protocol stack. As a result, the P_SEL, S_SEL, and T_SEL are not used for routing connections. Connections are routed solely based on the NWADDR parameter. Therefore, TMA OSI TP requires the NWADDR parameter be unique.

- Multiplexed Protocol

  The multiplexed protocol uses a TCP/IP connection. Therefore, if the multiplexed protocol is in use, P_SEL, S_SEL, and T_SEL parameters do not apply and are ignored if configured.

- Non-multiplexed Protocol

  The non-multiplexed protocol sends messages using OSI over TCP/IP (as implemented in RFC1006). When using the non-multiplexed protocol, you may need to specify the P_SEL, S_SEL, and/or T_SEL parameters, depending on the addressing requirements of the remote system. If the P_SEL, S_SEL, and/or T_SEL are configured, TMA OSI TP encodes the selectors for outgoing connections and TMA OSI TP validates incoming connections to verify the selectors are correct. However, TMA OSI TP does not require the use of P_SEL, S_SEL, and T_SEL.

**Table 3-1 OSI TP Application Components**

| Component | Description |
|---|---|
| Application Entity Title (AET) | A dotted integer based on the ISO Object Identifier Based NameForm that uniquely identifies the OSI TP node. See the following description of the AET, "Creating an Application Entity Title." |
| * Presentation Selector (P_SEL) | A logical name for the address of the software that provides the presentation layer services for OSI protocols. |
| * Session Selector (S_SEL) | A logical name for the address of the software that provides the session layer services for OSI protocols. |
| * Transport Selector (T_SEL) | A logical name for the address of the software that provides the transport layer services for OSI protocols. |
| Network Address (NWADDR) | A globally unique computer system address used to identify the OSI TP node. |

* Application components only valid for non-multiplexing communications.

# Creating an Application Entity Title

The Application Entity Title equals the APT (application process title) plus the AEQ (application entity qualifier). Each OSI TP node in your network must have a unique AET. If your site is participating in a global OSI network, you need to contact the OSI registration authority for a valid OSI Object-ID, otherwise, create your own unique AET as described below.

If your site is in a closed network, create an Object ID with at least three "arcs". Each arc is a dotted integer and represents an identifier for the Object ID. A valid OSI TP Object ID has either 0 or 1 for the first arc, and 0,1,2,or 3 for the second arc.

One suggested convention is to set APT to 1.3, followed by the IP address specified in the corresponding NWADDR parameter and set the AEQ to the port number specified in the NWADDR parameter. For example:

```
domain1
  AET="{1.3.123.55.222.51},{12344}"
  NWADDR="123.55.222.51:12344"
```

# DMCONFIG File Format

The format of a domain configuration file is as follows:

- The file is made up of eight possible specification sections. Lines beginning with an asterisk (*) indicate the beginning of a specification section. Each such line contains the name of the section immediately following the *. Refer to "DMCONFIG File Sections" for more information about the sections.

- Parameters are generally specified by: *KEYWORD* = *value*. This sets *KEYWORD* to *value*. Valid keywords are described in the following sections. *KEYWORD*s are reserved; they cannot be used as *value*s unless they are enclosed in quotes.

**Caution:** Enter all parameters on separate lines. The `NWADDR` parameter is the only exception; multiple network address values may be listed in a single line.

- Lines beginning with the reserved word, DEFAULT:, contain parameter specifications that apply to all the lines that follow in the section in which they appear. Default specifications can be used in any of the sections. Defaults can appear more than once in the same section. The format for these lines is:

  DEFAULT: { *KEYWORD1* = *value1* { *KEYWORD2* = *value2* {...}}}

  The values set on this line remain in effect until reset by another DEFAULT: line, or until the end of the section is reached. These values can also be overridden by specifying another value for the parameter on a non-DEFAULT: line. The override parameter setting is valid for that line only; lines that follow revert to the default setting. If DEFAULT: appears on a line without any keywords or values, all previously set defaults are cleared and their values revert to the system defaults.

- If a value is *numeric*, standard C notation is used to denote the base (that is, 0x prefix for base 16 (hexadecimal), 0 prefix for base 8 (octal), and no prefix for base 10 (decimal)). The range of values acceptable for a numeric parameter are given in the description of that parameter.

- If a value is an *identifier*, standard C rules are used. An *identifier* must start with an alphabetic character or underscore and contain only alphanumeric characters or underscores. The maximum allowable length of an identifier is 30 (not including the terminating null). An identifier cannot be the same as any *KEYWORD*.

- A value that is neither an integer number nor an identifier must be enclosed in double quotes. Certain special characters can be included in a string by preceding them with a backslash.

  - "\\" translates to a single backslash
  - "\"\"" translates to a double quote
  - "\n" translates to a new line
  - "\t" translates to a tab
  - "\f" translates to a form feed
  - "\x" (where x is any character other than one of the previously mentioned special characters) translates to x
- Input fields are separated by at least one space (or tab) character
- "#" introduces a comment. A new line ends a comment
- Blank lines and comments are ignored
- Comments can be freely attached to the end of any line
- Lines are continued by placing at least one tab after the new line. Comments cannot be continued

# DMCONFIG File Sections

The DMCONFIG file consists of the following sections and parameters that define new gateway configurations.

- DM_LOCAL_DOMAINS Section

  **Note:** The DM_LOCAL_DOMAINS section must precede the DM_REMOTE_DOMAINS section.
- DM_REMOTE_DOMAINS Section
- DM_OSITPX Section
- DM_ACCESS_CONTROL Section
- DM_LOCAL_SERVICES Section
- DM_REMOTE_SERVICES Section
- DM_ROUTING Section

Following is a sample configuration file and detailed descriptions of the DMCONFIG file sections and the parameters applicable to each section.

# Sample Configuration File

The following sample DMCONFIG file represents non-multiplexing communication.

**Listing 3-1 Sample DMCONFIG File (Non-multiplexing)**

```
*DM_LOCAL_DOMAINS

dalnt8
GWGRP       = OSIGRP
TYPE        = OSITPX
DOMAINID    = "dalnt8"
BLOCKTIME   = 30
DMTLOGDEV   = "D:\tuxedo\log\DMLOG"
SECURITY    = DM_PW  # turns link layer security on
DMTLOGNAME  = DMLOG


*DM_REMOTE_DOMAINS

dal2200 TYPE=OSITPX DOMAINID="dal2200"
openti  TYPE=OSITPX DOMAINID="openti"
icl2    TYPE=OSITPX DOMAINID="icl2"
aseries1 TYPE=OSITPX DOMAINID="aseries1"

*DM_OSITPX

dalnt8
   AET="{1.3.132.61.146},{3}"
   TAILOR_PATH="d:\tuxedo\configs\tailor.txt"
   NWADDR="//dalnt8:102"
   DNS_RESOLUTION=STARTUP # this is the default

dal2200
   AET="{1.3.132.61.46},{3}"
   XATMI_ENCODING="OLTP_TM2200"
       NWADDR="132.61.46.3;132.61.147.1" #redundant IP addresses
```

```
   T_SEL="OSITP"

openti
   AET="{1.3.122.62.103},{209}"
   NWADDR="122.62.103.209:2001"
   OPTIONS=SECURITY_SUPPORTED

icl2
   AET="{1.3.142.60.203},{4}"
   NWADDR="142.60.203.4"
   T_SEL="ICLTP"
   S_SEL="SSEL"
   P_SEL="PSEL"

aseries1
   AET="{1.3.123.55.222},{51}"
   NWADDR="123.55.222.51"
   XATMI_ENCODING="PRELIMINARY"
   T_SEL="0x5453"
   S_SEL="0x3F5C3F"

*DM_ACCESS_CONTROL
mylist  ACLIST = dalnt8, dal2200

*DM_LOCAL_SERVICES
TOUPPERF
  INRECTYPE="VIEW:view10"
  OUTBUFTYPE="FML:"
  COUPLING=LOOSE   #this is the default

TOUPPERF32
  INRECTYPE="VIEW:view10a"
  OUTBUFTYPE="FML32:"
  COUPLING=TIGHT


TOUPPERV
  INBUFTYPE="X_C_TYPE:v10"
```

```
  INRECTYPE="VIEW:upper"
  COUPLING=LOOSE

TOUPPERC OUTRECTYPE="X_OCTET" OUTBUFTYPE="CARRAY"
        INRECTYPE="X_OCTET"
        COUPLING=TIGHT

TOUPPERS  OUTRECTYPE="X_OCTET" OUTBUFTYPE="STRING"
        INRECTYPE="X_OCTET"

TOUPPERX  OUTRECTYPE="STRING" OUTBUFTYPE="STRING"
         INRECTYPE="X_OCTET"

*DM_REMOTE_SERVICES
DEFAULT: TRANTIME=300

ECHOXOCT RNAME="ECHOSRVR" OUTBUFTYPE="X_COMMON:ECHOVIEW" RDOM=aseries1
LDOM=dalnt8
ECHOXCOM RNAME="ECHOSRVR" RDOM=openti LDOM=dalnt8 AUTOPREPARE=Y

ECHOXCTYPE RNAME="ECHOSRVR"
         INBUFTYPE="X_C_TYPE:ECHOVIEW"
         INRECTYPE="X_COMMON:ECHOVIEW"
         OUTBUFTYPE="X_C_TYPE:ECHOVIEW"
         OUTRECTYPE="X_COMMON:ECHOVIEW"
         RDOM=aseries1
         LDOM=dalnt8
         CONV=Y
ECHOVIEW RNAME="ECHOSRVR"
         INBUFTYPE="VIEW:ECHOVIEW"
         INRECTYPE="X_COMMON:ECHOVIEW"
         OUTBUFTYPE="VIEW:ECHOVIEW"
         OUTRECTYPE="X_COMMON:ECHOVIEW"
         RDOM=openti
         LDOM=dalnt8
         TPSUT_TYPE = "PRINTABLESTRING"
         REM_TPSUT="tpmvs"
```

```
*DM_ROUTING
ACCOUNT FIELD = branchid BUFTYPE = "View:account"
        RANGE = "MIN - 1000:aseries1, 1001-3000:openti"
```

The following sample DMCONFIG file represents multiplexing communication.

**Listing 3-2 Sample DMCONFIG File (Multiplexing)**

```
*DM_LOCAL_DOMAINS

dalnt8
GWGRP       = OSIGRP
TYPE        = OSITPX
DOMAINID    = "dalnt8"
BLOCKTIME   = 30
DMTLOGDEV   = "D:\tuxedo\log\DMLOG"
SECURITY    = DM_PW  # turns link layer security on
DMTLOGNAME  = DMLOG


*DM_REMOTE_DOMAINS

openti  TYPE=OSITPX DOMAINID="openti"
aseries1 TYPE=OSITPX DOMAINID="aseries1"

*DM_OSITPX

dalnt8
   EXTENSIONS="MULTIPLEX_POLICY=DEMAND"
   AET="{1.3.132.61.146},{3}"
   TAILOR_PATH="d:\tuxedo\configs\tailor.txt"
   NWADDR="//dalnt8:2020"
   DNS_RESOLUTION=STARTUP # this is the default

openti
   EXTENSIONS="MULTIPLEX=Y"
```

```
   AET="{1.3.122.62.103},{209}"
   NWADDR="122.62.103.209:2001"
   OPTIONS=SECURITY_SUPPORTED

aseries1
   EXTENSIONS="MULTIPLEX=Y"
   AET="{1.3.123.55.222},{51}"
   NWADDR="123.55.222.51:12344"
   XATMI_ENCODING="NATIVE_A_SERIES"

*DM_ACCESS_CONTROL
mylist  ACLIST = dalnt8, openti

*DM_LOCAL_SERVICES
TOUPPERF
  INRECTYPE="VIEW:view10"
  OUTBUFTYPE="FML:"
  COUPLING=LOOSE   #this is the default

TOUPPERF32
  INRECTYPE="VIEW:view10a"
  OUTBUFTYPE="FML32:"
  COUPLING=TIGHT


TOUPPERV
  INBUFTYPE="X_C_TYPE:v10"
  INRECTYPE="VIEW:upper"
  COUPLING=LOOSE

TOUPPERC OUTRECTYPE="X_OCTET" OUTBUFTYPE="CARRAY"
        INRECTYPE="X_OCTET"
        COUPLING=TIGHT

TOUPPERS  OUTRECTYPE="X_OCTET" OUTBUFTYPE="STRING"
        INRECTYPE="X_OCTET"

TOUPPERX  OUTRECTYPE="STRING" OUTBUFTYPE="STRING"
```

```
         INRECTYPE="X_OCTET"

*DM_REMOTE_SERVICES
DEFAULT: TRANTIME=300

ECHOXOCT RNAME="ECHOSRVR" OUTBUFTYPE="X_COMMON:ECHOVIEW" RDOM=dal2200
LDOM=dalnt8
ECHOXCOM RNAME="ECHOSRVR" RDOM=openti LDOM=dalnt8 AUTOPREPARE=Y

ECHOXCTYPE RNAME="ECHOSRVR"
         INBUFTYPE="X_C_TYPE:ECHOVIEW"
         INRECTYPE="X_COMMON:ECHOVIEW"
         OUTBUFTYPE="X_C_TYPE:ECHOVIEW"
         OUTRECTYPE="X_COMMON:ECHOVIEW"
         RDOM=aseries1
         LDOM=dalnt8
         CONV=Y
ECHOVIEW RNAME="ECHOSRVR"
         INBUFTYPE="VIEW:ECHOVIEW"
         INRECTYPE="X_COMMON:ECHOVIEW"
         OUTBUFTYPE="VIEW:ECHOVIEW"
         OUTRECTYPE="X_COMMON:ECHOVIEW"
         RDOM=icl2
         LDOM=dalnt8
         TPSUT_TYPE = "PRINTABLESTRING"
         REM_TPSUT="tpmvs"

*DM_ROUTING
ACCOUNT FIELD = branchid BUFTYPE = "View:account"
        RANGE = "MIN - 1000:aseries1, 1001-3000:openti, *:dal2200"
```

# DM_LOCAL_DOMAINS Section

This section identifies local domains and their associated gateway groups. The section must have an entry for each gateway group (Local Domain). Each entry specifies the parameters required for the domain gateway processes running in that group.

## Format

DM_LOCAL_DOMAINS entries have the following format.

*LDOM* required parameters [*optional parameters*]

where

*LDOM* is an *identifier* value used to name each local domain.

*LDOM* must be unique within a particular configuration. As described in the DM_LOCAL_SERVICES section, *LDOM* is the identifier that connects local services with a particular gateway group.

## Valid Parameters

Following is a list of valid parameters for the DM_LOCAL_DOMAINS section:

**Table 3-2 Parameters for DM_LOCAL_DOMAINS**

| Parameter | Required/Optional | Description |
|---|---|---|
| AUDITLOG | Optional | Name of the audit log |
| BLOCKTIME | Optional | Maximum wait time allowed for a blocking call |
| DMTLOGDEV | Optional | Tuxedo file system that contains the domain transaction log |
| DMTLOGNAME | Optional | Name of the domain transaction log |
| DMTLOGSIZE | Optional | Size of the domain transaction log |
| DOMAINID | Required | Local domain |
| GWGRP | Required | Name of gateway server group |
| MAXRDTRAN | Optional | Maximum number of remote domains that can be involved in a transaction |
| MAXTRAN | Optional | Maximum number of simultaneous global transactions allowed on local domain |
| SECURITY | Optional | Link-level of security for local domain |
| TYPE | Required | Classification of local domain |

## Parameter Definitions

Following is more detailed information about each of the DM_LOCAL_DOMAINS section parameters:

**AUDITLOG = "*string*"**

Specifies the name of the audit log file for this local domain. The audit log feature is activated from the dmadmin command and records all the operations within this local domain. If the audit log feature is active and this parameter is not specified, the file, DMmmddyy.LOG (where mm=month, dd=day, and yy=year), is created in the directory specified by the $APPDIR environment variable or the APPDIR keyword of the *MACHINES section of the TUXCONFIG file.

**BLOCKTIME = *numeric***

Specifies the maximum wait time allowed for a blocking call. The value sets a multiplier of the SCANUNIT parameters specified in the TUXCONFIG file. The value SCANUNIT * BLOCKTIME must be greater than or equal to SCANUNIT and less than 32,768 seconds. BLOCKTIME may need to be increased due to remote network latency or if security is turned on. If this parameter is not specified, the default value is set to the value of the BLOCKTIME parameter specified in the TUXCONFIG file. A timeout always implies a failure of the affected request. Notice that the timeout specified for transactions in the TUXCONFIG is always used when the request is issued within a transaction.

**DMTLOGDEV= *"string"***

Specifies the Tuxedo file system that contains the Domain transaction log (DMTLOG) for this machine. The DMTLOG is stored as a Tuxedo System VTOC table on the device. If this parameter is not specified, the domain gateway group is not allowed to process requests in transaction mode. Local domains running on the same machine can share the same DMTLOGDEV file system, but each local domain must have its own log (a table in the DMTLOGDEV) named as specified by the DMTLOGNAME keyword.

**DMTLOGNAME = *"string"***

Specifies the name of the domain transaction log for this domain. This name must be unique when the same DMTLOGDEV is used for several local domains. If not specified, the default is the string "DMTLOG". The name must be 30 characters or less.

**DMTLOGSIZE = *numeric***

specifies the numeric size, in pages, of the domain transaction log for this machine. It must be greater than 0 and less than the amount of available space on the Tuxedo file system. If not specified, the default is 100 pages.

**DOMAINID = *"string"***

identifies the local domain. DOMAINID must be unique across both local and remote domains. The value of *string* can be a sequence of characters (for example, "BA.CENTRAL01"), or a sequence of hexadecimal digits preceded by "0x" (for example, "0x0002FF98C0000B9D6"). DOMAINID must be 32 octets or fewer in length. If the value is a string, it must be 31 characters or fewer.

**GWGRP = *identifier***

specifies the name of the gateway server group (the name provided in the TUXCONFIG file) representing this local domain. There is a one-to-one relationship between a DOMAINID and the name of the gateway server group, that is, each GWGRP must have its own, unique DOMAINID.

**MAXRDTRAN = *numeric***

specifies the maximum number of remote domains that can be involved in a transaction. It must be greater than 0 and less than 32,768. If not specified, the default is 16.

**MAXTRAN = *numeric***

specifies the maximum number of simultaneous global transactions allowed on this local domain. It must be greater than or equal to 0 and less than or equal to the MAXGTT parameter specified in the TUXCONFIG file. MAXGTT is the maximum number of transactions for all the domains on a given machine. If not specified, the default is the value of MAXGTT.

**SECURITY = {NONE | DM_PW}**

specifies whether link-level security for the local domain is turned on. When this parameter is set to DM_PW, incoming connections from remote domains are authenticated using the passwords defined in the *DM_PASSWORDS section of the BDMCONFIG file. In order to pass userids to the remote domain for user authentication, SECURITY=DM_PW must be set. The default is NONE and indicates that no security is used.

**Note:** This parameter must appear **AFTER** the TYPE=OSITPX parameter.

**TYPE = *identifier***

groups local domains into classes. TYPE can be set to one of the following values: TDOMAIN or OSITPX. The TDOMAIN value indicates that this local domain can only communicate with another Tuxedo System/Domain. The OSITPX value indicates that this local domain communicates with another TP Domain via the OSI TP protocol. Domain types must be defined in the $TUXDIR/udataobj/DMTYPE file. The TMA OSI TP install automatically updates the DMTYPE file with the required type needed.

## See Also

Refer to Table A-1 "DM_LOCAL_DOMAINS SECTION" for more information about setting these parameters through the `dmadmin` utility.

# DM_REMOTE_DOMAINS Section

This section identifies the known set of remote domains and their characteristics.

## Format

`DM_REMOTE_DOMAINS` entries have the following format:

*RDOM* required parameters [*optional parameters*]

where

*RDOM* is an *identifier* value used to identify each remote domain known to this configuration.

*RDOM* must be unique within the configuration.

## Valid Parameters

Following is a list of valid parameters for the `DM_REMOTE_DOMAINS` section:

**Table 3-3 Parameters for DM_REMOTE_DOMAINS**

| Parameter | Required/Optional | Description |
|---|---|---|
| `DOMAINID` | Required | ID of remote domain. |
| `TYPE` | Required | Class of remote domain |
| `CODEPAGE` | Optional | Codepage translation table for NATIVE_A_SERIES encoding. |

## Parameter Definitions

Following is more detailed information about each of the `DM_REMOTE_DOMAINS` section parameters:

**DOMAINID = "*string*"**

identifies a remote domain. DOMAINID must be 32 octets or fewer in length. If the value is a string, it must be 31 characters or fewer. DOMAINID must be unique across remote domains. The value of *string* can be a sequence of characters or a sequence of hexadecimal digits preceded by "0x".

**TYPE** = *identifier*

groups remote domains into classes. TYPE can be set to one of the following values: TDOMAIN or OSITPX. The TDOMAIN value indicates that this remote domain can only communicate with another Tuxedo System/Domain Domain. The OSITPX value indicates that this remote domain communicates with another TP domain via the OSI TP protocol.

**CODEPAGE** = *CPNAME*

specifies the codepage translation table for NATIVE_A_SERIES encoding. The cpname keyword is case sensitive. This parameter is only valid when XATMI_ENCODING=NATIVE_A_SERIES. Refer to Implementing NATIVE-A Encoding for more information.

## See Also

Refer to Table A-2 "DM_REMOTE_DOMAINS SECTION" for more information about setting these parameters through the dmadmin utility.

# DM_OSITPX Section

This section defines the addressing information required by domains of type OSITPX. This section should have at least one entry per gateway group (local domain), and at least one entry per remote domain of type OSITPX. The bridged configuration can have multiple gateways in a local domain.

## Format

DM_OSITPX entries have the following format.

*DOM* required parameters [*optional parameters*]

where

*DOM* is an *identifier* value used to identify a local domain (LDOM) or a remote domain (RDOM) in the DM_LOCAL_DOMAINS section or in the DM_REMOTE_DOMAINS section.

