SOME TIBCO SOFTWARE EMBEDS OR BUNDLES OTHER TIBCO SOFTWARE. USE OF SUCH EMBEDDED OR BUNDLED TIBCO SOFTWARE IS SOLELY TO ENABLE THE FUNCTIONALITY (OR PROVIDE LIMITED ADD-ON FUNCTIONALITY) OF THE LICENSED TIBCO SOFTWARE. THE EMBEDDED OR BUNDLED SOFTWARE IS NOT LICENSED TO BE USED OR ACCESSED BY ANY OTHER TIBCO SOFTWARE OR FOR ANY OTHER PURPOSE.

USE OF TIBCO SOFTWARE AND THIS DOCUMENT IS SUBJECT TO THE TERMS AND CONDITIONS OF A LICENSE AGREEMENT FOUND IN EITHER A SEPARATELY EXECUTED SOFTWARE LICENSE AGREEMENT, OR, IF THERE IS NO SUCH SEPARATE AGREEMENT, THE CLICKWRAP END USER LICENSE AGREEMENT WHICH IS DISPLAYED DURING DOWNLOAD OR INSTALLATION OF THE SOFTWARE (AND WHICH IS DUPLICATED IN THE LICENSE FILE) OR IF THERE IS NO SUCH SOFTWARE LICENSE AGREEMENT OR CLICKWRAP END USER LICENSE AGREEMENT, THE LICENSE(S) LOCATED IN THE “LICENSE” FILE(S) OF THE SOFTWARE. USE OF THIS DOCUMENT IS SUBJECT TO THOSE TERMS AND CONDITIONS, AND YOUR USE HEREOF SHALL CONSTITUTE ACCEPTANCE OF AND AN AGREEMENT TO BE BOUND BY THE SAME.

This document contains confidential information that is subject to U.S. and international copyright laws and treaties. No part of this document may be reproduced in any form without the written authorization of TIBCO Software Inc.

TIBCO and Two-Second Advantage are either registered trademarks or trademarks of TIBCO Software Inc. in the United States and/or other countries.

Enterprise Java Beans (EJB), Java Platform Enterprise Edition (Java EE), Java 2 Platform Enterprise Edition (J2EE), and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle Corporation in the U.S. and other countries.

All other product and company names and marks mentioned in this document are the property of their respective owners and are mentioned for identification purposes only.

THIS SOFTWARE MAY BE AVAILABLE ON MULTIPLE OPERATING SYSTEMS. HOWEVER, NOT ALL OPERATING SYSTEM PLATFORMS FOR A SPECIFIC SOFTWARE VERSION ARE RELEASED AT THE SAME TIME. SEE THEREADME FILE FOR THE AVAILABILITY OF THIS SOFTWARE VERSION ON A SPECIFIC OPERATING SYSTEM PLATFORM.

THIS DOCUMENT IS PROVIDED “AS IS” WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT.

THIS DOCUMENT COULD INCLUDE TECHNICAL INACCURACIES OR TYPOGRAPHICAL ERRORS. CHANGES ARE PERIODICALLY ADDED TO THE INFORMATION HEREIN; THESE CHANGES WILL BE INCORPORATED IN NEW EDITIONS OF THIS DOCUMENT. TIBCO SOFTWARE INC. MAY MAKE IMPROVEMENTS AND/OR CHANGES IN THE PRODUCT(S) AND/OR THE PROGRAM(S) DESCRIBED IN THIS DOCUMENT AT ANY TIME.

THE CONTENTS OF THIS DOCUMENT MAY BE MODIFIED AND/OR QUALIFIED, DIRECTLY OR INDIRECTLY, BY OTHER DOCUMENTATION WHICH ACCOMPANIES THIS SOFTWARE, INCLUDING BUT NOT LIMITED TO ANY RELEASE NOTES AND “READ ME” FILES.

Copyright © 2010-2015 TIBCO Software Inc. ALL RIGHTS RESERVED.