The *DOM* identifier must match a previously defined LDOM in the DM_LOCAL_DOMAINS sections or RDOM in the DM_REMOTE_DOMAINS section.

## Valid Parameters

Following is a list of valid parameters for the DM_OSITPX section:

**Table 3-4 Parameters for DM_OSITPX**

| Parameter | Required/ Optional | Description |
|---|---|---|
| AET | Required for LDOMs and RDOMs | Application entity title |
| DNS_RESOLUTION | Optional for LDOMs | Indicator for when DNS name is resolved |
| EXTENSIONS | Optional for LDOMs and RDOMs | Series of parameters to control operation of the LDOM and RDOM. |
| NWADDR | Required for LDOMs and RDOMs | List of IP addresses with their optional port numbers or a DNS name and its optional port number |
| OPTIONS | Optional for RDOMs | Optional flags to turn on OSI TP features such as security |
| P_SEL | Optional for LDOMs and RDOMs | Logical name of address for software that provides presentation layer services |
| S_SEL | Optional for LDOMs and RDOMs | Logical name of address for software that provides session layer services |
| T_SEL | Optional for LDOMs and RDOMs<br>This parameter is ignored for multiplexed LDOMs. | Logical name of address for software that provides transport layer services |

**Table 3-4 Parameters for DM_OSITPX**

| Parameter | Required/ Optional | Description |
|---|---|---|
| TAILOR_PATH | Optional for LDOMs | Path to optional OSI TP tailor file |
| XATMI_ENCODING | Optional for RDOMs | Version of XATMI protocol |

## Parameter Definitions

Following is more detailed information about each of the DM_OSITPX section parameters:

**AET = "*string*"**

indicates the application entity title that this LDOM or RDOM uses. This address must be unique among all hosts communicating in the OSI TP network. This number matches the local AE Title on the remote (OLTP) node. Refer to OSI TP Domains Components for more information about AETs.

The format accepted for the value of *string* is
"{*object identifier*}, {*integer*}"
The first element represents the APT defined as an object identifier (i.e., a sequence of integer values separated by periods) and the second element represents an AEQ defined as an integer constant, for example,
`AET = "{1.3.15.0.3},{1}".`

For more information about creating an application entity Title, refer to "Creating an Application Entity Title."

**Note:** The braces are part of the syntax and **must** be included within the quotes.

**DNS_RESOLUTION = {STARTUP | RUNTIME}**

indicates whether the DNS name should be resolved when the gateway is started or at runtime. The DNS name is for the network address defined by NWADDR. The runtime option allows support of DHCP networks. When using DNS_RESOLUTION as a runtime option, failed services may occur due to delays in resolving the DNS names. DNS_RESOLTUTION, when configured for an LDOM, indicates the policy for the entire gateway process.

The default is STARTUP.

STARTUP
: indicates the DNS name is resolved when the gateway is booted and never resolved again.

RUNTIME
: indicates the DNS name is resolved again when needed. If three minutes of inactivity has elapsed, the DNS server is pinged for resolution (in case the address changes).

**EXTENSIONS = *"string"***
: controls operations. Valid parameters are separated by a semi-colon ";" and include the following:

For RDOMs, valid parameters are:

"MULTIPLEX={Y|N}"
: The default is N.

Y indicates that this RDOM uses the multiplexed protocol.

N indicates that this RDOM uses the non-multiplexed protocol.

"MULTIPLEXCOMPRESS={Y|N}"
: The default is N.

Y indicates that this RDOM enables data compression.

N indicates that this RDOM does not enable data compression.

"ONLINE={N|Y}"
: The default is Y.

Y indicates that the RDOM is considered online initially. If MULTIPLEX=N and MULTIPLEX_POLICY=NONE for the corresponding LDOM, then a socket connection is established from the LDOM to the RDOM at startup to verify that the RDOM is available. If MUTLIPLEX=Y and MULTIPLEX_POLICY=STARTUP for the corresponding LDOM, then a socket connection is established from the LDOM to the RDOM at startup. If MUTLIPLEX=Y and MULTIPLEX_POLICY=DEMAND for the corresponding LDOM, then a socket connection is established from the LDOM to the RDOM the first time it is needed.

N indicates that the RDOM is considered to be offline initially. No socket connection is established from the LDOM to the RDOM at startup, even if MULTIPLEX_POLICY=STARTUP for the corresponding LDOM. No incoming socket requests are allowed from an offline RDOM.

"RDOMASSOCRETRY=*n*"

The default is given by the TAILOR parameter `RdomAssocRetry` (refer to "Tuning OSI TP-Specific Tables with the TAILOR File.") *n* is a numeric value which represents the time in seconds between association retries for unavailable RDOMs. This value only applies if MULTIPLEX=N. If MULTIPLEX=Y, this value is ignored; the multiplex protocol has its own algorithm for determining how often to retry connection attempts.

For LDOMs, valid parameters are:

MULTIPLEX_POLICY={STARTUP|DEMAND|NONE}

The default is NONE.

`NONE` indicates that this LDOM uses the non-multiplexed protocol.

`STARTUP` indicates that this LDOM uses the multiplexed protocol and a single socket connection is established between the LDOM and each RDOM that has MULTIPLEX=Y and ONLINE=Y at startup.

`DEMAND` indicates the LDOM uses the multiplexed protocol and a single socket connection is established between the LDOM and an RDOM that has MULTIPLEX=Y the first time it is needed.

**NWADDR = "*string*"**

indicates the network address that this `LDOM` or `RDOM` uses and, optionally, the port number. The network address may be either an IP address, if using TCP|IP networks, or a DNS name. The default port number is port 102. For local domains, the `NWADDR` specifies which IP address TMA OSI TP will listen on. For remote domains, the `NWADDR` specifies which network messages will be sent on. You may list multiple network addresses by listing each individual address separated by semicolons if the machine is equipped with multiple network cards. Make sure to enter all the IP addresses on one line and separate them with a semi-colon (;). You may want to configure redundant network paths: up to 8 may be specified.

**Note:** You must explicitly specify a port number greater than or equal to 1024 if *all* the following are true:

- the platform is Unix
- the userid running Tuxedo is *not* root
- this is an LDOM
- MULTIPLEX_POLICY is DEMAND or STARTUP (multiplexing is in use)

Examples:

`"#.#.#.#:port-number"` IP Address
`"//host-name:port-number"` DNS Name
`"//host-name:port-number; //host-name:port-number"` Redundant DNS Names

**OPTIONS = SECURITY_SUPPORTED**

indicates optional parameters for RDOMs. The `SECURITY_SUPPORTED` value indicates that this remote domain supports the OSITP security extension. This value provides backward compatibility and is valid only when describing an RDOM.

**P_SEL = "*string*" or** `"hex digits"`

specifies the logical name for the address of the software that provides the presentation layer services for OSI protocols. The value can be one to 4 ASCII non-control characters (those represented by the hexadecimal numbers 20 to 7E), one to 4 hexadecimal octets, or `NONE` (null). A value of `NONE` is the default. Examples: `"PSEL", "0x3F5C'`

**Note:** This value is ignored for multiplexed connections. For additional information about when to use this parameter for defining application addresses, refer to "OSI TP Application Addresses Used in the DMCONFIG File."

**S_SEL = "*string*" or** `"hex digits"`

specifies the logical name for the address of the software that provides the session layer services for OSI protocols. The value can be one to 16 ASCII non-control characters (those represented by the hexadecimal numbers 20 to 7E), one to 16 hexadecimal octets, or `NONE` (null). A value of `NONE` is the default. Examples: `"SSEL", "0x3F5C3F'`

**Note:** This value is ignored for multiplexed connections. For additional information about when to use this parameter for defining application addresses, refer to "OSI TP Application Addresses Used in the DMCONFIG File."

**T_SEL = "*string*" or** `"hex digits"`

represents the logical name for the address of the software that provides the transport layer services for OSI protocols. The value can be one to 32 ASCII non-control characters (those represented by the hexadecimal numbers 20 to 7E), one to 32 hexadecimal octets, or `NONE` (null). Examples: `"OSITP", "0x5453"`

**Note:** This value is ignored for multiplexed connections. For additional information about when to use this parameter for defining application addresses, refer to "OSI TP Application Addresses Used in the DMCONFIG File."

**TAILOR_PATH = "*string*"**

indicates the full path to the optional OSI TP tailor file used for tuning OSI TP-specific tables. Double quotes are required. If not specified, preset defaults are used. This parameter is valid only when describing an LDOM. Refer to Tuning OSI TP-Specific Tables with the TAILOR File for more information.

**XATMI_ENCODING = {CAE | PRELIMINARY | OLTP_TM2200 | NATIVE_A_SERIES}**

specifies the version of the XATMI protocol used to communicate with a remote application. This parameter is only valid for an RDOM. Valid values are:
`CAE` (default)
`PRELIMINARY` (used specifically with Unisys MCP OLTP systems)
`OLTP_TM2200` (used specifically with Unisys TM 2200 systems)
`NATIVE_A_SERIES` (used specifically with Unisys MCP OLTP systems that support this encoding type)

## See Also

Refer to Table A-3 "DM_OSITPX SECTION" for more information about setting these parameters through the `dmadmin` utility.

# DM_ACCESS_CONTROL Section

This section specifies the access control lists used by local domain.

## Format

`DM_ACCESS_CONTROL` entries have the following format.

*ACL_NAME* required parameters

where

*ACL_NAME* is a (*identifier*) name used to identify a particular access control list; it must be 15 characters or less in length.

## Valid Parameters

Following is a list of valid parameters for the `DM_ACCESS_CONTROL` section:

**Table 3-5 Parameter for DM_ACCESS_CONTROL**

| Parameter | Required/Optional | Description |
|---|---|---|
| ACLIST | Required | List of remote domain names |

## Parameter Definitions

Following is more detailed information about the DM_ACCESS_CONTROL section parameter:

**ACLIST = *identifier* [,*identifier*]**

indicates one or more remote domain names (RDOM) separated by commas. The wildcard character (*) can be used to specify that all the remote domains defined in the DM_REMOTE_DOMAINS section can access a local domain.

## See Also

Refer to Table A-7 "DM_ACCESS_CONTROL SECTION" for more information about setting these parameters through the dmadmin utility.

# DM_LOCAL_SERVICES Section

This section provides information on the services exported by each local domain. This section is optional and if it is not specified then all local domains defined in the DM_LOCAL_DOMAINS section accept requests to all of the services advertised by the Tuxedo System/Domain application. If this section is defined then it should be used to restrict the set of local services that can be requested from a remote domain.

## Format

DM_LOCAL_SERVICES entries have the following format.

*service* [*optional parameters*]

where

*service* is the (identifier) local name of the exported service, and it must be 15 characters or fewer in length.

This name corresponds to a name advertised by one or more servers running with the local Tuxedo System/Domain application. Notice that exported services inherit the

default or special properties specified for the service in an entry in the SERVICES section of the TUXCONFIG file. Some of these parameters are: LOAD, PRIO, AUTOTRAN, ROUTING, BUFTYPE, and TRANTIME.

## Valid Parameters

Following is a list of valid parameters for the DM_LOCAL_SEVICES section:

**Table 3-6 Parameters for DM_LOCAL_SERVICES**

| Parameter | Required/Optional | Description |
|---|---|---|
| ACL | Optional | Name of access control list |
| COUPLING | Optional | Indicator for type of coupling |
| INBUFTYPE | Optional | Type and subtype of buffer returned by local service |
| INRECTYPE | Optional | Type and format of the reply buffer expected by remote client |
| LDOM | Optional | Name of local domain exporting a service |
| OUTBUFTYPE | Optional | Type and format of request buffer expected by local service |
| OUTRECTYPE | Optional | Type and format of request buffer expected by local service |
| RNAME | Optional | Name of service exported to remote domains |

## Parameter Definitions

Following is more detailed information about the DM_LOCAL_SEVICES section parameters:

**ACL = *identifier***

specifies the name of the access control list (ACL) to be used by the local domain to restrict requests made to this service by remote domains. The name of the ACL is defined in the DM_ACCESS_CONTROL section. If this parameter is not specified then access control is not performed for requests to this service.

**COUPLING = {TIGHT | LOOSE}**

specifies service (transaction) coupling to be tight or loose when requests for this local service come from the same remote domain. The default is LOOSE. This means data base

updates made by the first request to this local service cannot be seen by the second request to the local service even though they are involved in the same global transaction. By making this value TIGHT, multiple calls to the same service from the same domain are tightly-coupled. Data base updates made by the first request can be seen by the second request. This option is only available when duplicate service requests come from the same RDOM. When the service requests are from different RDOMs, the requests are always loosely-coupled.

**INBUFTYPE = type[:subtype]**

specifies the type and subtype of the buffer. INBUFTYPE is used to enforce stronger type checking. In the DM_LOCAL_SERVICES section, the TYPE parameters are defined in reference to where the remote request originates. Refer to Managing Parameters for Buffer and Record Conversion for more information about these parameters.

**INRECTYPE = type[:subtype]**

specifies the type, and in some cases, the format of the reply buffer that a particular client requires. This parameter can be omitted if the local service sends a buffer that is identical in type and structure to the buffer the remote client expects. If you do not specify INRECTYPE, the type of buffer is unchanged. In the DM_LOCAL_SERVICES section, the TYPE parameters are defined in reference to where the remote request originates. Refer to Managing Parameters for Buffer and Record Conversion for more information about these parameters.

**LDOM = identifier**

specifies the name of the local domain exporting this service. If this keyword is not specified then all the local domains defined in the DM_LOCAL_DOMAINS section accept requests to this local service.

**OUTBUFTYPE = type[:subtype]**

specifies the type, and in some cases, the format of the request buffer that a particular local service expects. This parameter can be omitted if the remote client sends a buffer that is identical in type and structure to the buffer the local service expects. If you do no specify OUTRECTYPE, the type of the buffer is unchanged. In the DM_LOCAL_SERVICES section, the TYPE parameters are defined in reference to where the remote request originates. Refer to Managing Parameters for Buffer and Record Conversion for more information about these parameters.

**OUTRECTYPE = type[:subtype]**

specifies the type and subtype of the buffer sent by the remote client. This parameter is used to enforce stronger type checking. In the DM_LOCAL_SERVICES section, the TYPE parameters are defined in reference to where the remote request originates. Refer to

Managing Parameters for Buffer and Record Conversion for more information about these parameters.

**RNAME = "string"**

specifies the name of the service exported to remote domains. This name is used by the remote domains for request to this service. If this parameter is not specified, the local service name is used for the request.

## See Also

Refer to Table A-4 "DM_LOCAL_SERVICES SECTION" for more information about setting these parameters through the `dmadmin` utility.

# DM_REMOTE_SERVICES Section

This section provides information on services *imported* and available on remote domains.

## Format

`DM_REMOTE_SERVICES` entries have the following format.

*service* [*optional parameters*]

where

*service* is the (*identifier*) name used by the local Tuxedo System/Domain application for a particular remote service.

Remote services are associated with a particular remote domain.

## Valid Parameters

Following is a list of valid parameters for the `DM_REMOTE_SERVICES` section:

**Table 3-7 Parameters for DM_REMOTE_SERVICES**

| Parameter | Required/Optional | Description |
|---|---|---|
| `AUTOPREPARE` | Optional | Indicator to automatically prepare calls from tpcall. |
| `CONV` | Optional | Indicator that remote service is conversational. |
| `INBUFTYPE` | Optional | Type and subtype of buffer sent to the remote service. |

**Table 3-7 Parameters for DM_REMOTE_SERVICES**

| Parameter | Required/Optional | Description |
|---|---|---|
| INRECTYPE | Optional | Type and format of the request buffer expected by remote service. |
| LDOM | Optional | Name of local domain exporting a service. |
| OUTBUFTYPE | Optional | Type and format of reply buffer expected by local client. |
| OUTRECTYPE | Optional | Type and format of reply buffer returned by remote client. |
| RDOM | Optional | Name of remote domain responsible for execution of service. |
| REM_TPSUT | Optional | TP service user title. |
| RNAME | Optional | Name of service exported to remote domains. |
| ROUTING | Optional | Routing criteria used for data-dependent routing. |
| TPSUT_TYPE | Optional | Type for which the remote TP service user title is to be encoded. |
| TRANTIME | Optional | Default time-out value in seconds for transaction automatically started for associated service. |

## Parameter Definitions

Following is more detailed information about the DM_REMOTE_SERVICES section parameters:

**AUTOPREPARE = {N | Y}**

allows a single tpcall() involved in a global transaction to this remote service to automatically prepare the call. This optimization reduces the two-phase commit process to a single step. The remote OSITP domain must support this feature. The default is N.

**CONV = {Y | N}**

specifies whether or not the remote service is a conversational service. Use Y to specify the remote service is a conversational service. Use N to specify the remote service is not a conversational service. The default value is N.

**INBUFTYPE = type[:subtype]**

specifies the type and subtype of the buffer allocated by the client. This parameter is used to enforce stronger `type` checking. In the `DM_REMOTE_SERVICES` section, the TYPE parameters are defined in reference to where the local request originates. Refer to Managing Parameters for Buffer and Record Conversion for more information about these parameters.

**INRECTYPE = type[:subtype]**

specifies the type, and in some cases, the format of the request buffer that a particular remote service requires. This parameter can be omitted if the local client sends a buffer that is identical in type and structure to the buffer the remote service expects. If you do not specify `INRECTYPE`, the type of buffer is unchanged. In the `DM_REMOTE_SERVICES` section, the `TYPE` parameters are defined in reference to where the local request originates. Refer to Managing Parameters for Buffer and Record Conversion for more information about these parameters.

**LDOM = *identifier***

specifies the name of a local domain in charge of routing requests to this remote service. The gateway group associated with the local domain advertises *service* in the Tuxedo System/Domain Bulletin Board. If this parameter is not specified, then all the local domains are able to accept requests to this remote service. The service request is then redirected to a remote domain of the same type (see the following definition for `RDOM` keyword).

**OUTBUFTYPE = type[:subtype]**

specifies the type, and in some cases, the format of the reply buffer that a particular local client expects. This parameter can be omitted if the remote service returns a buffer that is identical in type and structure to the buffer the local client expects. If you do no specify `OUTRECTYPE`, the type of the buffer is unchanged. In the `DM_REMOTE_SERVICES` section, the `TYPE` parameters are defined in reference to where the local request originates. Refer to Managing Parameters for Buffer and Record Conversion for more information about these parameters.

**OUTRECTYPE = type[:subtype]**

specifies the type and subtype of the buffer sent by the remote service. This parameter is used to enforce stronger type checking. In the `DM_REMOTE_SERVICES` section, the `TYPE` parameters are defined in reference to where the local request originates. Refer to Managing Parameters for Buffer and Record Conversion for more information about these parameters.

**RDOM =** ***identifier***

specifies the name of the remote domain responsible for the actual execution of this service. If this parameter is not specified and a routing criteria (see the following definition for ROUTING keyword) is not specified, then the local domain assumes that any remote domain of the same type accepts this service and it selects a known domain (a domain to which a connection already exists) or remote domain from the DM_REMOTE_DOMAINS section.

**REM_TPSUT =** ***"string"***

identifies the TP service user title for the remote system. Some users of OSI TP implementations require this field. It is not required for OS 2200 OLTP-TM2200, OpenTI, A Series Open/OLTP, or BEA TMA OSI TP. If the TPSUT_TYPE value is PRINTABLESTRING, the maximum length is 60 characters, which must comply with Abstract Syntax Notation (ASN.1) type of PRINTABLESTRING. If the TPSUT_TYPE value is an INTEGER, the maximum length must fit into a LONG. The TPSUT_TYPE must be defined prior to defining the remote TPSUT.

**RNAME =** ***identifier***

specifies the remote service name expected by the remote domain. If this parameter is not specified, the value is the same as the name specified in the *service*.

**ROUTING =** ***identifier***

allows a local domain to perform data-dependent routing when more than one remote domain offers the same service. The *identifier* specifies the name of the routing criteria used for this data-dependent routing. If not specified, data-dependent routing is not done for this service. *identifier* must be 15 characters or less in length. If multiple entries exist for the same service name but with different RDOM parameters, the ROUTING parameter should be the same for all of these entries.

**TPSUT_TYPE = {INTEGER | PRINTABLESTRING}**

specifies the type of encoding to be performed on the REM_TPSUT parameter. The default type is PRINTABLESTRING. If TPSUT_TYPE is not specified, the default is used. The INTEGER and PRINTABLESTRING are ASN.1 types.

**TRANTIME =** ***integer***

specifies the default time-out value in seconds for a transaction automatically started for the associated service. The value must be greater than or equal to 0 and less than 2147483648. The default is 30 seconds. A value of 0 implies the maximum time-out value for the machine.

## See Also

Refer to Table A-5 "DM_REMOTE_SERVICES SECTION" for more information about setting these parameters through the `dmadmin` utility.

# DM_ROUTING Section

This section provides information for data-dependent routing of service requests using `FML`, `VIEW`, `X_C_TYPE`, and `X_COMMON` typed buffers.

## Format

DM_ROUTING entries have the following format.

*CRITERION_NAME* required parameters

where

*CRITERION_NAME* is the (*identifier*) name of the routing entry that was specified on the services entry. *CRITERION_NAME* must be 15 characters or less in length.

## Valid Parameters

Following is a list of valid parameters for the `DM_ROUTING` section:

**Table 3-8 Parameters for DM_ROUTING**

| Parameter | Required/Optional | Description |
|---|---|---|
| `BUFTYPE` | Required | Types and subtypes of data buffers for which this routing entry is valid |
| `FIELD` | Required | Name of routing field |
| `RANGES` | Required | Ranges and associated remote domain names (RDOM) for routing field |

## Parameter Definitions

Following is more detailed information about the `DM_ROUTING` section parameters:

**BUFTYPE = "*type1*[*:subtype1{,subtype2 . . .* ]][;*type2{:subtype3*[*, . . .* ]]] *. ..*"**

specifies types and subtypes of data buffers for which this routing entry is valid. The types are restricted to be either `FML, VIEW, X_C_TYPE,` or `X_COMMON`. No subtype can be specified for type `FML` and subtypes are required for the other types ("*" is not allowed). Duplicate type/subtype pairs cannot be specified for the same routing criterion name; more than one routing entry can have the same criterion name as long as the type/subtype pairs are unique. This parameter is required. If multiple buffer types are specified for a single routing entry, the data types of the routing field for each buffer type must be the same.

For `FML` buffers, if the field value is not set or does not match any specific range and a wildcard range has not been specified, an error is returned to the application process that requested the execution of the remote service.

**FIELD = *identifier***

specifies the name of the routing field. It must be 30 characters or less. This field is assumed to be a field name that is identified in an FML field table (for `FML` buffers) or an FML view table (for `VIEW, X_C_TYPE,` or `X_COMMON` buffers). The `FLDTBLDIR` and `FIELDTBLS` environment variables are used to locate FML field tables, and the `VIEWDIR` and `VIEWFILES` environment variables are used to locate FML view tables.

**RANGES = "*string*"**

specifies the ranges and associated remote domain names (RDOM) for the routing field. *string* must be enclosed in double quotes. The format of *string* is a comma-separated ordered list of range/RDOM pairs.

A range is either a single value (signed numeric value or character string in single quotes), or a range of the form "lower - upper" (where lower and upper are both signed numeric values or character strings in single quotes). Note that "lower" must be less than or equal to "upper." To embed a single quote in a character string value (as in O'Brien, for example), it must be preceded by two backslashes ('O\\'Brien').

The value `MIN` can be used to indicate the minimum value for the data type of the associated `FIELD`; for strings and arrays, it is the null string. For character fields, it is 0; for numeric values, it is the minimum numeric value that can be stored in the field.

The value `MAX` can be used to indicate the maximum value for the data type of the associated `FIELD`; for strings and arrays, it is effectively an unlimited string of octal-255 characters; for a character field, it is a single octal-255 character; for numeric values, it is the maximum numeric value that can be stored in the field.

Thus, "`MIN - -5`" is all numbers less than or equal to –5 and "`6 - MAX`" is all numbers greater than or equal to 6. The meta-character "*" (wildcard) in the position of a range

indicates any values not covered by the other ranges previously seen in the entry; only one wildcard range is allowed per entry and it should be last (ranges following it are ignored).

The routing field can be of any data type supported in FML. A numeric routing field must have numeric range values and a string routing field must have string range values.

String range values for string, array, and character field types must be placed inside a pair of single quotes and cannot be preceded by a sign. Short and long integer values are a string of digits, optionally preceded by a plus or minus sign. Floating point numbers are of the form accepted by the C compiler or `atof()`: an optional sign, then a string of digits optionally containing a decimal point, then an optional `e` or `E`, followed by an optional sign or space, followed by an integer.

When a field value matches a range, the associated RDOM value specifies the remote domain to which the request should be routed. An RDOM value of "*" indicates that the request can go to any remote domain known by the gateway group. Within a range/RDOM pair, the range is separated from the RDOM by a ":".

### See Also

Refer to Table A-6 "DM_ROUTING SECTION" for more information about setting these parameters through the `dmadmin` utility.

# Methods for Modifying Configurations

The BEA Tuxedo system provides three methods for modifying your TMA OSI TP configuration:

- BEA Administration Console

  A Web-based graphical user interface (GUI) you can use to dynamically configure your application. You can display and change configuration information, determine the state of each component in the system, and obtain statistical information about items such as executed requests and queued requests.

- Command-line utilities

  Most of the functionality needed for dynamic modification is provided by two commands: dmadmin and dmconfig. dmadmin is a shell-level command with over 70 subcommands for performing various administrative tasks, including dynamic system modification. tmconfig is a shell-level command that you can use to add and modify configuration entries while your system is running.

  Refer to the "Utilities Reference" section of this document for more information.

- MIB API

  A Management Information Base API that enables you to write your own programs to monitor your system and make dynamic changes to your system.

Refer to the "Dynamically Modifying an Application" section of your BEA Tuxedo 9.0 or 9.1 documentation for ATMI Administration for more detailed information about modifying your configuration.

CHAPTER 4

# Configuring BEA Tuxedo Mainframe Adapter for OSI TP

After the installation of BEA Tuxedo Mainframe Adapter for OSI TP is complete, you must configure the software. The proper configuration of TMA OSI TP sets up the gateway configuration.

This section covers the following topics:



## Configuration Prerequisites

The TMA OSI TP product software must be installed and accessible to your text editor. You must have file permission to access the `install` directory and modify the sample `DMCONFIG` file.

In addition, the following prerequisites must be met to successfully complete the configuration procedure:

- The `$TUXDIR/udataobj/DMTYPE` file defining the valid domain types must exist so the `dmloadcf` utility can load the binary configuration file and must contain a domain type of `OSITPX`. During the installation process if the `DMTYPE` file does not contain an OSI TP entry, the `DMTYPE` file is automatically updated with the required OSITP domain type.
- The effective user identifier of the person running `dmloadcf` must match the `UID` in the `RESOURCES` section of the `TUXCONFIG` file.

# Setting Environment Variables

Before you can invoke system commands, you must set several system environment variables. The following table provides descriptions of the four variables you must set. Most of the environment variables required by BEA Tuxedo Mainframe Adapter for OSI TP are set when you set up Tuxedo. Refer to your Tuxedo documentation for more information about setting the Tuxedo environment variables.

**Table 4-1 System Environment Variables**

| Variable | Optional/ Required | Default | Description |
|---|---|---|---|
| OSIRUNDIR | Required | None | Indicates location of TMA OSI TP runtime files. |
| GW_DFLT_TRANTIME | Optional | 300 seconds | Default transaction time for the server. Value is in seconds |
| GW_POLLING_INTERVAL | Optional | 300 seconds | Internal Polling time for requests to be serviced. Value is in seconds; valid range is 10-10,000 seconds. |
| GW_TIMER_INTERVAL | Optional | 300 (3 seconds) | Internal timer for global transaction requests. Value is in one hundredths (1/100) of a second. Valid values are 100-1000 indicating 1-10 seconds. |

You must set OSIRUNDIR, before you can boot the gateway or run the osiadmin utility. If you do not set the OSIRUNDIR environment variable before you boot the gateway, you will receive a message telling you to set OSIRUNDIR. This environment variable specifies the path that the TMA gateway uses for runtime files. You can set the OSIRUNDIR environment variable through a script, a command line entry, or through the Windows System Properties in the Control Panel. The variable value should include the path and directory as appropriate for your operating system. If the directory does not exist, the system will create it for you.

The default transaction time on the server is determined at startup by an optional environment variable called GW_DFLT_TRANTIME. If you do not set this variable, the default value is 5 minutes (300 seconds). This environment variable can be set to a different value at startup, but if the value exceeds the maximum allowed for a LONG, then the value is reset to 300 and a LIBGWO_CAT msg 2204 is sent to the ULOG to indicate that the maximum has been exceeded.

**Note:** The maximum for a LONG is 2147483647.

# Defining Gateway Configurations

Whether you are defining a new gateway configuration or modifying an existing one, both processes are similar. Defining a gateway configuration requires the following steps:

1. Define the TMA OSI TP servers in Tuxedo so that the BEA TMA system can recognize the gateway servers. Refer to Defining TMA OSI TP Servers for BEA Tuxedo for more information.
2. Run the osiadmin processor if you are upgrading from eLink OSI TP 1.3. Refer to Using the OSI TP Administration Utility for more information.
3. Determine which parameter information needs to be added or changed to define the gateway. Refer to Understanding the DMCONFIG File for more information.
4. Edit the OSI TP Tailor file if OSI TP-specific tables need to be tuned. Refer to Tuning OSI TP-Specific Tables with the TAILOR File for more information.
5. Create or edit the DMCONFIG file with new gateway information. Refer to Implementing Native-A Encoding and Understanding the DMCONFIG File for more information.
6. Generate a binary version of the dmconfig file by running the dmloadcf utility. Refer to Processing a Configuration File with the dmloadcf Utility for more information.

After you perform these steps, you are ready to start the gateway using the Tuxedo tmboot command. Refer to the *BEA Tuxedo Online Documentation* for more information about Tuxedo commands.

# Defining TMA OSI TP Servers for BEA Tuxedo

To establish a gateway configuration, the BEA Tuxedo system must recognize the gateway servers, DMADM, GWADM, and GWOSITP. You define the TMA OSI TP administrative and gateway servers to the BEA Tuxedo system by editing the UBBCONFIG file.

Perform the following steps to define TMA OSI TP servers for BEA Tuxedo:

1. In the GROUPS section of the UBBCONFIG file, add a server group using the following format:

   ```
   OSIGRP  GRPNO=1  LMID=SITE1
   ```

   **Note:** OSIGRP is used as an example. You may give the group any name you wish.

2. In the SERVERS section of the UBBCONFIG file, add the three server names: DMADM, GWADM, and GWOSITP.

   **Note:** The DMADM and GWADM entries should be placed in this order **BEFORE** GWOSITP in the UBBCONFIG file so the admin servers are loaded before the GWOSITP gateway server.

   It is recommended that you set the RESTART parameter in the SERVERS section to Y so that the gateway will automatically restart in case of failure.

# Sample UBBCONFIG File

The following file is a sample UBBCONFIG file that defines gateway servers to the BEA Tuxedo system.

**Listing 4-1 Sample UBBCONFIG File**

```
#----------------------------------
# TMA OSI TP Test; Client ubbconfig
#----------------------------------

*RESOURCES
#---------------
# Replace IPCKEY
#---------------
IPCKEY       52029
MASTER       SITE1
DOMAINID     FRONTEND
```

```
PERM         0660
MAXACCESSERS 40
MAXSERVERS   80
MAXSERVICES  80
MAXCONV      120
MODEL        SHM
LDBAL        Y
MAXGTT       120
MAXBUFTYPE   16
MAXBUFSTYPE  32
SCANUNIT     5
SANITYSCAN   10
DBBLWAIT     5
BBLQUERY     50
BLOCKTIME    15

*MACHINES
#--------------------
# Replace machine name
#--------------------
DALNT45      LMID=SITE1
#-----------------------------
# Replace directories as needed
#-----------------------------
  TUXDIR="c:\tuxedo"
  APPDIR="D:\dwh\base\ositp\test\client"
  TUXCONFIG="D:\dwh\base\ositp\test\client\tuxconfig"
  TLOGDEVICE="D:\dwh\base\ositp\test\client\TLOG"
  TLOGNAME="TLOG"

*GROUPS
OSIGRP2      GRPNO=2 LMID=SITE1
OSIGRP3      GRPNO=3 LMID=SITE1 TMSNAME="TMS" TMSCOUNT=2

*SERVERS
DEFAULT: RESTART=Y
DMADM      SRVID=101 SRVGRP=OSIGRP2 CLOPT="-A" REPLYQ=N
GWADM      SRVID=103 SRVGRP=OSIGRP2 CLOPT="-A"" REPLYQ=N
```

```
GWOSITP    SRVID=104 SRVGRP=OSIGRP2 CLOPT="-A" GRACE=0" REPLYQ=N
CRPCSERV   SRVID=8 SRVGRP=OSIGRP3 CLOPT="-A" RQADDR="rpcq"
CCONVSRV   SRVID=9 SRVGRP=OSIGRP3 CLOPT="-A" RQADDR="convq "CONV=Y
               MIN=3 MAX=5

*SERVICES
DEFAULT:      LOAD=50 AUTOTRAN=N
CTOUPPER      PRIO=50
CCONVRTN      PRIO=50
CCONVRTN2     PRIO=50
CCONVRTN3     PRIO=50
CTOUPPER2      PRIO=50
```

Refer to the *BEA Tuxedo Reference Manual* for additional information about the UBBCONFIG file.

# Example of a Multiple Gateway Configuration

Following is a sample UBBCONFIG file and corresponding DMCONFIG file for multiple gateways that reside on the same physical system. Note that use of LDBAL=Y in the UBBCONFIG file is not required for multiple gateways. Loads are automatically balanced for multiple gateways.

**Listing 4-2 UBBCONFIG for Multiple Gateways**

```
*RESOURCES
IPCKEY        65952
MASTER        "SITE1"
MODEL         SHM
PERM          0660
LDBAL         N   # not needed for gateway load balancing
MAXACCESSERS  40
MAXSERVERS    80
MAXSERVICES   80
MAXGTT        120
SCANUNIT      5
SANITYSCAN    10
BLOCKTIME     15
```

```
MAXCONV         120

*MACHINES
"SITE1"         LMID="SITE1"

                TUXCONFIG="D:\tuxedo\configs\TUXCONFIG"
                TUXDIR="D:\tuxedo"
                APPDIR="D:\tuxedo\apps"
                TLOGDEVICE="D:\tuxedo\log\TLOG"
                ULOGPFX="D:\tuxedo\log\ULOG"
                TLOGNAME=TLOG
                TLOGSIZE=20
                TYPE="SITE1"
*GROUPS

GROUP1          LMID="SITE1"
                GRPNO=1
GROUP2          LMID="SITE1"
                GRPNO=2

DMGRP           LMID="SITE1"
                GRPNO=3

*SERVERS

DEFAULT:        RQPERM=0666
                RPPERM=0666
                MIN=1
                MAX=1
                CONV=N
                MAXGEN=1
                GRACE=86400
                RESTART=Y
                SYSTEM_ACCESS=FASTPATH


DMADM           SRVGRP=DMGRP
                SRVID=20
```

```
                CLOPT="-A"
                RESTART=Y
                MAXGEN=2
                REPLYQ=N

GWADM           SRVGRP=GROUP1
                SRVID=21
                CLOPT="-A"
                RESTART=Y
                MAXGEN=2
                REPLYQ=N

GWOSITP         SRVGRP=GROUP1
                SRVID=22
                CLOPT="-A"
                RESTART=Y
                MAXGEN=2
                REPLYQ=N

GWADM           SRVGRP=GROUP2
                SRVID=23
                CLOPT="-A"
                RESTART=Y
                MAXGEN=2
                REPLYQ=N

GWOSITP         SRVGRP=GROUP2
                SRVID=24
                CLOPT="-A"
                RESTART=Y
                MAXGEN=2
                REPLYQ=N

*SERVICES

DEFAULT:  LOAD=50 AUTOTRAN=N
```

**Listing 4-3 DMCONFIG File for Multiple Gateways**

```
*DM_RESOURCES
VERSION="SITE1"
#
*DM_LOCAL_DOMAINS

"GW-1"
GWGRP       = GROUP1
TYPE        = OSITPX
DOMAINID    = "GW-1"
BLOCKTIME   = 30
DMTLOGDEV   = "D:\tuxedo\log\DMLOG"

"GW-2"
GWGRP       = GROUP2
TYPE        = OSITPX
DOMAINID    = "GW-2"
BLOCKTIME   = 30
DMTLOGDEV   = "D:\tuxedo\log\DMLOG2"

################################################################
*DM_REMOTE_DOMAINS

DEFAULT:

"OPENTI" TYPE="OSITPX" DOMAINID="OPENTI"

################################################################
*DM_OSITPX

"GW-1"
      AET="{1.3.145.62.103},{2}"
      TAILOR_PATH="d:\tuxedo\configs\tailor.txt"
                           # Inserted from OSITP's config file:
      NWADDR="//SITE1:102"

"GW-2"
```

```
      AET="{1.3.145.62.103},{3}"
      TAILOR_PATH="d:\tuxedo\configs\tailor2.txt"
                              # Inserted from OSITP's config file:
      NWADDR="//SITE1:2000" #second gateway must use another
                                    #IP or different port number

"OPENTI"
      AET="{1.3.122.61.203},{20}"
      NWADDR="122.61.203.20"


*DM_LOCAL_SERVICES


*DM_REMOTE_SERVICES
DEFAULT: TRANTIME=300
    # Tuxedo will alternate outgoing calls between the two LDOMs.

callSvc2   RDOM="OPENTI" LDOM="GW-1" PRIO=66
callSvc2   RDOM="OPENTI" LDOM="GW-2" PRIO=66
```

The following example uses the same UBBCONFIG file as in Listing 4-2 and shows how to configure one LDOM to support the non-multiplexed path and the other to support the multiplexed path.

**Listing 4-4 DMCONFIG File for Multiplexed and non-Multiplexed Connections**

```
*DM_RESOURCES
VERSION="SITE1"
#
*DM_LOCAL_DOMAINS

"GW-1"
GWGRP       = GROUP1
TYPE        = OSITPX
DOMAINID    = "GW-1"
```

```
BLOCKTIME   = 30
DMTLOGDEV   = "D:\tuxedo\log\DMLOG"

"GW-2"
GWGRP       = GROUP2
TYPE        = OSITPX
DOMAINID    = "GW-2"
BLOCKTIME   = 30
DMTLOGDEV   = "D:\tuxedo\log\DMLOG2"

#################################################################
*DM_REMOTE_DOMAINS

DEFAULT:

"OPENTI" TYPE="OSITPX" DOMAINID="OPENTI"
"dal2200" TYPE="OSITPX" DOMAINID="dal2200"

#################################################################
*DM_OSITPX

"GW-1"
      AET="{1.3.145.62.103},{2}"
      TAILOR_PATH="d:\tuxedo\configs\tailor.txt"
                            # Inserted from OSITP's config file:
      NWADDR="//SITE1:102"

"GW-2"
      AET="{1.3.145.62.103},{3}"
      TAILOR_PATH="d:\tuxedo\configs\tailor2.txt"
                            # Inserted from OSITP's config file:
      NWADDR="//SITE1:2000" #second gateway must use another
                                    #IP or different port number
      EXTENSIONS="MULTIPLEX_POLICY=DEMAND"

"OPENTI"
      AET="{1.3.122.61.203},{20}"
      NWADDR="122.61.203.20:2003"
```

```
      EXTENSIONS="MULTIPLEX=Y"

"dal2200"
      AET="{1.3.132.61.46},{3}"
      XATMI_ENCODING="OLTP_TM2200"
      NWADDR="132.61.46.3;132.61.147.1" #redundant IP addresses
      T_SEL="OSITP"



*DM_LOCAL_SERVICES


*DM_REMOTE_SERVICES
DEFAULT: TRANTIME=300
    # Tuxedo will alternate outgoing calls between the two LDOMs.

CallSvc1   RDOM="dal2200" LDOM="GW-1" PRIO=66
callSvc2   RDOM="OPENTI" LDOM="GW-2" PRIO=66
```

# Using the Tuxedo MP Model with the TMA OSI TP Gateway

It is useful to use the Tuxedo MP model (for multiprocessors that do not have global shared memory or for networked applications) when you require two TMA OSI TP systems to exist in the same domain. (Refer to the BEA Tuxedo documentation for more information about the `MODEL` parameter.) A practical example of this is setting up a Windows NT cluster. The TMA OSI TP gateway supports active-active failover on an NT cluster. In the MP model case, there are two unique nodes, one defined as the master and a second one defined as a slave or backup system in the case of clustering. There is one `UBBCONFIG` and one `DMCONFIG` that physically exist on the master node. At `TMBOOT` time, a copy of the `TUXCONFIG` is propagated to the slave or backup system.