TIBCO Software Inc. Confidential Information
## Contents

**Figures** ................................................................. 7  
**TIBCO Documentation and Support Services** ........................................ 8  
**Overview** ............................................................... 9  
- Plan a TIBCO MDM Project ........................................ 9  
- Multitiered versus Single Enterprise Tenancy ................. 9  
  - Multitiered and Single Enterprise Design Considerations ... 9  
  - Multitiered and Single Enterprise Comparison .............. 10  
**Installation and Configuration** ..................................... 11  
- Installation Choices ............................................... 11  
  - Single Node Installations ..................................... 11  
  - Highly Available or Fault Tolerant (HAFT) Installations .... 11  
  - TIBCO MDM Instance Configuration as Seeders in HAFT Installations .... 11  
- Subnet Configuration in HAFT Installations .................... 12  
- Considerations Sizing in HAFT Installations ................. 12  
- EMS Configuration in HAFT Installations ...................... 12  
- Security Considerations .......................................... 12  
- Value for Logging .................................................. 13  
- ActiveSpaces and TIBCO MDM .................................. 13  
- Seeder and Leech Modes ......................................... 13  
- Virtualized Environments and ActiveSpaces .................. 13  
- Considerations for Sizing in ActiveSpaces .................... 13  
- ActiveSpaces Configuration ..................................... 14  
- Optimize CPU Utilization ....................................... 14  
**Database Management** .............................................. 15  
- Table Spaces ....................................................... 15  
- Database Performance ............................................ 15  
- Database Large Object (LOB) management .................... 15  
- Miscellaneous Factors for Database Performance ............. 16  
**JMS Management** .................................................... 16  
- EMS ................................................................. 16  
- WebSphere MQ ..................................................... 16  
**Caches** ................................................................. 16  
- Sequence Numbers and Caches ................................ 16  
- Objects and Caches .............................................. 16  
- Management of Multiple TIBCO MDM Instances with Caches .... 16  
- Cache Configuration ............................................ 17  
**Security** ................................................................. 17  
- Managing Users and Roles ..................................... 17
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Work Supervisor Role</td>
<td>17</td>
</tr>
<tr>
<td>Support Engineer Role</td>
<td>18</td>
</tr>
<tr>
<td>Security Models</td>
<td>18</td>
</tr>
<tr>
<td>Lightweight Directory Access Protocol (LDAP) Integration</td>
<td>18</td>
</tr>
<tr>
<td>Data Encryption</td>
<td>18</td>
</tr>
<tr>
<td>Security Auditing</td>
<td>18</td>
</tr>
<tr>
<td>Analytics</td>
<td>18</td>
</tr>
<tr>
<td>Synchronization</td>
<td>19</td>
</tr>
<tr>
<td>Rulebases</td>
<td>19</td>
</tr>
<tr>
<td>Validation Rules</td>
<td>19</td>
</tr>
<tr>
<td>Initialization Rulebase</td>
<td>19</td>
</tr>
<tr>
<td>Use Enumerations with Rulebases</td>
<td>20</td>
</tr>
<tr>
<td>Security and Rulebases</td>
<td>20</td>
</tr>
<tr>
<td>Optimize Performance with Rulebases</td>
<td>20</td>
</tr>
<tr>
<td>Use Lookups with Rulebases</td>
<td>20</td>
</tr>
<tr>
<td>Indexes and Rulebases</td>
<td>20</td>
</tr>
<tr>
<td>Use Drop-Down Lists with Rulebases</td>
<td>20</td>
</tr>
<tr>
<td>Propagation and Rulebases</td>
<td>21</td>
</tr>
<tr>
<td>Constraints and Rulebases</td>
<td>21</td>
</tr>
</tbody>
</table>

**Design Considerations**                                           22

Repositories                                                            22

- Design of Data Models for Repositories                               22
- Design of Data Models for Repositories Using the Out-of-the-Box UI  23
- Attribute Groups                                                     23
- Identifying Related Entities and Attributes for a Repository         23
- Uses of TIBCO MDM Relationship Table in Repositories                24
- Sparse Repositories                                                  24

Record Relationships                                                   24

- Identify the Record Relationships during the Design Phase            25
- Manage the Cardinality of Record Relationships                       25

Management of Hierarchies                                              25

- Softlinks                                                             27
- Effective Dates                                                      27
- Data Source Identification and Design                                27
- Conventions of Naming                                                28
- User-defined Table and Column Names                                  28