**Listing 4-5 UBBCONFIG File for MP Model**

```
UBBCONFIG
```

```
*RESOURCES
IPCKEY        65952
MASTER        SITE1,SITE2
MODEL         MP
OPTIONS       LAN
PERM          0660
LDBAL         N   # not needed for gateway load balancing
MAXACCESSERS  40
MAXSERVERS    80
MAXSERVICES   80
MAXGTT        120
SCANUNIT      5
SANITYSCAN    10
BLOCKTIME     15
MAXCONV       120


*MACHINES
"SITE1"       LMID="SITE1"
              TUXCONFIG="D:\tuxedo\configs\TUXCONFIG"
              TUXDIR="D:\tuxedo"
              APPDIR="D:\tuxedo\apps"
              TLOGDEVICE="D:\tuxedo\log\TLOG"
              ULOGPFX="D:\tuxedo\log\ULOG"
              TLOGNAME=TLOG
              TLOGSIZE=20
              TYPE="INTEL"

"SITE2"       LMID="SITE2"
              TUXCONFIG="D:\tuxedo\configs\TUXCONFIG"
              TUXDIR="D:\tuxedo"
              APPDIR="D:\tuxedo\apps"
              TLOGDEVICE="D:\tuxedo\log\TLOG"
              ULOGPFX="D:\tuxedo\log\ULOG"
              TLOGNAME=TLOG
              TLOGSIZE=20
              TYPE="INTEL"
```

```
*GROUPS
GROUP1          LMID="SITE1"
                GRPNO=1
GROUP2          LMID="SITE2"
                GRPNO=2
DMGRP           LMID="SITE1"
                GRPNO=3


*NETWORK
SITE1           NADDR="/SITE1:5020"
                NLSADDR="//SITE1:5021"
SITE2           NADDR="//SITE2:5020"
                NLSADDR="//SITE2:5021"


*SERVERS
DEFAULT:        RQPERM=0666
                REPLYQ=Y
                RPPERM=0666
                MIN=1
                MAX=1
                CONV=N
                MAXGEN=1
                GRACE=86400
                RESTART=N
                SYSTEM_ACCESS=FASTPATH


DMADM           SRVGRP=DMGRP
                SRVID=20
                CLOPT="-A"
                RESTART=Y
                MAXGEN=2
                REPLYQ=N

GWADM           SRVGRP=GROUP1
```

```
                SRVID=22
                CLOPT="-A"
                RESTART=Y
                MAXGEN=2
                REPLYQ=N

GWOSITP         SRVGRP=GROUP1
                SRVID=23
                CLOPT="-A"
                RESTART=Y
                MAXGEN=2
                REPLYQ=N

GWADM           SRVGRP=GROUP2
                SRVID=24
                CLOPT="-A"
                RESTART=Y
                MAXGEN=2
                REPLYQ=N

GWOSITP         SRVGRP=GROUP2
                SRVID=25
                CLOPT="-A"
                RESTART=Y
                MAXGEN=2
                REPLYQ=N

*SERVICES
DEFAULT:  LOAD=50 AUTOTRAN=N
```

**Listing 4-6 DMCONFIG File for MP Model**

```
DMCONFIG

*DM_RESOURCES
VERSION="SITE1"
```

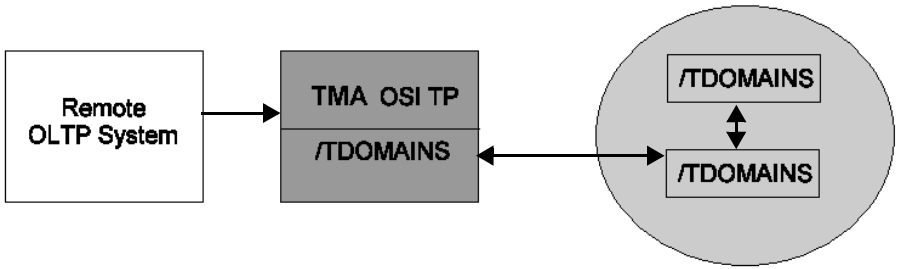
```
#
*DM_LOCAL_DOMAINS

"GW-1"
GWGRP       = GROUP1
TYPE        = OSITPX
DOMAINID    = "GW-1"
BLOCKTIME   = 30
DMTLOGDEV   = "D:\tuxedo\log\DMLOG1"
DMTLOGNAME  = "DMLOG"

"GW-2"
GWGRP       = GROUP2
TYPE        = OSITPX
DOMAINID    = "GW-2"
BLOCKTIME   = 30
DMTLOGDEV   = "D:\tuxedo\log\DMLOG2"
DMTLOGNAME  = "DMLOG"

################################################################
*DM_REMOTE_DOMAINS

DEFAULT:

"OPENTI" TYPE="OSITPX" DOMAINID="OPENTI"

################################################################
*DM_OSITPX

"GW-1"
               AET="{1.3.145.62.103},{2}"
               TAILOR_PATH="d:\tuxedo\configs\tailor1.txt"
               # Inserted from OSITPX's config file:
               NWADDR="//SITE1:102"

"GW-2"
               AET="{1.3.145.62.103},{3}"
               TAILOR_PATH="d:\tuxedo\configs\tailor2.txt"
```

```
                # Inserted from OSITPX's config file:
                NWADDR="//SITE2:102" # second gateway must use
                                     # another IP or different
                                     # port number

"OPENTI"
                AET="{1.3.122.61.203},{20}"
                NWADDR="122.61.203.20"


########################################################################*D
M_LOCAL_SERVICES


################################################################
*DM_REMOTE_SERVICES
DEFAULT: TRANTIME=300
# Each system will service different applications.

callSvc2   RDOM="OPENTI" LDOM="GW-1" PRIO=66
callSvc3   RDOM="OPENTI" LDOM="GW-2" PRIO=66
```

# Using TMA OSI TP as a Pass-Through to Other Tuxedo Systems

For this example, the TMA OSI TP gateway acts as a pass-through to allow access to services on other Tuxedo systems. The TMA OSI TP gateway receives service requests from a remote OLTP system and then forwards them through the `/TDOMAINS` gateway to a remote Tuxedo system. The systems are as follows:

**Figure 4-1 Example of TMA OSI TP Acting as a Pass-Through to Other Tuxedo Systems**




Listing 4-7 shows a sample UBBCONFIG file and Listing 4-8 shows the corresponding DMCONFIG file for a pass-through configuration.

**Listing 4-7 Sample UBBCONFIG File for Pass-Through Configuration**

```
*RESOURCES
IPCKEY          65952
MASTER          "SITE1"
MODEL           SHM
PERM            0777

*MACHINES
"SITE1" LMID="SITE1"

                TUXCONFIG="D:\tuxedo\appdir\TUXCONFIG"
                TUXDIR="D:\tuxedo"
                APPDIR="D:\tuxedo\appdir"
                TLOGDEVICE="D:\tuxedo\log\TLOG"
                ULOGPFX="D:\tuxedo\log\ULOG"
                TLOGNAME=TLOG
                TLOGSIZE=20
                TYPE="SITE1"
*GROUPS

ADMGRP          LMID="SITE1"
                GRPNO=1
```

```
OSIGRP          LMID="SITE1"
                GRPNO=2

TDOMGRP         LMID="SITE1"
                GRPNO=3
                OPENINFO=NONE

*SERVERS

DEFAULT:        RQPERM=0666
                RPPERM=0666
                MIN=1
                MAX=1
                CONV=N
                MAXGEN=1
                GRACE=86400
                RESTART=N
                SYSTEM_ACCESS=FASTPATH

DMADM           SRVGRP=ADMGRP
                SRVID=20
                CLOPT="-A"
                RESTART=N
                REPLYQ=N

GWADM           SRVGRP=OSIGRP
                SRVID=21
                CLOPT="-A"
                RESTART=N
                REPLYQ=N

GWOSITP         SRVGRP=OSIGRP
                SRVID=22
                CLOPT="-A"
                RESTART=Y
                REPLYQ=N

GWADM           SRVGRP=TDOMGRP
```

```
              SRVID=51
              CONV=N
              CLOPT="-A"
              REPLYQ=N
              RESTART=N

GWTDOMAIN     SRVGRP=TDOMGRP
              SRVID=52
              CLOPT="-A"
              REPLYQ=N
              RESTART=Y
*SERVICES

DEFAULT:  LOAD=50 AUTOTRAN=N
```

**Listing 4-8 Sample DMCONFIG File for Pass-Through Configuration**

```
*DM_RESOURCES
VERSION="SITE1"
#
*DM_LOCAL_DOMAINS

 #  SECURITY=NONE

"osi-local"
             GWGRP       = OSIGRP
             TYPE        = OSITPX
             DOMAINID    = "local"
             BLOCKTIME   = 2000
             AUDITLOG    = "D:\tuxedo\log\AUDIT"
             DMTLOGDEV   = "D:\tuxedo\log\DMLOG"
             DMTLOGSIZE  = 2048
             DMTLOGNAME  = "DMLOG"

"td-local"   GWGRP=TDOMGRP
             TYPE=TDOMAIN
```

```
            DOMAINID="td-local"
            DMTLOGDEV="D:\tuxedo\log\TDMLOG"


################################################################
*DM_REMOTE_DOMAINS

DEFAULT:

"osi-client" TYPE=OSITPX   DOMAINID="osi-client"
"td-backend" TYPE=TDOMAIN DOMAINID="td-tpaix1"

################################################################
*DM_TDOMAIN

"td-local"        NWADDR="192.63.22.2:5000"
"td-backend"      NWADDR="192.63.24.74:5000"
################################################################
*DM_OSITPX

"osi-local"
             AET="{1.3.192.63.22},{2}"
             TAILOR_PATH="d:\tuxedo\configs\tailor.txt"

# the NWADDR for OSI TP may have the same IP as /TDOMAINS, but
# requires a different port number
             NWADDR="192.63.22.2:102"

"osi-client"
              AET="{1.3.192.23.2},{3}"
             NWADDR="192.23.2.3"
              T_SEL="OSITP"

################################################################
*DM_LOCAL_SERVICES
# define the incoming services here, even though they reside on
# some remote /TDOMAIN machine.  Include views also on this machine
# for TMA OSI TP to process incoming messages
```

```
callSvc1

#################################################################
*DM_REMOTE_SERVICES
DEFAULT:

# define the actual remote service request here.  It will be
# routed by /TDOMAINS

callSvc1   RDOM="td-backend"   LDOM="td-local" RNAME="callSvc1"
```

Note that the service that resides on the backend /TDOMAIN system must be defined as a local service on the TMA OSI TP system, so TMA OSI TP can process the incoming request. It must also be defined as a remote service so that the /TDOMAIN gateway can pass the service request to the backend /TDOMAIN system. The gateway system must have available the viewfiles and corresponding environment variables that are required by the service, even though the service exists on the backend system.

# Setting up Security

BEA TMA OSI TP supports the following two types of security:

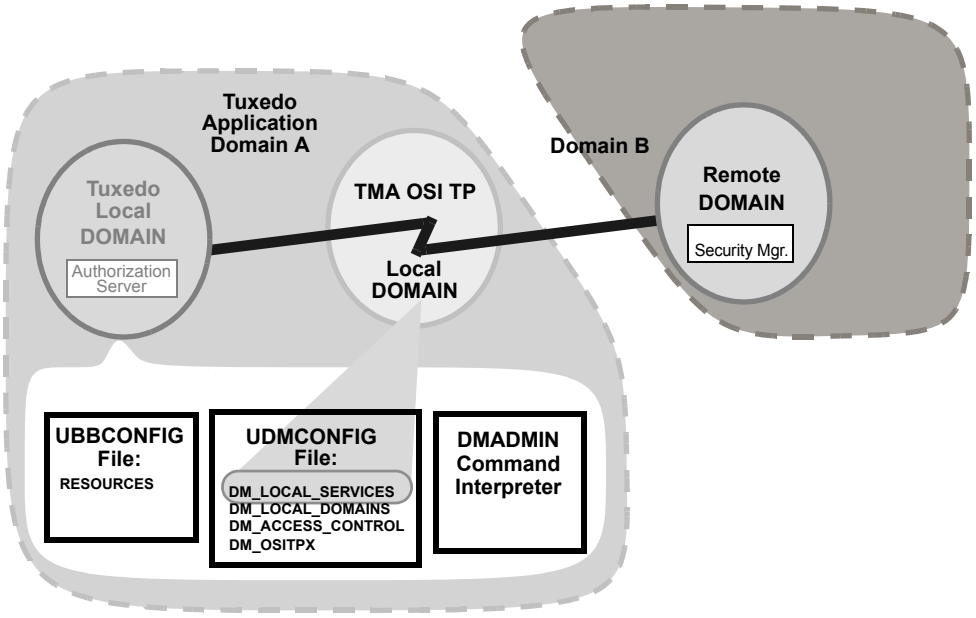
- Link Layer Security
- Tuxedo Authentication

The BEA Tuxedo UBBCONFIG and TMA OSI TP DMCONFIG files include five sections in which you specify security parameters for whichever type of security you want to implement:

- RESOURCES section of the UBBCONFIG file
- DM_LOCAL_DOMAINS section of the DMCONFIG file
- DM_OSITPX section of the DMCONFIG file
- DM_ACCESS_CONTROL section of the DMCONFIG file
- DM_LOCAL_SERVICES section of the DMCONFIG file

The following figure shows the relationships between security elements for TMA OSI TP.

**Figure 4-2 TMA OSI TP Security Elements**




# Enabling Link Layer Security

To enable security, both the local and remote domains must support security. For Link Layer Security (LLS), the administrator must define the `SECURITY` in the `*LOCAL_DOMAINS` section as `DM_PW` and the `OPTIONS` parameter in the `DM_OSITPX` section must be set to `SECURITY_SUPPORTED`. The Local and Remote domain passwords should be set by the administrator through the `dmadmin` command `passwd`.

Whenever Link Layer Security is defined for the local domain and the remote domain, the `passwd` entered through `dmadmin` is hashed with a private key. The hashed value and private key are sent to the remote domain where it is compared with the hashed `passwd` on the remote system. This is a challenge response mechanism used to secure connections with a remote domain.

Three sections in the `DMCONFIG` file contain parameters affecting how TMA OSI TP controls access to the local Tuxedo domain:

- `DM_LOCAL_DOMAINS` section contains a `SECURITY` parameter that specifies the type of security enforced for the Tuxedo local domain.

The SECURITY parameter settings in this section work in conjunction with the SECURITY parameter in the RESOURCES section of the Tuxedo domain's UBBCONFIG file to establish how TMA OSI TP controls access to the Tuxedo local domain. If this parameter is set to NONE or APP_PW, the TMA OSI TP domain takes no action with regard to security. If this parameter is set to DM_PW, the TMA OSI TP domain enforces security according to the security settings in the DM_PASSWORDS section of the BDMCONFIG file.

**Caution:** Do not delete the Tuxedo BDMCONFIG file. The DM_PW information will be lost if the file is deleted. When new passwords are entered, the GWOSITP service must be shut down and restarted for the passwords to take effect.

- DM_REMOTE_DOMAINS section contains an OPTIONS=SECURITY_SUPPORTED parameter that specifies the type of security enforced for the Tuxedo remote domain.
- DM_OSITPX section contains an OPTION of SECURITY_SUPPORTED, which indicates that the remote domain supports the OSI TP security extension. The OSI TP security extension allows OSI TP systems to perform link-layer security. The link layer security feature is activated when the DM_LOCAL_DOMAINS section has SECURITY=DM_PW set and OPTIONS=SECURITY_SUPPORTED is set for the remote domain.

Refer to "Enabling Tuxedo Authentication" for more information about Tuxedo security.

The following sample is a DMCONFIG file that defines the necessary parameters for establishing Link Layer Security.

**Listing 4-9 Sample DMCONFIG File for Establishing Link Layer Security**

```
#-----------------------------------
# TMA OSI TP test; Client dmconfig
#-----------------------------------
#
*DM_LOCAL_DOMAINS

dalnt8
GWGRP=G3
TYPE=OSITPX
DOMAINID="dalnt8"
BLOCKTIME=30
MAXDATALEN=56
DMTLOGDEV="D:\tuxedo\log\DMLOG"
```

```
SECURITY=DM_PW       # turns link layer security on
DMTLOGNAME="DMLOG"

################################################################
*DM_REMOTE_DOMAINS
dal2200       TYPE=OSITPX   DOMAINID=dal2200 ACL_POLICY=GLOBAL
openti        TYPE=OSITPX   DOMAINID=openti ACL_POLICY=GLOBAL
icl2          TYPE=OSITPX   DOMAINID=icl2
aseries       TYPE=OSITPX   DOMAINID="aseries1" ACL_POLICY=LOCAL
              LOCAL_PRINCIPAL_NAME="MYNAME'

*DM_OSITPX

dalnt8
              AET="{1.3.132.61.146},{3}"
              TAILOR_PATH="D:\tuxedo\configs\tailor.txt"
              NWADDR="//dalnt8:102"
              DNS_RESOLUTION=STARTUP  # This is the default

# Remote domain dal2200 supports security
dal2200
              AET="{1.3.132.61.46},{3}"
              XATMI_ENCODING="OLTP_TM2200"
              NWADDR="132.61.146.3"
              T_SEL="OSITP"
              OPTIONS=SECURITY_SUPPORTED

# Remote domain openti supports security
openti
              AET="{1.3.122.62.103},{209}"
              NWADDR="122.62.103.209:2001"
              OPTIONS=SECURITY_SUPPORTED

# Remote domain icl12 does not support security
icl2
              AET="{1.3.142.60.203},{4}"
              NWADDR="142.60.203.4"
              T_SEL="ICLTP"
```

```
                S_SEL="SSEL"
                P_SEL="PSEL"

# Remote domain aseries1 does not support security
# DOMAINID "aseries1" shall be used as LOCALPRINCIPAL NAME
aseries1
                AET="{1.3.123.55.222},{51}"
                NWADDR="123.55.222.51"
                XATMI_ENCODING="PRELIMINARY"
                T_SEL="0x5453"
                S_SEL="0x3F5C3F"

*DM_LOCAL_SERVICES
TOUPPERF
                INRECTYPE="VIEW:V10"
                OUTBUFTYPE="FML:"
                COUPLING=LOOSE

TOUPPERF32
                INRECTYPE="VIEW:V10"
                OUTBUFTYPE="FML32:"
                COUPLING=TIGHT
TOUPPERV
                INBUFTYPE="X_C_TYPE:V10"
                INRECTYPE="VIEW:upper"
                COUPLING=LOOSE

TOUPPERC
                INCRECTYPE="X_OCTET"
                COUPLING-TIGHT

TOUPPERS
                OUTRECTYPE="X_OCTET"
                OUTBUFTYPE="STRING"
                INRECTYPE="X_OCTET"

TOUPPERX
                OUTRECTYPE="STRING"
```

```
              OUTBUFTYPE="STRING"
              INRECTYPE="X_OCTET"

*DM_REMOTE_SERVICES
DEFAULT: TRANTIME=300

ECHOXOCT RNAME="ECHOSRVR"
              OUTBUFTYPE="X_COMMON:ECHOVIEW"
              RDOM=dal2200
              LDOM=dalnt8

ECHOXCOM RNAME="ECHOSRVR"
              RDOM=openti
              LDOM=dalnt8
              AUTOPREPARE=Y

ECHOTYPE
              RNAME="ECHOSRVR"
              INBUFTYPE="X_C_TYPE:ECHOVIEW"
              INRECTYPE="X_C_TYPE:ECHOVIEW"
              OUTBUFTYPE="X_C_TYPE:ECHOVIEW"
              OUTRECTYPE="X_C_TYPE:ECHOVIEW"

ECHOVIEW
              RNAME="ECHOSRVR"
              INBUFTYPE="VIEW:ECHOVIEW"
              INRECTYPE="X_COMMON:ECHOVIEW"
              OUTBUFTYPE="VIEW:ECHOVIEW"
              OUTRECTYPE="X_COMMON:ECHOVIEW"
              RDOM=icl2
              LDOM=dalnt8
              TPSUT_TYPE= "PRINTABLESTRING"
               REM_TPSUT="tpmvs"
```

# Enabling Tuxedo Authentication

BEA TMA OSI TP gateway supports Tuxedo authentication and authorization at both the client side (for Tuxedo clients) and server side. Authentication and authorization at the client side works the same as /T domains. Authentication and authorization on the server side requires that the Link Layer Security, described previously, is configured for both the LOCAL DOMAIN and the REMOTE DOMAIN involved in the service call.

In order for user authentication at the server side domain to be performed, the SECURITY parameter defined in the servers local domain *RESOURCE section must be defined as either "APP_PW", "USER_AUTH", "ACL", or "MANDATORY_ACL".

When SECURITY is set to either "APP_PW" or "USER_AUTH", user authentication is performed. If the ACL_POLICY for the remote domain from which the call was issued is defined as LOCAL, then the user ID used for authentication will be the LOCAL_PRINCIPAL_NAME of the remote domain if it has been defined. If the LOCAL_PRINCIPAL_NAME has not been defined, then the user ID will be the DOMAINID of the remote domain. If the ACL_POLICY for the remote domain from which the call was issued is defined as GLOBAL, then the userid passed with the call is used for user authentication. The user ID is authenticated against the user IDs defined in the tpusr file.

When SECURITY is set to "ACL", then user authentication is performed as previously defined for "APP_PW" and "USER_AUTH". User authorization of the service, requested in the call, is also performed. ACL authorization requires that the services have been defined in the ACL Service list and that the user be a member of a group that is allowed access to this service. Services are added to the ACL Service list through the tpacladd command. If the requested service has not been added to the ACL Service list, then all users are allowed access to this service.

When SECURITY is set to "MANDATORY_ACL" authentication and authorization are performed identical to that for SECURITY equal "ACL", however, the request service must be defined in the ACL Service list and the users group must be allowed access to this service.

For more information regarding Tuxedo commands tpgrpadd, tpusradd, tpacladd, tpgrpmod, tpusrmod, tpaclmod, tpgrpdel, tpasrdel, and tpacldel, refer to the online BEA Tuxedo documentation at http://edocs.bea.com/tuxedo/tux80/index.htm.

The following sample shows a UBBCONFIG file. The example defines the necessary parameters for establishing Link Layer Security.

**Listing 4-10 Sample UBBCONFIG File Showing Security Set to MANDATORY_ACL.**

```
#-------------------------------------------------
# TMA OSI TP test ubbconfig for servers
#-------------------------------------------------
*RESOURCES
#-------------
Chnage IPCKEY
#-------------
IPCKEY        52029
MASTER        SITE1
DOMAINID      FRONTEND
PERM          0660
MAXACCESSSERS 40
MAXSERVERS    80
MAXSERVICES   80
MAXCONV       120
MODEL         SHM
LDBAL         Y
MAXGTT        120
MAXBUFTYPE    16
MAXBUFSTYPE   32
SCANUNIT      5
SANITYSCAN    10
DBBLWAIT      5
BBLQUERY      50
BLOCKTIME     15
SECURITY      MANDATORY_ACL
#
*MACHINES
#-------------
#Replace machine name
#-------------
DALNT45       LMID=SITE1
#-------------
# Replace directories as needed
#-------------
```

```
              TUXDIR="c:\tuxedo"
              APPDIR="c:\ositp/test
              TUXCONFIG="c:\tuxedo/udataobj\tuxconfig"
              TLOGDEVICE="c:\ositp\test\TLOG
              TLOGNAME=TLOG

*GROUPS
OSIGRP2       GRPNO=2  LMID-SITE1  TMSNAME="TMS"  TMSCOUNT=2
OSIGRP2       GRPNO=3  LMID=SITE1

*SERVERS
DEFAULT:RESTART=N  REPLYQ=Y
DMADM    SRVID=101  SRVGRP=OSIGRP3  CLOPT="-A" REPLYQ=N
GWADM    SRVID=102  SRVGRP=OSIGRP3  CLOPT="-A" REPLYQ=N
GWOSITP  SRVID=103  SRVGRP=OSIGRP3  CLOPT="-A" GRACE=0 REPLYQ=N
CRPCSERV SRVID=8    SRVGRP=OSIGRP3  CLOPT="-A" RQADDR="rpcq"
CCONSRV  SRVID=9    SRVGRP=OSIGRP3  CLOPT="-A" RQADDR="convq"
                    CONV=Y

*SERVICES
DEFAULT       LOAD=50  AUTOTRAN=n
CTOUPPER      PRIO=50
CCONVRTN      PRIO=50
CCONVRTN2     PRIO=50
CCONVRTN3     PRIO=50
CTOUPPER2     PRIO=50
```

The following sample shows a DMCONFIG file. The example defines the necessary parameters for user authentication and authorization.