**Import**                                                               30

- Initial Loads                                                        30
- Current Data Acceptance                                              30
- Initial Load with Cleansing                                          30
<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input Maps</td>
<td>31</td>
</tr>
<tr>
<td>Importing Nested Data Mapped to Multiple Repositories</td>
<td>31</td>
</tr>
<tr>
<td>Avoid Failures During Import</td>
<td>31</td>
</tr>
<tr>
<td>Import of Large Batches</td>
<td>32</td>
</tr>
<tr>
<td>Changes in the Input Map During Import</td>
<td>32</td>
</tr>
<tr>
<td>Imports in the Approval Stage</td>
<td>32</td>
</tr>
<tr>
<td>Use of CONTAINS Attribute for Imports</td>
<td>32</td>
</tr>
<tr>
<td>Date Loading</td>
<td>32</td>
</tr>
<tr>
<td>Use of SQL-Based Datasources</td>
<td>32</td>
</tr>
<tr>
<td>Import of Control Switches</td>
<td>33</td>
</tr>
<tr>
<td>Creation of Indexes for Data Sources for Database Loader</td>
<td>33</td>
</tr>
<tr>
<td><strong>Web Services</strong></td>
<td>34</td>
</tr>
<tr>
<td>Optimization of Performance with Web Services</td>
<td>34</td>
</tr>
<tr>
<td>Single Sign-On and Web Services</td>
<td>34</td>
</tr>
<tr>
<td>Use of Cache with Web Services</td>
<td>34</td>
</tr>
<tr>
<td>Concurrent Web Services</td>
<td>34</td>
</tr>
<tr>
<td>Synchronous Web Services</td>
<td>34</td>
</tr>
<tr>
<td>Workflows and Synchronous Web Services</td>
<td>35</td>
</tr>
<tr>
<td><strong>Workflows</strong></td>
<td>36</td>
</tr>
<tr>
<td>Customization of Workflows</td>
<td>36</td>
</tr>
<tr>
<td>Use of Java with Customized Workflows</td>
<td>36</td>
</tr>
<tr>
<td>Work with Transitions in Customized Workflows</td>
<td>36</td>
</tr>
<tr>
<td>Work with Caches in Customized Workflows</td>
<td>36</td>
</tr>
<tr>
<td>Use of Rules in Customized Workflows</td>
<td>36</td>
</tr>
<tr>
<td>Performance Considerations in Customized Workflows</td>
<td>37</td>
</tr>
<tr>
<td>Error Handling</td>
<td>37</td>
</tr>
<tr>
<td>Activities Tuning</td>
<td>37</td>
</tr>
<tr>
<td>Subflows</td>
<td>37</td>
</tr>
<tr>
<td>Synchronous and Asynchronous Subflows</td>
<td>37</td>
</tr>
<tr>
<td>Error Handling in Subflows</td>
<td>38</td>
</tr>
<tr>
<td>Spawned Workflows</td>
<td>38</td>
</tr>
<tr>
<td>Best Practices for Common Workflows</td>
<td>38</td>
</tr>
<tr>
<td>Best Practices for In-memory Workflows</td>
<td>38</td>
</tr>
<tr>
<td><strong>Deployment</strong></td>
<td>39</td>
</tr>
<tr>
<td>Task Prioritization through Isolation</td>
<td>39</td>
</tr>
<tr>
<td>TIBCO MDM Memory Utilization</td>
<td>39</td>
</tr>
<tr>
<td>Multiple TIBCO MDM Instances</td>
<td>40</td>
</tr>
<tr>
<td>Failover</td>
<td>40</td>
</tr>
<tr>
<td>Capacity Planning</td>
<td>40</td>
</tr>
<tr>
<td>Use of TIBCO MDM Studio</td>
<td>40</td>
</tr>
</tbody>
</table>
Figures

Message Prioritization ........................................................................................................... 39
TIBCO Documentation and Support Services

Documentation for this and other TIBCO products is available on the TIBCO Documentation site:
https://docs.tibco.com

Documentation on the TIBCO Documentation site is updated more frequently than any documentation that might be included with the product. To ensure that you are accessing the latest available help topics, please visit us at https://docs.tibco.com.

Product-Specific Documentation

Documentation for TIBCO products is not bundled with the software. Instead, it is available on the TIBCO Documentation site. To directly access documentation for this product, double-click the following file:
TIBCO_HOME/release_notes/TIB_mdm_version_docinfo.html

where TIBCO_HOME is the top-level directory in which TIBCO products are installed. On Windows, the default TIBCO_HOME is C:\tibco. On UNIX systems, the default TIBCO_HOME is /opt/tibco.