**Listing 4-11 Sample DMCONFIG File Showing LLS set and ACL_POLICY of LOCAL**

```
#---------------------------------
TMA OSI TP Test; Client dmconfig
#---------------------------------
```

```
#
*DM_LOCAL DOMAINS

dalnt8
GWGRP=G3
TYPE=OSITPX
DOMAINID="dalnt8"
BLOCKTIME=30
MAXDATALEN=56
DMTLOGDEV="D:\tuxedo\log\DMLOG"
DMTLOGNAME="DMLOG"
SECURITY=DM_PW       # turns link layer security on

############################################################
*DM_REMOTE_DOMAINS
DAL2200    TYPE=OSITPX  DOMAINID=DAL2200  ACL_POLICY=LOCAL

openti     TYPE=OSITPX  DOMAINID=openti   ACL_POLICY=GLOBAL

icl2       TYPE=OSITPX DOMAINID=icl2
aseries    TYPE=OSITPX  DOMAINID="aseries1"

*DM_OSITPX

dalnt8
           AET="{1.3.132.61.146},{3}"
           TAILOR_PATH="d:\tuxedo\configs\tailor.txt"
           NWADDR="//dalnt8:102"
           DNS_RESOLUTION=STARTUP  # This is the default

# Remote domain dal2200 supports security
dal2200
           AET="{1.3.132.61.47},{3}"
           XATMI_ENCODING="OLTP_TM2200"
           NWADDR="132.61.47.3"
           T_SEL="OSITP"
           OPTIONS=SECURITY_SUPPORTED

# Remote domain openti supports security
openti
```

```
            AET="{1.3.122.62.103},{209}"
            NWADDR="122.62.103.209:2001"
            OPTIONS=SECURITY_SUPPORTED

# Remote domain icl2 does not support security
icl2
            AET="{1.3.142.60.203},{4}"
            NWADDR="142.60.203.4"
            T_SEL="ICLTP"
            S_SEL="SSEL"
            P_SEL="PSEL"

# Remote domain aseries1 does not support security
aseries1
            AET="{1.3.123.55.222},{51}"
            NWADDR="123.55.222.51"
            XATMI_ENCODING="PRELIMINARY"
            T_SEL="0X5453"
            S_SEL="0X3F5C3F"

*DM_LOCAL_SERVICES
TOUPPERF
            INRECTYPE="VIEW:view10"
            OUTBUFTYPE="FML:"
            COUPLING=LOOSE

TOUPPERF32
            INRECTYPE="VIEW:view10"
            OUTBUFTYPE="fml32"
            COUPLING=TIGHT

TOUPPERV
            INBUFTYPE="X_C_TYPE:v10"
            INRECTYPE="VIEW:upper"
            COUPLING=LOOSE

TOUPPERC
            INCRECTYPE="X_OCTET"
```

```
            COUPLING=TIGHT

TOUPPERS
            OUTRECTYPE="X_OCTET"
            OUTBUFTYPE="STRING"
            INRECTYPE="X_OCTET"

TOUPPERX
            OUTRECTYPE="STRING"
            OUTBUFTYPE="STRING"
            INRECTYPE="X_OCTET"

*DM_REMOTE_SERVICES
DEFAULT:  TRANTIME=300

ECHOXOCT
            RNAME="ECHOSRVR"
            OUTBUFTYPE="X_COMMON:ECHOVIOEW"
            RDOM=dal2200
            LDOM=dalnt8

ECHOXCOM
            RNAME="ECHOSRVR"
            RDOM=openti
            LDOM=dalnt8
            AUTOPREPARE=Y

ECHOTYPE
            RNAME="ECHOSRVR"
            INBUFTYPE="X_C_TYPE:ECHOVIEW"
            INRECTYPE="X_C_TYPE:ECHOVIEW"
            OUTBUFTYPE="X_C_TYPE:ECHOVIEW"
            OUTRECTYPE="X_C_TYPE:ECHOVIEW"

ECHOVIEW
            RNAME="ECHOSRVR"
            INBUFTYPE="VIEW:ECHOVIEW"
            INRECTYPE="X_COMMON:ECHOVIEW"
```

```
OUTBUFTYPE="VIEW:ECHOVIEW"
OUTRECTYPE="X_COMMON:ECHOVIEW"
RDOM=icl2
LDOM=dalnt8
TPSUT_TYPE="PRINTABLESTRING"
REM_TPSUT="tpmvs"
```

# Implementing Native-A Encoding

The Native-A encoding feature in TMA OSI TP converts Tuxedo views into a format that is native to Unisys MCP mainframe systems. This feature moves most of the encode/decode processing from the Unisys MCP mainframe systems to the Tuxedo system.

## Using the XATMI_ENCODING Type

Currently, the type of XATMI encoding must be configured for each RDOM using the XATMI_ENCODING parameter in the DM_OSITPX section of the DMCONFIG file. An XATMI_ENCODING keyword value of NATIVE_A_SERIES is used to indicate that the Tuxedo system will handle the encode/decode of data into the Native MCP format, not the Unisys MCP machine.

The following example is a *DM_OSITPX section of the DMCOMFIG file.

```
*DM_OSITPX
aseries1
       AET="{1.3.123.55.22},{51}"
       NWADDR="123.55.22.51"
       XATMI_ENCODING="NATIVE_A_SERIES"
       T_SEL="0x5453"
       S_SEL="0x3F5C3F"
```

## Using the CODEPAGE Parameter

There is an optional CODEPAGE parameter on the RDOM statement in the DM_REMOTE_DOMAINS section of the DMCONFIG file. The CODEPAGE parameter is configured to specify the pair of code sets involved when translating character strings between the Tuxedo system and the MCP (A-Series) system. If XATMI_ENCODING is not set to NATIVE_A_SERIES, then the CODEPAGE parameter is ignored.

The syntax is shown below:

```
*DM_REMOTE_DOMAINS
...
rdom TYPE=OSITPX DOMAINID="domainid" CODEPAGE="cpname"
```

Where *cpname* is a case-insensitive keyword from the following table.

**Table 4-2 cpname Keywords**

| cpname keyword | ISO Character Set | MCP Character Set |
|---|---|---|
| Tuxedo | ASCII | ASERIESEBCDIC |
| 88591xL1EBC | ISO 8859-1 (Latin-1) | Latin1EBCDIC |
| 88592xL2EBC | ISO 8859-2 (Latin-2) | Latin2EBCDIC |
| 88599xL5EBC | ISO 8859-9 (Latin-5) | Latin5EBCDIC |
| 885915xL9EBC | ISO 8859-15 (Latin-9) | Latin9EBCDIC |
| 88595xLCYEBC | ISO 8859-5 (Latin Cyrillic) | LatinCyrillicEBC |
| 88597xLGREBC | ISO 8859-7 (Latin Greek) | LatinGreekEBCDIC |
| 88591xIBMSEBC | ISO 8859-1 (Latin1) | IBMSwedenEBCDIC |
| AR20xAR20EBC | Arabic20ISO | Arabic20EBCDIC |

# Rules for Viewfile Character Types

If XATMI_ENCODING is not set to NATIVE_A_SERIES, then no conversion of character strings occurs. If XATMI_ENCODING is set to NATIVE_A_SERIES, then conversions occur according to the rules described in the following subsections.

## Rules for Type string

All view fields of type string must be null-terminated on both the Tuxedo and NATIVE_A_SERIES encoding feature, string fields may contain any non-zero bytes, followed by a zero byte as a null-terminator.

All view fields of type string are always translated from the ISO character set to the MCP character set (as specified by the CODEPAGE parameter) when passing from the Tuxedo system to the MCP system. The input string must be null-terminated; any bytes after the null-terminator

are ignored. The resulting string on the Unisys MCP system is null-terminated; any remaining space in the field is also padded with zero bytes.

Conversly, all view fields of type `string` are translated from the MCP character set to the ISO character set (as specified by the `CODEPAGE` parameter) when passing from the Unisys MCP system to the Tuxedo system. The input string must be null-terminated; any bytes after the null-terminator are ignored. The resulting string on the Tuxedo system is null-terminated; any remaining space in the field is also padded with zero bytes.

## Rules for Type carray

View fields of type `carray` need not be null-terminated. `Carray` fields may contain any non-zero bytes; if a zero byte is detected, it is treated as a null-terminator and scanning stops.

All view fields of type `carray` are always translated from the ISO character set to the MCP character set (as specified by the `CODEPAGE` parameter) when passing from the Tuxedo system to the Unisys MCP system. The input string may or may not be null-terminated; any bytes after a null-terminator are ignored. The resulting string on the MCP system is not null-terminated; any remaining space in the field is padded with EBCDIC space characters (0x40 bytes).

Conversly, all view fields of type `carray` are translated from the MCP character set to the ISO character set (as specified in the `CODEPAGE` parameter) when passing from the Tuxedo system to the Unisys MCP system. The input string may or may not be null-terminated; any bytes after a null-terminator are ignored. If the input string is not null-terminated, then any trailing EBCDIC space characters (0x40 bytes) are discarded before translation starts. The resulting string on the Tuxedo system is null-terminated if there is room; any remaining space in the field is also padded with zero bytes.

For existing users who plan to use the Native-A feature, it is important to note that fields of type `carray` are always translated. If you wish to transmit binary data that should not be translated, then you must change your view field type from `carray` to an array of type `char`.

## Rules for Type char

View fields of type `char` may contain any arbitrary binary data. View fields of type `char` are never translated.

For existing users who plan to use the Native-A feature, it is important to note that fields of type `char` are never translated. If you want to have fields of type `char` translated, you need to change your view field type from `char` to `carray`. See the following example.

**Listing 4-12 Example of char Field Changed to carray**

```
VIEW viewx
#
#type   cname       fbname  count  flag   size  null
#
int     accountNum   -      1      -      -     -
string  firstName    -      1      -      20    -
# Change middleInit from char to carray so it is translated
# char  middleInit   -      1      -      -     -
carray  middleInit   -      1      -      1     -
string  lasstName    -      1      -      20    -
END
```

Conversly, you may wish to change other data types (e.g., `carray`) to type `char` to prevent then from being translated. See the following example:

**Listing 4-13 Example of carray Field Changed to char**

```
VIEW viewy
#
#type   cname       fbname  count  flag   size  null
#
int     accountNum   -      1      -      -     -
string  name         -      1      -      50    -
#Change encryptKey from carray to char so it is not
#translated:
#carray encryptKey   -      1      -      15    -
char    encryptKey   -      15     -      -     -
END
```

# Enabling Data Compression

TMA OSI TP can optionally send messages to a peer DTP system compressed if the peer also supports OSI TP data compression. For compression to occur, the messages must follow these requirements.

- message must contain repeating characters
- compression must result in at least a 10% size savings
- connection must use OSI TP multiplexing
- peer has data compression turned on

To enable data compression, add the option "MULTIPLEXCOMPRESS=Y" to the EXTENSIONS parameter for the remote domain for which to send compressed data. A sample entry in the DMCONFIG file for data compression to occur would be as follows:

```
"MY-CLEARPATH"
        AET="{1.3.192.33.322.12},{2476}
        NWADDR="192.33.322.12:2476"
        EXTENSIONS="MULTIPLEXCOMPRESS=Y"
```

# Editing the DMCONFIG File

If you are upgrading from eLink OSI TP 1.3, it is recommended that you use the `osiadmin` utility to update your `DMCONFIG` file; however, you may edit the `DMCONFIG` file manually. If you are upgrading from eLink OSI TP 4.0, you do not need to change your `udmconfig` input file. You can use it as input to the TMA OSI TP 9.1 `dmloadcf` utility. Similarly, if you are upgrading from eLink OSI TP 4.1 or 4.2, you do not need to change your `dmconfig` input file. You can use it as input to the TMA OSI TP 9.1 `dmloadcf` utility. Refer to Chapter 5, "Using the OSI TP Administration Utility," for more information about the `osiadmin` utility.

To edit the `DMCONFIG` file manually, perform the following steps:

1. Find the `DMCONFIG` file in your installation directory and open it in any text editor.
2. Edit the `DMCONFIG` file as necessary. Refer to the parameter descriptions in this section for details about defining your TMA OSI TP configuration.
3. When editing is complete, save the `DMCONFIG` file.

   **Note:** You may want to save the original `DMCONFIG` file with a different name or in a different directory.

4. Process the DMCONFIG file with the dmloadcf utility. This parses the input and creates a binary file: the BDMCONFIG file, which is used by GWOSITP.Refer to Understanding the DMCONFIG File for more detailed information about the parameters in the DMCONFIG file.

# Steps for Modifying the DMCONFIG File Parameters

Perform the following steps to modify the DMCONFIG file parameters:

- Step 1 - Define Local Domains
- Step 2 - Define Remote Domains
- Step 3 - Specify Addressing Information for OSI TP Domains
- Step 4 - Specify Access Control for OSI TP Domains
- Step 5 - Specify Available Local Tuxedo Services
- Step 6 - Specify Available Remote Tuxedo Services
- Step 7 - Specify Routing Information

## Step 1 - Define Local Domains

You must define the local domains that use the OSI TP server group you defined in your Tuxedo UBBCONFIG file. Refer to Defining TMA OSI TP Servers for BEA Tuxedo for more information about the UBBCONFIG file.

Perform the following steps to define a local domain in the DM_LOCAL_DOMAINS section of the DMCONFIG file:

1. Specify the local domain ID with the DOMAINID parameter.
2. Specify the gateway group you defined in the UBBCONFIG file with the GWGRP parameter.
3. Specify the domain type of OSITPX with the TYPE parameter.
4. Specify the size of the domain transaction log with the DMTLOGSIZE parameter.
5. Specify any of the optional DM_LOCAL_DOMAINS parameters that you require: AUDITLOG, BLOCKTIME, DMTLOGDEV, DMTLOGNAME, MAXRDTRAN, MAXTRAN, and SECURITY.

**Listing 4-14 Example of DM_LOCAL_DOMAINS Section**

```
*DM_LOCAL_DOMAINS

dalnt8
GWGRP       = OSIGRP
TYPE        = OSITPX
DOMAINID    = "dalnt8"
BLOCKTIME   = 30
DMTLOGDEV   = "D:\tuxedo\log\DMLOG"
DMTLOGNAME  = "DMLOG"
SECURITY     = DM_PW  # turns link layer security on
```

Refer to Sample Configuration File for more detailed information.

## Step 2 - Define Remote Domains

It is recommended that you use the `importcfg` command in the `osiadmin` utility to update remote domains if you are upgrading from eLink OSI TP 1.3; however, you can manually define remote domains. Refer to "Using the OSI TP Administration Utility" for more information about the `osiadmin` utility.

Perform the following steps to define remote domains in the `DM_REMOTE_DOMAINS` section of the `DMCONFIG` file:

1. Specify the remote domain ID with the `DOMAINID` parameter.
2. Specify the `OSITPX` domain type with the `TYPE` parameter.

There are no optional parameters for the `DM_REMOTE_DOMAINS` section.

**Listing 4-15 Example of DM_REMOTE_DOMAINS Section**

```
*DM_REMOTE_DOMAINS

dal2200 TYPE=OSITPX DOMAINID="dal2200"
openti  TYPE=OSITPX DOMAINID="openti"
```

```
icl2    TYPE=OSITPX DOMAINID="icl2"
aseries1 TYPE=OSITPX DOMAINID="aseries1"
```

Refer to DM_REMOTE_DOMAINS Section for more detailed information.

## Step 3 - Specify Addressing Information for OSI TP Domains

Perform the following steps to define addressing information for OSI TP domains in the DM_OSITPX section of the DMCONFIG file:

1. Specify the Application Entity Title for each local and remote OSI TP domain with the AET parameter.
2. Specify the IP address or DNS name and port number for each local and remote OSI TP domain with the NWADDR parameter. If you are using multiple IP addresses, make sure to enter all the addresses on one line, and separate them with a semi-colon (;). Put double quotes around the entire address.
3. Specify any of the optional DM_OSITPX parameters that you require: DNS_RESOLUTION, P_SEL, S_SEL, T_SEL, OPTIONS, TAILOR_PATH, and XATMI_ENCODING.

**Listing 4-16 DM_OSITPX Section**

```
*DM_OSITPX

dalnt8
               AET="{1.3.144.23.103},{208}"
               TAILOR_PATH="d:\tuxedo\configs\tailor.txt"
               NWADDR="//dalnt8:102"
               DNS_RESOLUTION=STARTUP # this is the default

dal2200
               AET="{1.3.132.61.146},{3}"
               XATMI_ENCODING="OLTP_TM2200"
               NWADDR="132.61.146.3;132.61.147.1"
               T_SEL="OSITP"

openti
```

```
                AET="{1.3.122.62.103},{209}"
                NWADDR="122.62.103.209"

icl2
              AET="{1.3.142.60.203},{4}"
              NWADDR="142.60.203.4"
              T_SEL="ICLTP"
              S_SEL="SSEL"
              P_SEL="PSEL"


aseries1
              AET="{1.3.123.55.222},{51}"
              NWADDR="123.55.222.51"
              XATMI_ENCODING="PRELIMINARY"
              T_SEL="0x5453"
              S_SEL="0x3F5C3F"
              OPTIONS=SECURITY_SUPPORTED
```

Refer to DM_OSITPX Section for more detailed information.

## Step 4 - Specify Access Control for OSI TP Domains

In the DM_ACCESS_CONTROL section of the DMCONFIG file, specify a list of all the remote OSI TP domain IDs that can access the local domain with the ACLIST parameter. This parameter is optional.

**Listing 4-17 Example of DM_ACCESS_CONTROL Section**

```
*DM_ACCESS_CONTROL
mylist  ACLIST = dalnt8, dal2200
```

Refer to DM_ACCESS CONTROL Section for more detailed information.

## Step 5 - Specify Available Local Tuxedo Services

In the DM_LOCAL_SERVICES section of the DMCONFIG file, specify the Tuxedo services that will be made available to OSI TP applications and define their options with the ACL, COUPLING, INBUFTYPE, INRECTYPE, LDOM, OUTBUFTYPE, OUTRECTYPE, and RNAME parameters. If the local service supports transactions, make sure the group it belongs to contains a TMS name.

These DM_LOCAL_SERVICES parameters are all optional.

**Listing 4-18 Example of DM_LOCAL_SERVICES Section**

```
*DM_LOCAL_SERVICES
TOUPPERF
  INRECTYPE="VIEW:view10"
  OUTBUFTYPE="FML:"
  COUPLING=LOOSE

TOUPPERF32
  INRECTYPE="VIEW:view10a"
  OUTBUFTYPE="FML32:"
  COUPLING=TIGHT


TOUPPERV
  INBUFTYPE="X_C_TYPE:v10"
  INRECTYPE="VIEW:upper"
  COUPLING=LOOSE

TOUPPERC OUTRECTYPE="X_OCTET" OUTBUFTYPE="CARRAY"
        INRECTYPE="X_OCTET"
        COUPLING=TIGHT

TOUPPERS  OUTRECTYPE="X_OCTET" OUTBUFTYPE="STRING"
        INRECTYPE="X_OCTET"

TOUPPERX  OUTRECTYPE="STRING" OUTBUFTYPE="STRING"
         INRECTYPE="X_OCTET"
```

Refer to DM_LOCAL_SERVICES Section for more detailed information.

## Step 6 - Specify Available Remote Tuxedo Services

In the DM_REMOTE_SERVICES section of the DMCONFIG file, specify the remote services that can be requested by Tuxedo applications and define their options with the AUTOPREPARE, CODEPAGE, CONV, INBUFTYPE, INRECTYPE, LDOM, OUTBUFTYPE, OUTRECTYPE, RDOM, RNAME, ROUTING, and TRANTIME parameters. These parameters are all optional.

**Listing 4-19 Example of DM_REMOTE_SERVICES Section**

```
*DM_REMOTE_SERVICES
DEFAULT: TRANTIME=300

ECHOXOCT RNAME="ECHOSRVR" OUTBUFTYPE="X_COMMON:ECHOVIEW" RDOM=dal2200
LDOM=dalnt8
ECHOXCOM RNAME="ECHOSRVR" RDOM=openti LDOM=dalnt8 AUTOPREPARE=Y

ECHOXCTYPE RNAME="ECHOSRVR"
         INBUFTYPE="X_C_TYPE:ECHOVIEW"
         INRECTYPE="X_COMMON:ECHOVIEW"
         OUTBUFTYPE="X_C_TYPE:ECHOVIEW"
         OUTRECTYPE="X_COMMON:ECHOVIEW"
         RDOM=aseries1
         LDOM=dalnt8
         CONV=Y
ECHOVIEW RNAME="ECHOSRVR"
         INBUFTYPE="VIEW:ECHOVIEW"
         INRECTYPE="X_COMMON:ECHOVIEW"
         OUTBUFTYPE="VIEW:ECHOVIEW"
         OUTRECTYPE="X_COMMON:ECHOVIEW"
         RDOM=icl2
         LDOM=dalnt8
         TPSUT_TYPE="PRINTABLESTRING"
         REM_TPSUT="tpmvs"
```

Refer to DM_REMOTE_SERVICES Section for more detailed information.

### Step 7 - Specify Routing Information

Perform the following steps to define routing information for service requests in the DM_ROUTING section of the DMCONFIG file:

1. Specify the name of the routing field with the FIELD parameter.
2. Specify the data buffer type and subtype for which the routing entry is valid with the BUFTYPE parameter.
3. Specify the ranges and associated remote domain names for the routing field with the RANGES parameter.

**Listing 4-20 Example of DM_ROUTING Section**

```
*DM_ROUTING
ACCOUNT FIELD = branchid BUFTYPE = "View:account"
        RANGE = "MIN - 1000:aseries1, 1001-3000:openti, *:dal2200"
```

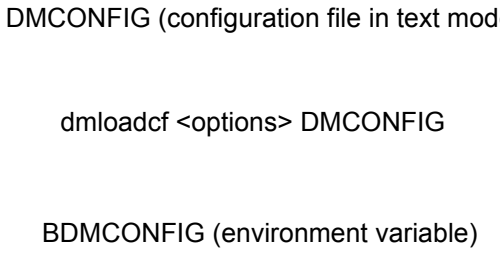
Refer to DM_ROUTING Section for more detailed information.