The following documents for this product can be found on the TIBCO Documentation site:

- TIBCO MDM Release Notes
- TIBCO MDM Installation and Configuration Guide
- TIBCO MDM User’s Guide
- TIBCO MDM System Administration
- TIBCO MDM Customization Guide
- TIBCO MDM Workflow Reference
- TIBCO MDM Web Services Guide
- JAVA API Reference
- TIBCO MDM Best Practices Guide
- TIBCO MDM Performance Tuning Guide

How to Contact TIBCO Support

For comments or problems with this manual or the software it addresses, contact TIBCO Support:

- For an overview of TIBCO Support, and information about getting started with TIBCO Support, visit this site:
  http://www.tibco.com/services/support

- If you already have a valid maintenance or support contract, visit this site:
  https://support.tibco.com
  Entry to this site requires a user name and password. If you do not have a user name, you can request one.

How to Join TIBCOMmunity

TIBCOMmunity is an online destination for TIBCO customers, partners, and resident experts. It is a place to share and access the collective experience of the TIBCO community. TIBCOMmunity offers forums, blogs, and access to a variety of resources. To register, go to:
https://www.tibcommunity.com
Overview

TIBCO MDM best practices based on contributions from TIBCO MDM users who develop the software and implement it in a variety of TIBCO MDM projects.

Some best practices may contradict others, because different targeted audiences may have mutually exclusive goals for the usage of the software.

Apply your judgement and experience with TIBCO MDM to determine whether a particular best practice applies to your environment.

Plan a TIBCO MDM Project

TIBCO MDM should be implemented in a phased manner to suit the requirements of your organization.

A so-called big-bang approach delays completion of the implementation and the realization of return-on-investment (ROI). If you use the big-bang approach, it can take years to achieve a full implementation of all requirements across all business functions.

Instead, you should start with a smaller project with phased implementation cycles and define goals and ROI for each phase. Each project or phase can provide ROI to your business. With a phased approach the implementation team can design incrementally, make corrections in subsequent phases, and better understand TIBCO MDM architecture’s quirks and best practices. For a successful TIBCO MDM project, you must train the implementation staff on the tools and methodology.

Multienterprise versus Single Enterprise Tenancy

An enterprise (also referred to as a company) is a logical unit that has almost complete data isolation (some global objects, such as global partners and data sources are shared).

With TIBCO MDM, you can manage multiple enterprises in the same instance. Managing one enterprise in an instance is called single enterprise tenancy. Managing multiple enterprises in an instance is called multienterprise tenancy.

Multienterprise and Single Enterprise Design Considerations

You must decide early in the design process whether to use TIBCO MDM with multi-enterprise or single-enterprise tenancy.

You should take several factors into consideration when making this decision. For example, multienterprise tenancy is a good design choice if you use all the enterprises similarly and you want a single point of operational control.

Single enterprise tenancy is a good choice if you want physically separated enterprises to which you can separately assign the required resources. This choice, however, comes with overhead costs: the total resources needed to maintain multiple single enterprise tenancies is significantly higher than those required for a multienterprise tenancy.

Before deciding on a multienterprise or single enterprise tenancy, consider whether your configuration is likely to change. Separating an enterprise from a multienterprise tenancy to create a single enterprise tenancy is possible. However, doing so is tedious and requires several manual steps and consultation with TIBCO Support. Merging an enterprise from a single enterprise tenancy to a multienterprise tenancy is relatively easier but does require some manual steps. These steps may include recreating the enterprise in multienterprise tenancy and importing the data and meta-data into the new environment.

In multienterprise tenancies, using a single data store you can use a single set of reporting tools for data analysis and aggregation across enterprises.

You should also consider performance characteristics. In most cases, TIBCO recommends a multienterprise tenancy if the enterprises are small (fewer than 50 million records). Enterprises with records exceeding 300 million records should in most cases be configured as single enterprise tenancies.
## Multienterprise and Single Enterprise Comparison

The following table compares the features in a multienterprise tenancy with the features in a single enterprise tenancy.