## Processing a Configuration File with the dmloadcf Utility

The dmloadcf utility compiles the DMCONFIG file and creates a binary configuration file, BDMCONFIG, which is used by the DMADM server to control the run-time environment.

Figure 4-3 shows how the dmloadcf utility processes the configuration file. A description of the process follows the figure.

**Figure 4-3 dmloadcf Process**




# Invoking the dmloadcf Utility

The `dmloadcf` utility is invoked from a command line with the following syntax:

`dmloadcf [-c] [-n] [-y] [-b` *`blocks`*`] [-k]` *`DMCONFIG_file`*

where the following options are valid:

**-c**

Prints minimum IPC resources needed for each local domain (gateway group) in this configuration. The `BDMCONFIG` file is not updated.

**-n**

Checks only the syntax of the ASCII `DMCONFIG` file without actually updating the `BDMCONFIG` file.

**-y**

Suppresses a prompt to create and initialize the `BDMCONFIG` file. This parameter **must** be entered before the `DMCONFIG` file name.

**-b blocks**

Indicates the number of blocks the device should use to create the Tuxedo file system. If the value of the `-b` option is large enough to hold the new `BDMCONFIG` tables, `dmloadcf` uses the specified value to create the new file system; otherwise, `dmloadcf` prints an error message and exits. If the `-b` option is not specified, `dmloadcf` creates a new file system large enough to hold the `BDMCONFIG` tables. The `-b` option is ignored if the file system already exists. The `-b` option is highly recommended if `BDMCONFIG` is a raw device (that has not been initialized) and should be set to the number of blocks on the raw device.

**dmconfig_file**

Specifies the name of the input configuration file to `dmloadcf`.

# How the dmloadcf Utility Works

The dmloadcf utility prints an error message if any required section of the DMCONFIG file is missing. If a syntax error is found while parsing the input file, dmloadcf exits without performing any updates to the BDMCONFIG file.

A Tuxedo DMTYPE file is required to define the valid domain types. If this file does not exist, *dmloadcf* exits without performing any updates to the BDMCONFIG file.

The effective user ID of the person running dmloadcf must match the UID in the RESOURCES section of the TUXCONFIG file.

After syntax checking, dmloadcf verifies that the file pointed to by BDMCONFIG exists, is a valid Tuxedo System file system, and contains BDMCONFIG tables. If these conditions are not true and the -y option was not entered on the command line, the user is prompted to create and initialize the file with

```
Initialize BDMCONFIG file: path {y, q}?
```

where *path* is the complete file name of the BDMCONFIG file and "Y" indicates that the configuration file should be created.

If the BDMCONFIG file is determined to already have been initialized, dmloadcf ensures that the local domain described by that BDMCONFIG file is not running. If a local domain is running, dmloadcf prints an error message and exits. Otherwise, dmloadcf confirms that the file should be overwritten by prompting the user with:

```
"Really overwrite BDMCONFIG file {y, q}?"
```

Prompting is suppressed if the standard input or output are not terminals. Any response other than "y" or "Y" causes dmloadcf to exit without creating the configuration file. If the BDMCONFIG file is not properly initialized and the user has responded with "Y", dmloadcf creates the Tuxedo file system and then creates the BDMCONFIG tables.

If the SECURITY parameter is specified in the RESOURCES section of the TUXCONFIG file, then dmloadcf flushes the standard input, turns off terminal echo, and prompts the user for an application password.

Assuming no errors, and if all checks have passed, dmloadcf loads the DMCONFIG file into the BDMCONFIG file and overwrites all existing information found in the BDMCONFIG tables.

The following example shows how a binary configuration file is loaded from the bank.DMCONFIG ASCII file. The BDMCONFIG device is created (or reinitialized) with 2000 blocks:

```
dmloadcf -b 2000 -y bank.dmconfig
```

### Diagnostics

If an error is detected in the input, the erroneous line is printed to the standard error log along with a message indicating the problem. If a syntax error is found in the DMCONFIG file or the system is currently running, no information is updated in the BDMCONFIG file and dmloadcf exits.

If dmloadcf is run on an active node, the following error message is displayed:

```
*** dmloadcf cannot run on an active node ***
```

If dmloadcf is run by a person whose effective user ID doesn't match the UID specified in the TUXCONFIG file, the following error message is displayed:

```
*** UID is not effective user ID ***
```

Upon successful completion, dmloadcf exits. If the BDMCONFIG file is updated, a userlog message is generated to record this event.

## Tuning OSI TP-Specific Tables with the TAILOR File

The OSI TP TAILOR file is external to the DMCONFIG and is used for tuning OSI TP- specific tables. All parameters in the TAILOR file are optional with preset defaults.

Following is a list of valid TAILOR parameters:

**Table 4-3 Parameters for OSI TP TAILOR FIle**

| Parameter | Default | Description |
|---|---|---|
| * FreeOldRetryTime | 600 seconds | Time in seconds between automatic terminations of old connections |
| MaxConnections | 500 | Maximum size of various internal tables |
| ** KeepAliveCheck | 60 | Time in seconds of connection inactivity before a keepalive packet is sent on the connection. |

**Table 4-3 Parameters for OSI TP TAILOR FIle**

| Parameter | Default | Description |
|---|---|---|
| `** KeepAliveTimeout` | 10 | Time in seconds to wait for an acknowledgement message when a keepalive packet is sent. If this timeout is exceeded, then the connection is re-established. |
| `MaxRemoteNodes` | 1000 | Maximum number of total remote domains |
| `* OldAssocTimeout` | 3600 seconds | Time in seconds denoting an "old" connection (association) |
| `* RdomAssocRetry` | 60 seconds | Time in seconds between automatic retries of associations to unavailable RDOMS. |
| `* TCPSocketsKeepAlives` | N | Toggle for TCP keepalive packets |
| `** StartFlowControlThreshold` | 1,048,576 | Number of bytes of data which may be buffered to a particular RDOM before flow control is started. |
| `** StopFlowControlThreshold` | 102,400 | Number of bytes that the data buffered to the RDOM must fall below before flow control is relieved (when flow control is in effect.) |
| `TCPSocketsLinger` | -1 | Amount of time TCP/IP socket connection stays open |
| `TCPSocketsListenQueueDepth` | 5 | Number of held TCP/IP connections waiting to be accepted by TMA OSI TP |

**Table 4-3 Parameters for OSI TP TAILOR FIle**

| Parameter | Default | Description |
|---|---|---|
| TCPSocketsNoDelay | N | Toggle to delay sends of TCP/IP packets until an ACK is received from remote machine |
| TraceIpcKey | 32800 | IPC key value for the OSI TP log and trace shared memory section |

**Note:** The parameters in the previous table with an asterisk (*) are valid for non-multiplexed connections only.

The parameters with double asterisks (**) apply only if you are using the multiplexing protocol.

Following is more detailed information about each of the TAILOR file parameters:

**FreeOldRetryTimer = *numeric***

Specifies the time in seconds between automatic terminations of old connections (associations). OSI TP reuses established socket connections to a remote domain. The default value is 600 seconds.

**Note:** This parameter is valid for non-multiplexed connections only.

**KeepAliveCheck = *numeric***

Specifies the time in seconds of connection inactivity before a keepalive packet is sent on the connection. The default value is 60 seconds.

**Note:** This parameter is only valid when using the multiplexing protocol.

**KeepAliveTimeout = *numeric***

Specifies the amount of time in seconds to wait for an acknowledgement message when a keepalive packet is sent. If this timeout is exceeded, then the connection is re-established. The default is 10 seconds.

**Note:** This parameter is only valid when using the multiplexing protocol.

**MaxConnections = *numeric***

Specifies the size of various internal tables. This value should be at least as large as the maximum number of total remote domains, including those added dynamically. The default value is 1000.

**MaxRemoteNodes =** `numeric`

Specifies the maximum number of total remote domains, including those added dynamically. The default value is 1000.

**OldAssocTimeout =** ***numeric***

Specifies the time in seconds denoting an "old" connection (association). Any connection to a remote domain that remains unused by a `tpcall()` for this amount of time is subject to automatic termination. This default value is 3600 seconds.

**Note:** This parameter is valid for non-multiplexed connections only.

**RdomAssocRetry =** ***numeric***

Specifies the time in seconds between automatic retries of associations to unavailable RDOMs. This value must be greater than zero. The default value is 60. This value may be overridden on each RDOM with the `EXTENSION` parameter and the `RdomAssocRetry` keyword.

**Note:** This parameter is valid for non-multiplexed connections only.

**StartFlowControlThreshold =** ***numeric***

Specifies the number of bytes of data which may be buffered to a particular RDOM before flow control is started. When the data buffered to one particular RDOM exceeds this value, the services advertised for that RDOM/LDOM combination are suspended. The default value is 1,048,576 bytes.

**Note:** This parameter is only valid when using the multiplexing protocol.

**StopFlowControlThreshold =** ***numeric***

Specifies the number of bytes that the data buffered to the RDOM must fall below before flow control is relieved (when flow control is in effect.) When buffering drops below this value for one particular RDOM, the services for that RDOM/LDOM combination are advertised again. The default value is 102,400 bytes.

**Note:** This parameter is only valid when using the multiplexing protocol.

**TCPSocketsKeepAlives = {Y | N}**

Specifies whether the TCP keepalive packets are sent. This is useful to insure the integrity of the TCP connection. If `Y` is specified the TCP/IP packets are sent out in a time period specified by the operating system. Most operating systems use a 2 hour time period. The default is `N`.

**Note:** This parameter is valid for non-multiplexed connections only.

**TCPSocketsLinger = {-1 | 0 | *numeric*}**

Specifies the amount of the time that the TCP/IP socket connection will stay open. This can be used to timeout hung connections.

Possible values are:

| | |
|---|---|
| `-1` (default) | Time that a TCP/IP socket connection stays in the `TIME_WAIT` state is determined by the operating system. |
| `0` | TCP/IP connection is closed immediately. |
| `n` | TCP/IP connection stays in the `TIME_WAIT` state for `n` seconds before closing. |

**TCPSocketsListenQueueDepth = *numeric***

Specifies the number of held TCP/IP connections waiting to be accepted by TMA OSI TP.

Possible values are:

| | |
|---|---|
| `5` (default) | A minimum of 5 incoming TCP/IP connections are held. |
| `>5` | More than 5 incoming TCP/IP connections are held. The operating system may only supports a number up to a "reasonable value". |

**TCPSocketsNoDelay = {Y | N}**

Specifies if subsequent sends of TCP/IP packets are held until an `ACK` is received from a remote machine. The default is `N`, the subsequent sends of TCP/IP packets may be held until an `ACK` is received. If `Y` is specified, the TCP/IP packets are sent immediately without waiting for an `ACK` of the previous send.

**TraceIpcKey = *numeric***

For UNIX systems, specifies the IPC key value for the OSI TP log and trace shared memory segment. If there are multiple local domains, then each domain must have a unique IPC key value. On Windows NT systems, this value is ignored. The default is 32800.

# Enabling Automatic Suspension of Remote Services

The BEA TMA OSI TP service suspend feature provides greater reliability for remote services by automatically suspending services for a remote domain that is unreachable. Automatic suspension prevents repeated calls to remote services that are no longer available. This feature is especially critical when load balancing between remote domains.

TMA OSI TP provides this service suspend feature by maintaining at least one association to every remote RDOM to determine its availability. If an association cannot be established, the services associated with that RDOM will immediately fail. Redundantly defined services will not attempt to use the unavailable `RDOM` and are diverted.

By default, TMA OSI TP will retry connections to an RDOM that has failed every 60 seconds. This delay may be overridden with the `RdomAssocRetry` tailor parameter described in the "Tuning OSI TP-Specific Tables with the TAILOR File" section. It may also be overridden for specific RDOMs using the `EXTENSIONS="RdomAssocRetry=`*`n`*`"` parameter in the `DM_OSITPX` section, where *`n`* is the number of seconds to wait between retries. The value specified on the `EXTENSIONS` parameter overrides the corresponding tailor parameter. Refer to "DM_OSITPX Section" for more information.

In test environments, RDOMs may be configured that are unavailable and are considered offline. TMA OSI TP continually tries to connect to these systems. To disable the RDOMs and the remote services configured to them, specify `EXTENSIONS="ONLINE=N"` on each offline RDOM in the `DM_OSITPX` section of the `DMCONFIG` file. At run-time, these RDOMs can be brought online using the `ONLINE osiadmin` command described in "Using osiadmin Commands".

CHAPTER 5

# Using the OSI TP Administration Utility

This section covers the following topics:



## About the osiadmin Processor

Administration of the OSITP functions for BEA Tuxedo Mainframe Adapter for OSI TP is provided by the `osiadmin` processor. Administration commands are input from a command line or script file. `osiadmin` can be run regardless of whether or not the TMA OSI TP process is booted, allowing the user to perform configuration utilities, such as export and import, independently.

The `osiadmin` processor provides the following features:

- Sets internal traces while the OSITP process is up and running.
- Provides a snapshot dump of `NW_BEA` and `NW_UNISYS` internal tables.
- Performs test of OSITP connection to a remote domain.
- Imports configurations from eLink OSI TP 1.3 (`DMCONFIG`, `OSITP` configuration, and `ULS` configuration) and creates a comparable configuration in proper format for eLink OSI TP 4.x or BEA Tuxedo Mainframe Adapter for OSI TP 8.1 or 9.1.

- Exports the dmconfig file and creates the correct format of the config-in.txt file needed for the UNISYS ClearPath systems.
- Administers multiple local domains on the same system through one osiadmin processor.
- Sends optional trace messages to the Tuxedo ULOG.

# Initiating osiadmin

The osiadmin processor can be initiated in interactive, script, or batch mode. The OSIRUNDIR environment variable must be set in order to run the osiadmin utility. Refer to Setting Environment Variables for more information.

## Initiating osiadmin in Interactive Mode

Enter the following interactive command at the command line:

```
osiadmin [LDOM]
```

The local domain name (LDOM) may be specified at osiadmin execution time. The LDOM is the LDOM identifier specified in the Local_Domains section of the DMCONFIG file.

**Note:** The LDOM identifier is case sensitive.

For example,

```
osiadmin BA.CENTRAL01
```

where BA.CENTRAL01 is the LDOM.

The LDOM is necessary for CONNECTIONSTATUS, DUMP, TRACE, TESTCONNECTION, and LISTRDOM commands. If the LDOM is not specified on the command line, the user must enter an LDOM command before using any of these commands.

## Initiating osiadmin in Script Mode

Enter the following command at the command line:

```
osiadmin [LDOM]<input_script_name>osiadmin_output
```

where input_script_name is the name of a script file that you created with a series of one or more commands you wish to execute. If the script file is in a different directory, you must also enter the directory path. The script file is a text file that you can create with any text editor. Following is an example run of a script that might be used with osiadmin:

**Listing 5-1 Example of an Osiadmin Input Script**

```
>page off             Do not prompt for long output
>echo on              Echo input commands
>LDOM BA.CENTRAL01    Set local domain name for
                         following commands
>lr                   List remote domains in
                         BA.CENTRAL01 configuration
 BA.BRANCH01
 BA.BRANCH02
>tc BA.BRANCH01       Verify connection between
                         BA.CENTRAL01 and BA.BRANCH01
<Successfully connected to BA.BRANCH01
>tc BA.BRANCH02       Verify connection between
                         BA.CENTRAL01 and BA.BRANCH02
 Successfully connected to BA.BRANCH02
>exit
```

# Initiating osiadmin in Batch Mode

Enter the following command at the command line:

```
osiadmin -b
```

The "-b" option initiates the `osiadmin` in batch mode with the following conditions: the menu of commands that is normally displayed upon startup of `osiadmin` is not printed, pagination is set to off, command echo is set to on, and interactive prompts (for example, "Overwrite file, Y or N") are suppressed.

# What Happens When You Execute an osiadmin Command?

When you initiate `osiadmin` or enter any of the other commands in interactive mode, the command executes if the syntax is correct. If the syntax is incorrect, an error message displays showing the correct syntax. When the command executes, results are printed to the screen. If the result is more than one page and `PAGINATE=ON` (which is the default), the prompt, "`CONTINUE? Y or N`" displays. Select `Y` to continue displaying another 20 lines or `N` to discontinue displaying the results.

At any time after a command has been executed, you can press `Enter` to redisplay the command syntax as shown by the Help command.

When you initiate `osiadmin` and enter commands using a script, the commands execute in order and the results are printed to the screen as the commands execute.

# Using osiadmin Commands

All `osiadmin` commands are initiated through interactive mode or through a script containing one or more commands. The following sections explain how to get Help for `osiadmin` commands and provide descriptions for each command in detail.

## Getting Help for osiadmin Commands

You can request a list of the valid `osiadmin` commands, request help for specific commands, or display command syntax after a command has been executed.

**To display Help in interactive mode:**

You can display a list of valid commands by entering the following command at the command line:

```
HELP
```

**To display Help for a specific command:**

Enter the following command at the command line:

```
HELP command
```

where `command` is any of the valid commands.

The exact syntax for the specified command displays.

## osiadmin Commands

Following are descriptions of the valid `osiadmin` commands.

### CONNECTIONSTATUS

Allows a user to display the connection status for one or more `RDOMS` or a pattern containing wildcard characters (*). For example, `CONNECTIONSTATUS *` lists the connection status for all `RDOMS`. `CONNECTIONSTATUS x*` lists the connection status for all `RDOMS` starting with the letter x.

When a wildcard character is used to specify the pattern for the CONNECTIONSTATUS command, only those RDOMs using the same protocol as the current LDOM are considered. For example, if the current LDOM is using the multiplex protocol, only the RDOMS configured with MULTIPLEX=Y are checked for connection status.

Syntax:

```
CONNECTIONSTATUS RDOM|pattern
```

OR

```
CS RDOM|pattern
```

## DUMP

Produces a dump of internal TMA OSI TP tables including the `NW_BEA` and the `NW_UNISYS` portion of code for diagnostic purposes. If a filename is entered, that file is used for the dump. If the filename is not entered, the default dump file, `OSITPDUMP.TXT`, is used.

Syntax:

```
DUMP [filename]
```

OR

```
DU [filename]
```

## ECHO

Displays input command lines as they are entered when set to `ON`. If no option is given, the current setting is toggled and the new setting is printed. The initial setting is `OFF`.

Syntax:

```
ECHO [OFF|ON]
```

OR

```
E [OFF|ON]
```

## EXPORTCFG

Exports TMA OSI TP configuration information from the `DMCONFIG` file and creates configuration files readable by OSI-TP on UNISYS ClearPath servers. The names of the files produced are `<output directory>TAILOR-IN.TXT` and `<output`

directory>CONFIG-IN.TXT. The output directory is optional. The current directory is the default if none is specified.

Syntax:

```
EXPORTCFG RDOM DMCONFIG-SOURCE [<output directory>]
```

OR

```
EXCFG RDOM DMCONFIG-SOURCE [<output directory>]
```

## IMPORTCFG

Imports the correct configuration information for the TMA gateway by automatically upgrading configurations from version 1.3 of the eLink gateway. `importcfg` specifies the `DMCONFIG` file, `OSITP` configuration file, and the `ULS` configuration file to be used for input. The `DMCONFIG` file is used as input to the `dmloadcf` processor. A `DMCONFIG` file is created as the output. The `ULS-TAILOR` and `NEW-TAILOR` files are optional and are only valid on Unix. If the output file already exists, a message is sent to confirm before overwriting the file. If no parameters are specified, the user is prompted for the filenames.

Syntax:

```
IMPORTCFG [<outfile> DMCONFIG-SOURCE OSITP-CONFIGFILE {ULS-TAILOR-SOURCE
NEW-TAILOR-SOURCE}]
```

OR

```
IMCFG [<outfile> DMCONFIG-SOURCE OSITP-CONFIGFILE {ULS-TAILOR-SOURCE
NEW-TAILOR-SOURCE}]
```

**Note:** `OSITP-CONFIGFILE` is the source configuration file and not the `current.cfg` file that was required for older versions of BEA Tuxedo Mainframe Adapter for OSI TP.

## LDOM

Sets the local domain name. The `LDOM` must be set for the `DUMP, CONNECTIONSTATUS, TRACE, TESTCONNECTION`, and `LISTRDOM` commands. The `LDOM` can also be specified as an argument when executing the `osiadmin`.

Syntax:

```
LDOM [LDOM name]
```

## LISTRDOM

Retrieves and displays a list of all RDOMs. The list indicates if the RDOMs are offline or online, indicates if the online RDOMs are currently available, and indicates that communications with unavailable RDOMs are not operational. TMA OSI TP will try to establish communications until successful.

When the LISTRDOM command is run, only those RDOMs using the same protocol as the current LDOM are displayed. For example, if the current LDOM is using the multiplex protocol, only the RDOMS configured with MULTIPLEX=Y are listed.

### Syntax:

```
LISTRDOM
```

OR

```
LR
```

## ONLINERDOM

Resets RDOM configured to be offline at startup to online. Once an RDOM is set OFFLINE, it must be manually reset to ONLINE before communications with the peer can continue. All remote services associated with the RDOM will be advertised.

The ONLINERDOM command changes the state of an RDOM to online. The state also immediately changes to available.

### Multiplex Protocol

If the current LDOM and the specified RDOM are using multiplexed protocol, and the multiplex policy is STARTUP, then TMA OSI TP immediately attempts to establish a connection from the LDOM to the RDOM. If this fails, the state changes to online and unavailable; then connection attempts are retried until successful.

If the current LDOM and the specified RDOM are using multiplexed protocol, and the multiplex policy is DEMAND, then TMA OSI TP does not attempt to establish a connection from the LDOM to the RDOM until it is needed to perform a CALL or CONNECTION. If this later connection fails, the state changes to online and unavailable; then connection attempts are retried until successful.

### Non-Multiplex Protocol

If the current LDOM and the specified RDOM are using non-multiplexed protocol, then TMA OSI TP immediately attempts to establish a connection from the LDOM to the RDOM. If this

later connection fails, the state changes to online and unavailable; then connection attempts are retried until successful.

### Syntax:

```
ONLINERDOM RDOM
```

OR

```
ONL RDOM
```

## PAGINATE

Turns paginate mode ON or OFF. Default is ON.

### Syntax:

```
PAGINATE {OFF|ON}
```

OR

```
PAGE   {OFF|ON}
```

## TESTCONNECTION

Use the TESTCONNECTION command to verify that a connection to the RDOM can be established. This is useful to verify that the configuration is correct and the network is working properly.

### Multiplex Protocol

If the current LDOM is using multiplexed protocol, this command checks to see if a connection already exists. If so, the command returns immediately indicating success. If no connection exists, this command attempts to establish a new connection and waits for the result before returning. The new connection is left open for future use.

### Non-Multiplex Protocol

If the current LDOM is using non-multiplexed protocol, this command attempts to establish a new connection and waits for the result before returning. Then the new connection is closed.

### Syntax:

```
TESTCONNECTION RDOM
```

OR

```
TC RDOM
```

## TRACE

Sets trace levels in TMA OSI TP for diagnostics and debugging purposes. When `TRACE` is set to `ON`, messages are written to the Tuxedo `ULOG`. If the `TRACE` command is entered without any options, it returns the current trace level.

### Syntax:

```
TRACE {OFF|ON}
```

OR

```
TR {OFF|ON}
```

# Utilities Reference

This section covers the following utilities that are commonly used for TMA OSI TP:

- DMADM
- dmadmin
- GWADM Tuxedo Administrator's Guide

## DMADM

/Domain administrative server.

### SYNOPSIS

```
DMADM SRVGRP = "identifier"
SRVID = "number"
REPLYQ = "N"
```

### DESCRIPTION

The /DOMAIN administrative server DMADM is a Tuxedo-supplied server that provides run-time access to the BDMCONFIG file. When DMADM is booted, the BDMCONFIG environment variable should be set to the pathname of the file containing the binary version of the DMCONFIG file.

DMADM is described in the *SERVERS section of the UBBCONFIG file as a server running within a group, e.g., DMADMGRP. There should be only one instance of the DMADM running in this group and it must not have a reply queue (REPLYQ must be set to "N").

The following server parameters can also be specified for the DMADM server in the *SERVERS section: SEQUENCE, ENVFILE, MAXGEN, GRACE, RESTART, RQPERM and SYSTEM_ACCESS.

## PORTABILITY

DMADM is supported as a Tuxedo-supplied server on non-/WS System operating systems.

### INTEROPERABILITY

The initial release of TMA OSI TP can only be installed on a node running Tuxedo Release 6.5 or Release 7.1.

## EXAMPLES

The following example illustrates the definition of the administrative server and a gateway group in the UBBCONFIG file.

```
#
*GROUPS
DMADMGRP  LMID=mach1 GRPNO=1
gwgrp     LMID=mach1 GRPNO=2
#
*SERVERS
DMADM SRVGRP="DMADMGRP" SRVID=1001 REPLYQ=N RESTART=Y GRACE=0
GWADM SRVGRP="gwgrp" SRVID=1002 REPLYQ=N RESTART=Y GRACE=0
GWOSITP SRVGRP="gwgrp" SRVID=1003 RQADDR="gwgrp"
RESTART=Y MIN=1 MAX=1
```

## SEE ALSO

*Tuxedo /Domain Guide, Tuxedo Administrator's Guide*

# dmadmin

OSITP Domain Administration Command Interpreter.

## SYNOPSIS

```
dmadmin [-c]
```

## DESCRIPTION

dmadmin is an interactive command interpreter used for the administration of domain gateway groups defined for a particular OSITP application. dmadmin can operate in two modes: administration mode and configuration mode.

dmadmin enters *administration* mode when called with no parameters. This is the default. In this mode, dmadmin can be run on any active node (excluding workstations) within an active application. Application administrators can use this mode to obtain or change parameters on any active domain gateway group.

**Note:** dmadmin is one of the three methods you can use to modify your configuration. Refer to "Methods for Modifying Configurations" for more information.

Application administrators may also use this mode to create, destroy, or reinitialize the DMTLOG for a particular local domain. In this case, the domain gateway group associated with that local domain must not be active, and dmadmin must be run on the machine assigned to the corresponding gateway group.

dmadmin enters *configuration* mode when it is invoked with the -c option or when the *config* subcommand is invoked. Application administrators can use this mode to update or add new configuration information to the binary version of the domain configuration file BDMCONFIG.

dmadmin requires the use of the DOMAIN administrative server DMADM for the administration of the BDMCONFIG file and the gateway administrative server GWADM for the re-configuration of active DOMAIN gateway groups (there is one GWADM per gateway group).

## ADMINISTRATION MODE COMMANDS

Once dmadmin has been invoked, commands may be entered at the prompt (">") according to the following syntax:

```
command [arguments]
```

Several commonly occurring arguments can be given default values via the default command. Commands that accept parameters set via the default command check default to see if a value has been set. If no value is set, an error message is returned.

Once set, a default value remains in effect until the session is ended, unless changed by another default command. Defaults may be overridden by entering an explicit value on the command line, or unset by entering the value "*". The effect of an override lasts for a single instance of the command.

Output from dmadmin commands is paginated according to the pagination command in use (see the definition for the paginate subcommand).

Commands may be entered either by their full name or their abbreviation (shown in parentheses) followed by any appropriate arguments. Arguments appearing in square brackets, [], are optional; those in curly braces, {}, indicate a selection from mutually exclusive options. Note that for many

commands *local_domain_name* is a required argument, but note also that it can be set with the default command.

The following commands are available in administration mode

`addumap [ options ]`
: Add local user mappings to remote user mappings for a local/remote domain pair. Mappings are defined to be inbound, outbound or both. See the `addumap` manual page for an explanation of the available options and for examples.

`addusr (addu) [ options ]`
: Add remote usernames and passwords to the remote user and password tables of a remote domain. See the `addusr` manual page for an explanation of the available options and for examples.

`advertise (adv) -d` *local_domain_name* `[{ -all | service}]`
: Advertise all remote services provided by the named local domain or the specified remote service.

`audit (audit) -d` *local_domain_name* `[{off | on}]`
: Activate (on) or deactivate (off) the audit trace for the named local domain. If no option is given, then the current setting will be toggled between the values `on` and `off`, and the new setting will be printed. The initial setting is `off`.

`chbktime (chbt) -d` *local_domain_name* `-t` *bktime*
: Change the blocking timeout for a particular local domain.

`config (config)`
: Enter configuration mode. Commands issued in this mode follow the conventions defined in the section "CONFIGURATION MODE COMMANDS" on page A-7.

`crdmlog (crdlg) -d` *local_domain_name*
: Create the domain transaction log for the named local domain on the current machine (that is, the machine where `dmadmin` is running). The command uses the parameters specified in the `DMCONFIG` file. This command fails if the named local domain is active on the current machine or if the log already exists.

`default (d) [-d` *local_domain_name*`]`
: Set the corresponding argument to be the default local domain. Defaults may be unset by specifying "*" as an argument.

  If the `default` command is entered with no arguments, the current default values are printed.

`delumap [ options ]`
: Delete local to remote user mappings for a local/remote domain pair. See the `delumap` manual page for an explanation of the available options and for examples.

`delusr (delu) [ options ]`
: Delete remote usernames and passwords from the remote user and password tables of a remote domain. See the `delusr` reference page for an explanation of the available options and for examples.

`dsdmlog (dsdlg) -d` *`local_domain_name`* `[ -y ]`
: Destroy the domain transaction log for the named local domain on the current machine (that is, the machine where `dmadmin` is running). An error is returned if a `DMTLOG` is not defined for this local domain, if the local domain is active, or if outstanding transaction records exist in the log. The term outstanding transactions means that a global transaction has been committed but an end-of-transaction has not yet been written. This command prompts for confirmation before proceeding unless the `-y` option is specified.

`echo (e) [{off | on}]`
: Echo input command lines when set to `on`. If no option is given, then the current setting is toggled, and the new setting is printed. The initial setting is `off`.

`forgettrans (ft) -d` *`local_domain_name`* `[ -t` *`tran_id`*`]`
: Forget one or all heuristic log records for the named local domain. If the transaction identifier *`tran_id`* is specified, then only the heuristic log record for that transaction will be forgotten. The transaction identifier *`tran_id`* can be obtained from the `printtrans` command or from the `ULOG` file.

`help (h) [`*`command`*`]`
: Print help messages. If *`command`* is specified, the abbreviation, arguments, and description for that command are printed. Omitting all arguments causes the syntax of all commands to be displayed.

`indmlog (indlg) -d` *`local_domain_name`* `[ -y ]`
: Reinitialize the domain transaction log for the named local domain on the current machine (that is, the machine where `dmadmin` is running). An error is returned if a `DMTLOG` is not defined for this local domain, if the local domain is active, or if outstanding transaction records exist in the log. The term outstanding transactions means that a global transaction has been committed but an end-of-transaction has not yet been written. The command prompts for confirmation before proceeding unless the `-y` option is specified.

`modusr (modu) [ options ]`
: Change remote passwords in the password tables of a remote domain.

`paginate (page) [{off | on}]`

Paginate output. If no option is given, then the current setting will be toggled, and the new setting is printed. The initial setting is on, unless either standard input or standard output is a non-tty device. Pagination may only be turned on when both standard input and standard output are tty devices. The shell environment variable PAGER may be used to override the default command used for paging output. The default paging command is indigenous to the native operating system environment.

`passwd (passwd) [ -r ] local_domain_name remote_domain_name`

Prompts the administrator for new passwords for the specified local and remote domains. The `-r` option specifies that existing passwords and new passwords should be encrypted using a new key generated by the system. The password is truncated after at most eight characters. The TMA OSI TP gateway must be shut down and restarted for new passwords to take effect.

`printdomain (pd) -d local_domain_name`

Print information about the named local domain. Information printed includes connected remote domains, global information shared by the gateway processes, and additional information that is dependent on the domain type instantiation.

`printstats (stats) -d local_domain_name`

Print statistical and performance information gathered by the named local domain. The information printed is dependent on the domain gateway type.

`printtrans (pt) -d local_domain_name`

Print transaction information for the named local domain.

`quit (q)`

Terminate the session.

`resume (res) -d local_domain_name [{ -all | service}]`

Resume processing of the specified service or for all remote services handled by the named local domain.

`stats (stats) -d local_domain_name [{ off | on | reset }]`

Activate (*on*), deactivate (*off*), or reset (*reset*) statistics gathering for the named local domain. If no option is given, then the current setting will be toggled between the values *on* and *off*, and the new setting will be printed. The initial setting is `off`.

`suspend (susp) -d local_domain_name [{ -all | service}]`

Suspend one or all remote services for the named local domain.

`unadvertise (unadv) -d local_domain_name [{ -all | service}]`

Unadvertise one or all remote services for the named local domain.

`verbose (v) [{off | on}]`

Produce output in verbose mode. If no option is given, then the current setting will be toggled, and the new setting is printed. The initial setting is `off`.

`! shellcommand`

Escape to shell and execute *`shellcommand`*.

`!!`

Repeat previous shell command.

`# [text]`

Lines beginning with "#" are comment lines and are ignored.

`<CR>`

Repeat the last command.

## CONFIGURATION MODE COMMANDS

The `dmadmin` command enters configuration mode when executed with the `-c` option or when the `config` subcommand is used. In this mode, `dmadmin` allows run-time updates to the `BDMCONFIG` file. `dmadmin` manages a buffer that contains input field values to be added or retrieved, and displays output field values and status after each operation completes. The user can update the input buffer using any available text editor.

`dmadmin` first prompts for the desired section followed by a prompt for the desired operation.

The prompt for the section is as follows:

```
Sections:
       1) RESOURCES          2) LOCAL_DOMAINS
       3) REMOTE_DOMAINS     4) LOCAL_SERVICES
       5) REMOTE_SERVICES    6) ROUTING
       7) ACCESS_CONTROL     8) PASSWORDS
       9) TDOMAINS           10)OSITPS
       11)SNADOMS            12)LOCAL_REMOTE_USER
       13)REMOTE_USERS       14)SNACRMS
       15)SNASTACKS          16)SNALINKS
       18)TOPEND             19)OSITPX
       q) QUIT
Enter Section [1]
```

The number of the default section appears in square brackets at the end of the prompt. You can accept the default by pressing RETURN or ENTER. To select another section enter its number, then press RETURN or ENTER.

dmadmin then prompts for the desired operation.

```
Operations:
       1) FIRST              2) NEXT
       3) RETRIEVE           4) ADD
       5) UPDATE             6) DELETE
       7) NEW_SECTION        8) QUIT
Enter Operation [1]:
```

The number of the default operation is printed in square brackets at the end of the prompt. Pressing **RETURN** or **ENTER** selects this option. To select another operation enter its number, then press **RETURN** or **ENTER**.

The currently supported operations are

**1 FIRST**

Retrieve the first record from the specified section. No key fields are needed (they are ignored if in the input buffer).

**2 NEXT**

Retrieve the next record from the specified section, based on the key fields in the input buffer.

**3 RETRIEVE**

Retrieve the indicated record from the specified section by key field(s) (see the following fields description).

**4 ADD**

Add the indicated record in the specified section. Any fields not specified (unless required) take their default values as specified in dmconfig. The current value for all fields is returned in the output buffer. This operation can only be done by the System/T administrator.

**5 UPDATE**

Update the record specified in the input buffer in the selected section. Any fields not specified in the input buffer remain unchanged. The current value for all fields is returned in the input buffer. This operation can only be done by the System/T administrator.

**6 DELETE**

Delete the record specified in the input buffer from the selected section. This operation can only be done by the System/T administrator.

**7 NEW SECTION**

Clear the input buffer (all fields are deleted). After this operation, dmadmin immediately prompts for the section again.

**8 QUIT**

Exit the program gracefully (dmadmin is terminated). A value of q for any prompt also exits the program.

For configuration operations, the effective user identifier must match the System/T administrator user identifier UID for the machine on which this program is executed. When a record is updated or added, all default values and validations used by udmloadcf are enforced.

dmadmin then prompts whether or not to edit the input buffer.

```
Enter editor to add/modify fields [n]?
```

Entering a value of y will put the input buffer into a temporary file and execute the text editor. The environment variable EDITOR is used to determine which editor to be used; the default is "ed". The input format is in field name/field value pairs and is described in the CONFIGURATION INPUT FORMAT section that follows. The field names associated with each DMCONFIG section are listed in tables in the subsections that follow. The semantics of the fields and associated ranges, default values, restrictions, etc., are described in dmconfig. In most cases, the field name is the same as the KEYWORD in the DMCONFIG file, prefixed with "TA_DM". When the user completes editing the input buffer, dmadmin reads it. If more than one line occurs for a particular field name, the first occurrence is used and other occurrences are ignored. If any errors occur, a syntax error will be printed and dmadmin prompts whether or not to correct the problem.

```
Enter editor to correct?
```

If the problem is not corrected (response n), then the input buffer will contain no fields. Otherwise, the editor is executed again.

Finally, dmadmin asks if the operation should be done.

```
Perform operation [y]?
```

When the operation completes, dmadmin prints the return value as in

```
Return value TAOK
```

followed by the output buffer fields. The process then begins again with a prompt for the section. All output buffer fields are available in the input buffer unless the buffer is cleared.

Entering break at any time restarts the interaction at the prompt for the section.

When "QUIT" is selected, dmadmin prompts for authorization to create a backup ASCII version of the configuration:

```
Unload BDMCONFIG file into ASCII backup [y]?
```

If a backup is selected, dmadmin prompts for the file name.

```
Backup filename [DMCONFIG]?
```

On success, dmadmin indicates that a backup was created, otherwise an error is printed.

## CONFIGURATION INPUT FORMAT

Input packets consist of lines formatted as follows:

```
fldname<tabs>fldval
```

The field name is separated from the field value by one or more tabs (or spaces).

Lengthy field values can be continued on the next line by having the continuation line begin with one or more tabs (which are dropped when read back into dmadmin).

Empty lines consisting of a single newline character are ignored.

To enter an unprintable character in the field value or to start a field value with a tab, use a backslash followed by the two-character hexadecimal representation of the desired character. A space, for example, can be entered in the input data as \20. A backslash can be entered using two backslash characters. dmadmin recognizes all input in this format, but its greatest usefulness is for non-printing characters.

## CONFIGURATION LIMITATIONS

The following are general limitations of the dynamic domain re-configuration capability:

- Values for key fields (as indicated in the following sections) may not be modified. Key fields can be modified, when the system is down, by reloading the configuration file.
- Dynamic deletions cannot be applied when local domains are active (the corresponding gateway group is running).

## RESTRICTIONS FOR CONFIGURATION FIELD IDENTIFIERS/UPDATES

The following sections describe, for each DMCONFIG section, what the field identifiers are for each DMCONFIG field, what the field type of the identifier is, and when the field can be updated. All applicable field values are returned with the retrieval operations. Fields that are allowed and/or required for adding a record are described in dmconfig. The following fields indicated as *key* are

key fields that are used to uniquely identify a record within section. These key fields are required to be in the input buffer when updates are done and are not allowed to be updated dynamically. The `Update` column indicates when a field can be updated. The possible values are

**Yes**
Can be updated at any time.

**NoGW**
Cannot be updated dynamically while the gateway group representing the local domain is running.

**No**
Cannot be updated dynamically while at least one gateway group is running.

## CONFIGURING THE DM_LOCAL_DOMAINS SECTION

The following table lists the fields in the `DM_LOCAL_DOMAINS` section.

**Table A-1 DM_LOCAL_DOMAINS SECTION**

| Field Identifier | Field Type | Update | Notes |
|---|---|---|---|
| TA_AUDITLOG | string | Yes | |
| TA_BLOCKTIME | numeric | Yes | |
| TA_DMTLOGDEV | string | NoGW | |
| TA_DMTLOGNAME | string | NoGW | |
| TA_DMTLOGSIZE | numeric | NoGW | |
| TA_DOMAINID | string | NoGW | |
| TA_GWGRP | string | NoGW | |
| TA_LDOM | string | NoGW | key |
| TA_MAXDATALEN | numeric | Yes | |
| TA_MAXRDOM | numeric | Yes | |
| TA_MAXRDTRAN | numeric | NoGW | |
| TA_MAXTRAN | numeric | NoGW | |

**Table A-1 DM_LOCAL_DOMAINS SECTION**

| Field Identifier | Field Type | Update | Notes |
|---|---|---|---|
| TA_SECURITY | string | Yes | format: {NONE \| APP_PW \| DM_PW} |
| TA_TYPE | string | NoGW | format: {TDOMAIN \| OSITP \| OSITPX \| SNA} |

Refer to the "DM_LOCAL_DOMAINS Section" in "Understanding the DMCONFIG File" for information about some of these fields information.

## CONFIGURING THE DM_REMOTE_DOMAINS SECTION

The following table lists the fields in the DM_REMOTE_DOMAINS section.

**Table A-2 DM_REMOTE_DOMAINS SECTION**

| Field Identifier | Field Type | Update | Notes |
|---|---|---|---|
| TA_DOMAINID | string | No | |
| TA_RDOM | string | No | key |
| TA_TYPE | string | No | format: {TDOMAIN \| OSITP \| OSITPX \| SNA} |

Refer to the "DM_REMOTE_DOMAINS Section" in "Understanding the DMCONFIG File" for related information about some of these fields.

## CONFIGURING THE DM_OSITPX SECTION

The DM_OSITPX section contains the network addressing parameters required by OSITP type domains. The following lists the fields in this section:

**Table A-3 DM_OSITPX SECTION**

| Field Identifier | Field Type | Update | Notes |
|---|---|---|---|
| TA_AET | string | No/NoGW | |
| TA_DNSRESOLUTION | string | No/NoGW | |
| TA_EXTENSIONS | string | No/NoGW | |

**Table A-3 DM_OSITPX SECTION**

| Field Identifier | Field Type | Update | Notes |
|---|---|---|---|
| TA_LDOM or TA_RDOM | string | No/NoGW | key |
| TA_MAXENCRYPTBITS | string | No/NoGW | |
| TA_MINENCRYPTBITS | string | No/NoGW | |
| TA_MULTIPLEXING | string | No/NoGW | |
| TA_NWADDR | string | No/NoGW | |
| TA_OPTIONS | string | No/NoGW | |
| TA_OSITPX | string | No/NoGW | |
| TA_PSEL | string | No/NoGW | |
| TA_SSEL | string | No/NoGW | |
| TA_TAILORPATH | string | No/NoGW | |
| TA_TSEL | string | No/NoGW | |
| TA_XATMIENCODING | string | No/NoGW | |

If the domain identifier (TA_LDOM) is a local domain identifier, then the other fields in this table can be updated if the gateway group representing that local domain is not running.

Refer to the "DM_OSITPX Section" in "Understanding the DMCONFIG File" for related information about some of these fields.

## CONFIGURING THE DM_LOCAL_SERVICES SECTION

The following table lists the fields in the DM_LOCAL_SERVICES section.