**Comparison between multi-enterprise and single-enterprise tenancy**

<table>
<thead>
<tr>
<th>Feature</th>
<th>Multienterprise</th>
<th>Single Enterprise</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Setup</strong></td>
<td>Multiple enterprises in one TIBCO MDM instance. Database, cache, JMS, among others, shared by all enterprises in instance. (You cannot assign quotas to each enterprise.)</td>
<td>Each enterprise is in a separate TIBCO MDM instance. Database, cache, JMS, among others, separate for each enterprise.</td>
</tr>
<tr>
<td><strong>Maintenance</strong></td>
<td>Software maintenance for all enterprises is managed together.</td>
<td>Software maintenance for each enterprise is managed separately.</td>
</tr>
<tr>
<td><strong>Configuration</strong></td>
<td>All configurations are shared, including single sign-on and role mapping, installed plug-ins and language packs, message prioritization, message listeners, file watchers, ConfigValues and configurations. Some of the customizations, such as look and feel, business process rules, workflows, and rulebases, can be enterprise-specific.</td>
<td>Configurations for each enterprise are separate.</td>
</tr>
<tr>
<td><strong>Data Isolation</strong></td>
<td>Data stored in shared database and cache. TIBCO MDM enforces logical separation between the enterprise data. (Global business partners and lookup data sources defined for the TIBCO MDM enterprise through rulebases are shared across all enterprises.)</td>
<td>All data is isolated.</td>
</tr>
<tr>
<td><strong>Performance</strong></td>
<td>Performance requirements of different enterprises can potentially conflict. A large enterprise will consume a large share of system resources.</td>
<td>Performance characteristics of each enterprise can be managed separately.</td>
</tr>
</tbody>
</table>
Installation and Configuration

Best practices are not limited to using MDM, it also involves installing MDM as well. Installation and configuration also has best practices that ensures effective implementation of MDM on your enterprise.

Installation Choices

The choice of installation decides how effective MDM is implemented on your enterprise. The installation choices are:

- Single node installations
- Highly Available or Fault Tolerant (HAFT) installations
- ActiveSpaces and TIBCO MDM
- Optimize CPU utilization

Single Node Installations

Single-node installations provide the simplest installation and permit quick starts. For most development purposes single-node installations are acceptable and the best choice.

For single-node installations, all the required software is installed on one machine, port numbers are unchanged, and TIBCO MDM is started in seeder mode with no external server.

If you are going to perform a single-node installation, consider using the embedded PostgreSQL database so you can use the simplified installer without separate database installs.

The default setup works well for small datasets (for example, under 50K). The default setup also works for larger datasets if you import or export data in chunks of 50K. For larger datasets, database tuning may be required.

User testing (such as User Acceptance Testing) generally requires a separate environment.

Highly Available or Fault Tolerant (HAFT) Installations

TIBCO recommends using highly-available or fault-tolerant (HAFT) installations in any production or preproduction environment.

To create an HAFT installation, configure several ActiveSpaces nodes across your production and nonproduction environments to support failover. As of TIBCO MDM 8.3.1 release version, you can also create the spaces externally and configure them as needed.

You need not set up the application servers into clustered mode to ensure they are highly available. TIBCO MDM nodes use a load balancer to distribute the load between multiple TIBCO MDM nodes. In most cases, using a software load balancer such as a webserver is sufficient.

Ensure that all TIBCO MDM instances use the same cache configuration file.

TIBCO MDM Instance Configuration as Seeders in HAFT Installations

If you configure TIBCO MDM instances as seeders, each instance connects to the same metaspace and forms a cluster.

Seeders can continue operating even after their connection to external ActiveSpaces nodes is lost. In addition, seeders reduce network traffic.
**Subnet Configuration in HAFT Installations**

Typically, production and preproduction boxes are installed on similar hardware in the same network space (subnet).

This type of setup can cause a configuration error to easily confuse the cache and messaging setups between these environments. To avoid such confusion, ensure that the:

- Configured ActiveSpaces nodes do not read from each other’s metaspaces by using separate metaspaces and modifying listen and discovery URLs for each environment. The suggested naming convention for metaspace names is `env_MDM` where `<env>` is a three to five character acronym for the environment.
- You can partition the JMS setup between environments by using different port numbers or JMS server machines.

**Considerations Sizing in HAFT Installations**

When sizing your environments, set the CPU maximum loading to 75% for any single TIBCO MDM node when one or more of the other nodes fail.

CPU usage is much lower in normal operation, but in this type of sizing the application can perform without degradation in case of failure. Contact TIBCO support for assistance in setting up your environment.