**Table A-4 DM_LOCAL_SERVICES SECTION**

| Field Identifier | Field Type | Update | Notes |
|---|---|---|---|
| TA_AUTOPREPARE | string | Yes | |
| TA_BLOCKTIME | string | No | |
| TA_BUFSTYPE | string | Yes | INBUF SUBTYPE |

**Table A-4 DM_LOCAL_SERVICES SECTION**

| Field Identifier | Field Type | Update | Notes |
|---|---|---|---|
| TA_BUFTYPE | string | Yes | INBUF |
| TA_CONV | string | NoGW | format: {Y \| N} |
| TA_COUPLING | string | Yes | |
| TA_INRECSTYPE | string | Yes | INRECTYPE SUBTYPE |
| TA_INRECTYPE | string | Yes | |
| TA_LDOM | string | No | key |
| TA_LOAD | string | No | |
| TA_OBUFSTYPE | string | Yes | OUTBUF SUBTYPE |
| TA_OBUFTYPE | string | Yes | OUTBUF |
| TA_OUTRECSTYPE | string | Yes | OUTRECTYPE SUBTYPE |
| TA_PRIO | string | No | |
| TA_RDOM | string | No | key |
| TA_REMTPSUT | string | No/NoGW | |
| TA_RNAME | string | Yes | |
| TA_ROUTINGNAME | string | No | |
| TA_SERVICENAME | string | No | key |
| TA_TPSUTTYPE | string | No/NoGW | |
| TA_TRANTIME | string | Yes | |

Refer to the "DM_LOCAL_SERVICES Section" in "Understanding the DMCONFIG File" for related information about some of these fields.

## CONFIGURING THE DM_REMOTE_SERVICES SECTION

The following table lists the fields in the DM_REMOTE_SERVICES section.

**Table A-5 DM_REMOTE_SERVICES SECTION**

| Field Identifier | Field Type | Update | Notes |
|---|---|---|---|
| TA_BUFSTYPE | string | Yes | INBUF SUBTYPE |
| TA_BUFTYPE | string | Yes | INBUF |
| TA_CONV | string | NoGW | format: { Y \| N } |
| TA_INRECSTYPE | string | Yes | INRECTYPE SUBTYPE |
| TA_LDOM | string | No | key |
| TA_OBUFSTYPE | string | Yes | OUTBUF SUBTYPE |
| TA_OBUFTYPE | string | Yes | OUTBUF |
| TA_OUTRECSTYPE | string | Yes | OUTRECTYPE SUBTYPE |
| TA_RDOM | string | No | key |
| TA_RNAME | string | Yes | |
| TA_ROUTINGNAME | string | Yes | |
| TA_SERVICENAME | string | No | key |
| TA_TRANTIME | numeric | Yes | |

Refer to the "DM_REMOTE_SERVICES Section" in "Understanding the DMCONFIG File" for related information about some of these fields.

## CONFIGURING THE DM_ROUTING SECTION

The following table lists the fields in the DM_ROUTING section.

**Table A-6 DM_ROUTING SECTION**

| Field Identifier | Field Type | Update | Notes |
|---|---|---|---|
| TA_ROUTINGNAME | string | No | key |
| TA_FIELD | string | Yes | |
| TA_RANGE | string | Yes | |
| TA_BUFTYPE | string | Yes | |

Refer to the "DM_ROUTING Section" in "Understanding the DMCONFIG File" for related information.

## CONFIGURING THE DM_ACCESS_CONTROL SECTION

The following table lists the fields in the DM_ACCESS_CONTROL section.

**Table A-7 DM_ACCESS_CONTROL SECTION**

| Field Identifier | Field Type | Update | Notes |
|---|---|---|---|
| TA_ACLNAME | string | No | key |
| TA_RDOM | string | Yes | |

Refer to the "DM_ACCESS_CONTROL Section" in "Understanding the DMCONFIG File" for related information.

## CONFIGURING THE DM_PASSWORDS SECTION

The following table lists the fields in the DM_PASSWORDS section.

**Table A-8 DM_PASSWORDS SECTION**

| Field Identifier | Field Type | Update | Notes |
|---|---|---|---|
| TA_LDOM | string | No | key |
| TA_RDOM | string | No | key |

**Table A-8 DM_PASSWORDS SECTION**

| Field Identifier | Field Type | Update | Notes |
|---|---|---|---|
| TA_LPWD | string | Yes | format: { Y \| N \| U } |
| TA_RPWD | string | Yes | format: { Y \| N \| U } |

The TA_LPWD and TA_RPWD show the existence of a defined password for the local and/or the remote domain. Passwords are not displayed. If an UPDATE operation is selected, the value of the corresponding field must be set to U. The program will then prompt with echo turned off for the corresponding passwords.

## DIAGNOSTICS IN CONFIGURATION MODE

dmadmin fails if it cannot allocate an FML typed buffer, if it cannot determine the /etc/passwd entry for the user, or if it cannot reset the environment variables FIELDTBLS or FLDTBLDIR.

The return value printed by dmadmin after each operation completes indicates the status of the requested operation. There are three classes of return values.

The following return values indicate a problem with permissions or an OSITP communications error. They indicate that the operation did not complete successfully.

**[TAEPERM]**
The calling process specified an ADD, UPDATE, or DELETE operation but it is not running as the System/T administrator. Update operations must be run by the administrator (that is, the user specified in the UID attribute of the RESOURCES section of the TUXCONFIG file).

**[TAESYSTEM]**
An OSITP error has occurred. The exact nature of the error is written to userlog.

**[TAEOS]**
An operating system error has occurred.

**[TAETIME]**
A blocking timeout occurred. The input buffer is not updated so no information is returned for retrieval operations. The status of update operations can be checked by doing a retrieval on the record that was being updated.

The following return values indicate a problem in doing the operation itself and generally are semantic problems with the application data in the input buffer. The string field TA_STATUS will be set in the output buffer and will contain short text describing the problem. The string field

TA_BADFLDNAME will be set to the field name for the field containing the value that caused the problem (assuming the error can be attributed to a single field).

**[TAECONFIG]**
An error occurred while reading the BDMCONFIG file.

**[TAEDUPLICATE]**
The operation attempted to add a duplicate record.

**[TAEINCONSIS]**
A field value or set of field values are inconsistently specified.

**[TAENOTFOUND]**
The record specified for the operation was not found.

**[TAENOSPACE]**
The operation attempted to do an update but there was not enough space in the BDMCONFIG file.

**[TAERANGE]**
A field value is out of range or is invalid.

**[TAEREQUIRED]**
A field value is required but not present.

**[TAESIZE]**
A field value for a string field is too long.

**[TAEUPDATE]**
The operation attempted to do an update that is not allowed.

The following return values indicate that the operation was successful.

**[TAOK]**
The operation succeeded. No updates were done to the BDMCONFIG file.

**[TAUPDATED]**
The operation succeeded. Updates were made to the BDMCONFIG file.

When using dmunloadcf to print entries in the configuration, optional field values are not printed if they are not set (for strings) or 0 (for integers). These fields will always appear in the output buffer when using dmadmin. In this way, it makes it easier for the administrator to retrieve an entry and update a field that previously was not set. The entry will have the field name followed by a tab but no field value.

## CONFIGURATION EXAMPLE

In the following example, `dmadmin` is used to add a new remote domain. For illustration purposes, `ed` is used for the editor.

```
$EDITOR=vi dmadmin -c
dmadmin - Copyright (c) 1996-1999 BEA Systems, Inc.
Portions * Copyright 1986-1997 RSA Data Security, Inc.
All Rights Reserved.
Distributed under license by BEA Systems, Inc.
Tuxedo is a registered trademark.
Section:
         1) RESOURCES       2) LOCAL_DOMAINS
         3) REMOTE_DOMAINS  4) LOCAL_SERVICES
         5) REMOTE_SERVICES 6) ROUTING
         7) ACCESS_CONTROL  8) PASSWORDS
         9) TDOMAINS       10) OSITPS
        11) SNADOMS        12) LOCAL_REMOTE_USER
        13) REMOTE_USERS   14) SNACRMS
        15) SNASTACKS      16) SNALINKS
        18) TOPEND         19) OSITPX
        q) QUIT
Enter Section [1]:3

Operations:
        1) FIRST        2) NEXT
        3) RETRIEVE     4) ADD
        5) UPDATE       6) DELETE
        7) NEW_SECTION  8) QUIT
Enter Operation [1]:4
Enter editor to add/modify fields [n]? y
In VI...
a
TA_RDOM       B05
TA_DOMAINID   BA.BANK05
TA_TYPE       OSITPX
wq!
Perform operation [y]?
```

```
Return value TAUPDATED
Buffer contents:
TA_OPERATION    4
TA_CLASS        3
TA_OCCURS       1
TA_DMINPRIORITY 0
TA_STATUS       CMDGW_CAT:1761: INFO: Update completed successfully
TA_TYPE OSITPX
TA_DOMAINID     BA.BANK05
TA_RDOM B05
TA_CODEPAGE
TA_DMACLPOLICY  LOCAL
TA_DMCONNPRINCIPALNAME
TA_DMLOCALPRINCIPALNAME
TA_DMPRIORITY_TYPE      LOCAL_RELATIVE
TA_DMCREDENTIALPOLICY   LOCAL

Section:
         1) RESOURCES       2) LOCAL_DOMAINS
         3) REMOTE_DOMAINS  4) LOCAL_SERVICES
         5) REMOTE_SERVICES 6) ROUTING
         7) ACCESS_CONTROL  8) PASSWORDS
         9) TDOMAINS       10) OSITPS
        11) SNADOMS        12) LOCAL_REMOTE_USER
        13) REMOTE_USERS   14) SNACRMS
        15) SNASTACKS      16) SNALINKS
        18) TOPEND          19) OSITPX
        q) QUIT
Enter Section [3]:

Operations:
        1) FIRST        2) NEXT
        3) RETRIEVE     4) ADD
        5) UPDATE       6) DELETE
        7) NEW_SECTION  8) QUIT
Enter Operation [4]:7
Buffer cleared
Section:
```

```
         1) RESOURCES       2) LOCAL_DOMAINS
         3) REMOTE_DOMAINS  4) LOCAL_SERVICES
         5) REMOTE_SERVICES 6) ROUTING
         7) ACCESS_CONTROL  8) PASSWORDS
         9) TDOMAINS       10) OSITPS
        11) SNADOMS        12) LOCAL_REMOTE_USER
        13) REMOTE_USERS   14) SNACRMS
        15) SNASTACKS      16) SNALINKS
        18) TOPEND          19) OSITPX
        q) QUIT
Enter Section [3]:19

Operations:
        1) FIRST        2) NEXT
        3) RETRIEVE     4) ADD
        5) UPDATE       6) DELETE
        7) NEW_SECTION  8) QUIT
Enter Operation [7]:4
Enter editor to add/modify fields [n]? y
From VI editor...
a
TA_OSITPX         B05
TA_NWADDR         206.189.43.44
TA_AET            {1.3.9999.1},{1}
wq!

Perform operation [y]?

Return value TAUPDATED
Buffer contents:
TA_OPERATION    4
TA_CLASS        19
TA_OCCURS       1
TA_MULTIPLEXING 0
TA_MINENCRYPTBITS       0
TA_MAXENCRYPTBITS       0
TA_STATUS       CMDGW_CAT:1761: INFO: Update completed successfully
TA_NWADDR       206.189.43.44
```

```
TA_AET  {1.3.9999.1},{1}
TA_OSITPX       B05
TA_TSEL
TA_TAILORPATH
TA_PSEL
TA_SSEL
TA_EXTENSIONS
TA_DNSRESOLUTION        STARTUP
TA_OPTIONS
TA_XATMIENCODING        PRELIMINARY

Section:
         1) RESOURCES       2) LOCAL_DOMAINS
         3) REMOTE_DOMAINS  4) LOCAL_SERVICES
         5) REMOTE_SERVICES 6) ROUTING
         7) ACCESS_CONTROL  8) PASSWORDS
         9) TDOMAINS       10) OSITPS
        11) SNADOMS        12) LOCAL_REMOTE_USER
        13) REMOTE_USERS   14) SNACRMS
        15) SNASTACKS      16) SNALINKS
        18) TOPEND          19) OSITPX
        q) QUIT
Enter Section [19]:q
Unload BDMCONFIG file into ASCII backup [y]? y
Backup filename [DMCONFIG]?
Configuration backed up in DMCONFIG
q

The dmadmin program ends.
```

## SECURITY

If `dmadmin` is run with the application administrator's `UID`, it assumes a trusted user and Security is bypassed. If `dmadmin` is run with another user ID, and if the security option is enabled in the `TUXCONFIG` file, then the corresponding application password is required to start the `dmadmin` program. If standard input is a terminal, then `dmadmin` will prompt the user for the password with echo turned off. If standard input is not a terminal, the password is retrieved from the environment variable, `APP_PW`. If this environment variable is not specified and an application password is required, then `dmadmin` will fail to start.

When running with another user ID (other than the UID of the administrator) only a limited set of commands is available.

## ENVIRONMENT VARIABLES

`dmadmin` resets the `FIELDTBLS` and `FLDTBLDIR` environment variables to pick up the ${TUXDIR}/udataobj/dmadmin field table. Hence, the `TUXDIR` environment variable should be set correctly.

If the application requires security and the standard input to `dmadmin` is not from a terminal, then the `APP_PW` environment variable must be set to the corresponding application password.

The `TUXCONFIG` environment variable should be set to the pathname of the OSITP configuration file.

## GENERAL DIAGNOSTICS

If the `dmadmin` command is entered before the system has been booted, the following message is displayed:

```
No bulletin board exists. Only logging commands are available.
```

`dmadmin` then prompts for the corresponding commands.

If an incorrect application password is entered or is not available to a shell script through the environment, then a log message is generated, the following message is displayed, and the command terminates:

```
Invalid password entered.
```

## SEE ALSO

*Tuxedo /Domain Guide*

# GWADM

/Domain gateway administrative server.

## SYNOPSIS

```
GWADM SRVGRP = "identifier" SRVID = "number" REPLYQ = "N"
      CLOPT = "-A -- [-a {on | off}] [-s services]
      [-t {on | off}]"
```

## DESCRIPTION

The gateway administrative server GWADM is a Tuxedo-supplied server that provides administrative functions for a /Domain gateway group.

GWADM should be defined in the *SERVERS section of the UBBCONFIG file as a server running within a particular gateway group, that is, SRVGRP must be set to the corresponding GRPNAME tag specified in the *GROUPS section. The SVRID parameter is also required and its value must consider the maximum number of gateways allowed within the gateway group.

There should be only one instance of a GWADM per /Domain gateway group, and it should NOT be part of the MSSQ defined for the gateways associated with the group. Also, GWADM should have the REPLYQ attribute set to N.

The CLOPT option is a string of command line options that is passed to the GWADM when it is booted. This string has the following format:

```
CLOPT="-A -- <gateway group runtime parameters>"
```

The following runtime parameters are recognized for a gateway group

`-a {on | off}`

This option turns off or on the audit log feature for this local domain. The default is off. The dmadmin program can be used to change this setting while the gateway group is running (see dmadmin).

`-s services`

Specifies the remote *services* that should be initially offered by the domain gateway. The specifications for these services are found in the DMCONFIG file. For example, the specification

```
-s x,y,z
```

implies that the gateway should initially advertise remote services x, y, and z. Spaces are not allowed between commas and the -s option may appear several times.

`-t {on | off}`

This option turns off or on the statistics gathering feature for the local domain. The default is off. The dmadmin program can be used to change this setting while the gateway group is running (see dmadmin).

The GWADM server must be booted before the corresponding gateways.

## PORTABILITY

GWADM is supported on Tuxedo-supplied servers, using non-/WS operating systems.

## INTEROPERABILITY

The initial release of TMA OSI TP can only be installed on a node running Tuxedo Release 6.5 or Release 7.1.

## EXAMPLES

The following example illustrates the definition of the administrative server in the UBBCONFIG file.

```
#
*GROUPS
DMADMGRP  GRPNO=1
gwgrp    GRPNO=2
#
*SERVERS
DMADM SRVGRP="DMADMGRP" SRVID=1001 RESTART=Y GRACE=0
GWADM SRVGRP="gwgrp" SRVID=1002 RESTART=Y GRACE=0
      CLOPT="-A -- -a on -t on"
GWTDOMAIN SRVGRP="gwgrp" SRVID=1003
RESTART=Y MIN=1 MAX=1
```

## SEE ALSO

dmadmin, tmboot

dmconfig, DMADM, servopts, ubbconfig

*Tuxedo /Domain Guide*

*Tuxedo Administrator's Guide*

# Manually Upgrading to BEA Tuxedo Mainframe Adapter for OSI TP 9.1

This section covers the following topics for manually upgrading your `product`:

- Upgrading From eLink OSI TP Version 1.3

  If you are upgrading from eLink OSI TP 1.3 to BEA Tuxedo Mainframe Adapter for OSI TP version 9.1, you must modify your existing `DMCONFIG` file to create a `DMCONFIG` file compatible with Tuxedo 9.0 or 9.1.

  **Note:** It is recommended that you use the `osiadmin` utility to upgrade your `DMCONFIG` file, but you can perform this upgrade manually.

- Upgrading From eLink OSI TP Version 4.0

## Upgrading From eLink OSI TP Version 1.3

To manually upgrade your `DMCONFIG` file for TMA OSI TP Version 9.1, perform the following General Checklist steps. Each step is detailed in the following sections.

1. After installing BEA Tuxedo Mainframe Adapter for OSI TP 9.1, locate the original `DMCONFIG` file, referred to as `DMCONFIG.TXT`, and the `ositp configfile` you wish to upgrade and make backup copies of the files before making any modifications.
2. Modify the `DMCONFIG` file.

Following are the detailed steps for upgrading your `DMCONFIG` file:

# Step 1 - Backup Existing Configuration Files

It is recommended that you make a backup copy of your original DMCONFIG.TXT file and OSITP CONFIGFILE before making any changes or running any utilities. A backup helps prevent the loss of your current configuration and allows you to restore any information in the event there is a problem. Enter the following command from your command line or DOS prompt to copy your DMCONFIG.TXT file to DMCONFIG.bak:

```
copy DMCONFIG.TXT DMCONFIG.bak
```

# Step 2 - Modify Parameters in the DMCONFIG File

You must change the DM_OSITP section name to DM_OSITPX, and you must modify the TYPE field in the DM_LOCAL_DOMAINS and DM_REMOTE_DOMAINS sections to be TYPE=OSITPX. In order to use some of the newer features, you may optionally want to modify the DM_LOCAL_SERVICES and DM_REMOTE_SERVICES definitions as well. Changes to each of these sections consist of removing obsolete parameters, adding new parameters, and modifying values for existing parameters. You can modify entries in the DMCONFIG file using any text editor.

Following is a summary of the changes that need to be made to your configuration:

| Section | Parameter | Change |
|---|---|---|
| DM_OSITP | | Obsolete |
| DM_OSITPX | | Available in eLink OSI TP 4.x and TMA OSI TP 8.1 and 9.1 |
| DM_OSITPX | ACN | Obsolete |
| | AEID | Obsolete |
| | AEQ | Obsolete |
| | AET | Now required for each domain |
| | APID | Obsolete |
| | APT | Obsolete |

| Section | Parameter | Change |
|---|---|---|
| | DNS_RESOLUTION | Available in eLink OSI TP 4.x and TMA OSI TP 8.1 and 9.1 |
| | MAX_LISTENING_EP | Obsolete |
| | NWADDR | Now required for each domain |
| | NWDEVICE | Obsolete |
| | OPTIONS | Available in eLink OSI TP 4.x and TMA OSI TP 8.1 and 9.1 |
| | P_SEL | Available in eLink OSI TP 4.x and TMA OSI TP 8.1 and 9.1 |
| | PROFILE | Obsolete |
| | S_SEL | Available in eLink OSI TP 4.x and TMA OSI TP 8.1 and 9.1 |
| | T_SEL | Available in eLink OSI TP 4.x and TMA OSI TP 8.1 and 9.1 |
| | TAILOR_PATH | Available in eLink OSI TP 4.x and TMA OSI TP 8.1 and 9.1 |
| | URCH | Obsolete |
| DM_LOCAL_DOMAINS | SECURITY | Available in eLink OSI TP 4.x and TMA OSI TP 8.1 and 9.1 |
| | TYPE=OSITP | Obsolete |

| Section | Parameter | Change |
|---|---|---|
| | TYPE=OSITPX | Available in eLink OSI TP 4.x and TMA OSI TP 8.1 and 9.1 |
| DM_LOCAL_SERVICES | COUPLING | Available in eLink OSI TP 4.x and TMA OSI TP 8.1 and 9.1 |
| | INRECTYPE | Available in eLink OSI TP 4.x and TMA OSI TP 8.1 and 9.1 |
| | OUTRECTYPE | Available in eLink OSI TP 4.x and TMA OSI TP 8.1 and 9.1 |
| DM_REMOTE_DOMAINS | TYPE=OSITP | Obsolete |
| | TYPE=OSITPX | Available in eLink OSI TP 4.x and TMA OSI TP 8.1 and 9.1 |
| CODEPAGE | | Available in eLink OSI TP 4.1, 4.2 and TMA OSI TP 8.1 and 9.1 |
| DM_REMOTE_SERVICES | AUTOPREPARE | Available in eLink OSI TP 4.x and TMA OSI TP 8.1 and 9.1 |
| | INRECTYPE | Available in eLink OSI TP 4.x and TMA OSI TP 8.1 and 9.1 |
| | OUTRECTYPE | Available in eLink OSI TP 4.x and TMA OSI TP 8.1 and 9.1 |

| Section | Parameter | Change |
|---|---|---|
| | REM_TPSUT | Available in eLink OSI TP 4.x and TMA OSI TP 8.1 and 9.1 |
| | TPSUT_TYPE (must be specified **BEFORE** REM_TPSUT) | Available in eLink OSI TP 4.x and TMA OSI TP 8.1 and 9.1 |

Refer to Understanding the DMCONFIG File for more information about these new parameters.

# Upgrading From eLink OSI TP Version 4.0

There are no changes required to the eLink OSI TP 4.0 UDMCONFIG input file. Perform a dmloadcf of this file to create a BDMCONFIG binary file compatible with Tuxedo 9.0 or 9.1. If your ubbconfig file contains eLink OSI TP 4.0 Administrative Server UDMADM, which is obsolete for TMA OSI TP 9.1, delete it and perform a tmloadcf of the modified ubbconfig.

# Index

## D



## E



## F



## G

## I



## L



## M



## N



## O



## P

## R



## S



## T

## U



## W



## X