Set up a web server or load balancer to equally distribute the load among all TIBCO MDM instances. Ensure that sticky sessions are configured. Sticky sessions mean that once a session is started on one of the TIBCO MDM instances all subsequent requests for that session will be sent to same TIBCO MDM instance.

Size the cache to keep all the master data in memory. If the required memory is greater than 16 GB, consider using an external cache server (and switching TIBCO MDM to use leech mode). External cache servers maintain cached data even when TIBCO MDM is restarted, to avoid reloading.

Use the disk sizing spread sheet provided by TIBCO Support to estimate the disk space required, as the space requirement depends on your repository definitions.

**EMS Configuration in HAFT Installations**

When setting up highly-available or fault-tolerant (HAFT) EMS servers for TIBCO MDM, use database-backed queue storage (against a DB cluster).

This alleviates any issues with SAN boxes and file-based replication. Note that a database-backed EMS storage can negatively impact performance. Consult the EMS documentation to understand the benefits and drawbacks of using EMS. TIBCO recommends that you set the expiry on Change Notification Message queue to a small number, for example one to two minutes.

**Security Considerations**

Perform the security consideration practices to improve your system’s security.

The following information lists some important security considerations to use:

- Use SSL configurations between TIBCO MDM and the EMS cluster to improve security if the EMS servers are in different geographical and network locations.
- Use authentication to secure the EMS connections.
- Create a list of allowed servers to protect communication between Patterns and TIBCO MDM when you start Patterns engines.
- Set up an authentication realm to secure the JMX connection.
- Use SSL for browser to TIBCO MDM server connections.
Value for Logging

Set logging to DEBUG for development and test environments and to INFO or ERROR for production environments.

If access to the TIBCO MDM machine is restricted for developers:

- Move the log files to separate directories and drives or mounts so developers can read the contents.
- Delegate an TIBCO MDM user as the Support Engineer to login to TIBCO MDM to get logs and run diagnostic queries.
- Set up the TIBCO MDM instance to access JMX remotely.

ActiveSpaces and TIBCO MDM

TIBCO MDM uses ActiveSpaces, an in-memory grid, to minimize the reading the database and reducing the end-to-end response time.

Configuring ActiveSpaces to provide the best performance is an iterative process. Use the following guidelines to help configure ActiveSpaces for your own needs.

Seeder and Leech Modes

Although TIBCO MDM is configured in Seeder mode, it is not always the optimal choice.

You can change the modes using the TIBCO MDM Configurator or by making changes to the ConfigValues.xml file. You must restart all the TIBCO MDM instances after making such changes.

The following information describes and compares Seeder and Leech modes:

- **Seeder**: In Seeder mode, TIBCO MDM uses the CacheConfig.xml file to create its own embedded cache node on the same machine that TIBCO MDM is running. This setup is advantageous because the allocated memory is running on the same machine as TIBCO MDM and therefore has substantial performance gain. However, this increases the memory requirements on the host machine considerably.

- **Leech**: TIBCO MDM in leech mode does not create an embedded cache and relies on external cache servers. This setup means that you can spread out the cache over several machines. If TIBCO MDM cannot access any cache nodes it fails and does not function correctly.

If TIBCO MDM instance is a Seeder, the memory used by the TIBCO MDM instance is also the same as the cache configuration. This memory is in addition to the JVM heap assigned.

Virtualized Environments and ActiveSpaces

TIBCO does not recommend virtualized environments for ActiveSpaces and any operation on a virtual machine that stops or re-initializes the network makes the cache non-functional.

For example, if you plan to take a snapshot, you should first shut down the caches.

Considerations for Sizing in ActiveSpaces

TIBCO Support has a collection of resources for accurately estimating the space required for your solution.

TIBCO MDM provides a Cache Calculation spreadsheet to use in your calculations. For more information on documentation, refer to TIBCO MDM System Administration.

As in all memory-based applications (especially long running ones), ActiveSpaces is susceptible to memory fragmentation similar to the main memory fragmentation. Symptoms include increased CPU utilization, slower response times, and higher page reads.

TIBCO recommends scheduling a restart to clean and rebuild the cache. Also, never allocate more than 80% of physical memory to ActiveSpaces. For the cache to utilize the memory efficiently, all instances of cache must use the same CacheConfig.xml.
**ActiveSpaces Configuration**

TIBCO MDM ships three configurations, any of which are a good starting point for different environments.

The three configurations are as follows:

- Config/CacheConfig.xml
- CacheConfig.dev.xml
- CacheConfig.large.xml

All the TIBCO MDM nodes must use identical cache configuration file and all seeders must have identical memory allocations. You can accomplish this by using the same CacheConfig.xml for each TIBCO MDM instance. When an external cache server is added, it must support the memory specified in this configuration file.

Setting the replication count to greater than 0 configures ActiveSpaces to make copies of data across different ActiveSpaces nodes, but it requires more memory. CacheConfig.large.xml has preconfigured caches which must be replicated. It is not required that other caches be replicated, however if replication is configured, it increases the data availability when one of the nodes fails.

Review the following two example CacheConfig.xml files provided with TIBCO MDM:

- CacheConfig.dev
  and
- CacheConfig.large

For more information about your configuration, refer to *TIBCO MDM System Administration*.

Setting the replication count to greater than 0 configures ActiveSpaces to make copies of data across different ActiveSpaces nodes, but it requires more memory. CacheConfig.large.xml has preconfigured caches which must be replicated.

**Optimize CPU Utilization**

Due to the inbound-outbound nature of TIBCO MDM, you cannot achieve more than 45% CPU utilization.

A large number of cores and CPUs produce a large number of threads, which result in a high throughput. The CPU utilization increases if the Inbound-Outbound is performing well including the database, networks, and file system performance. To increase the CPU usage, you can:

- Increase thread counts for workflow queue, HTTP, web services, AsyncCall Queue, and Active login. As you increase the threads, you need to allocate more memory to JVM. A reasonable estimate is 300 MB per workflow or async queue listener (whichever has a higher number of listeners) or 200 MB per web service thread. Increasing the thread count without increasing the JVM heap, may result in Out of Memory errors. If your bundles are of average size (20-40 records per bundles), the memory per thread is 250 MB and 150 MB respectively.
- Start more TIBCO MDM instances on the same machine and assign each node identical memory and threads to maintain a well balanced load.
Database Management

Best practices are also defined for database management in MDM.

The best practices for database management includes:

- Table spaces
- Database performance

Table Spaces

Generally large tables should be kept in separate table spaces but newer technologies may make this practice redundant.

Oracle Automatic Storage Management (ASM), for example, does not require storing large tables in separate tablespaces. Nevertheless, if you keep a large table in its own separate tablespace, the database administrator can manage the tables more efficiently.

Database Performance

Database performance changes as data is added or deleted. When more than 10% of data has changed or been added, a database may require DBA attention.

The DBA should review the following:

- Set up a job to collect optimization statistics regularly.
- Set up a job to generate Automatic Database Diagnostic Monitor (ADDM), Automatic Workload Repository (AWR), or similar reports at regular intervals.
- Review the report for recommendations and adjust database parameters accordingly. For example, reports may indicate changes to memory allocated to a database instance. If an ADDM report is regularly checked and acted upon, no database performance issues can occur.
- Regularly purge data using the purge program. (See Database Purge Scheduling.)
- If there are many deletes (due to purge), indexes and tables may become fragmented and after reviewing the statistics report you may have to defragment the indexes regularly.
- If a database report shows that inserts or deletes are running slow, it may indicate that:
  - Disks are slow or access paths are slow. Even with a fast SAN, disk performance can be affected if database storage options are not configured correctly. For example, for Oracle using ASM with a FAST SAN resolves most of the disk related issues.
  - Table or index is fragmented. This happens when you need to import and delete a lot of data using purge. Defragment the indexes.
  - Too many indexes have been created. Eliminate some of the indexes if you can.
  - Some records may have too many versions. Consider purging the older versions.
  - Too much concurrency. Consider better database configuration or bigger capacity hardware. Or consider more cache to take some load off the database.
- Take a full backup if you are using an SQL Server. Incremental backups can take significant amounts of time and perform many reads and writes.

Database Large Object (LOB) management

Most of the XML documents generated during the workflow are stored as large objects (LOBs) in GENERALDOCUMENT tables.

LOBs are special objects and require special attention. For example, Oracle may not release the space assigned for LOB storage based on configuration.
LOBs are sometimes difficult to manage, therefore TIBCO recommends planning early on.

**Miscellaneous Factors for Database Performance**

As the number of rows in a table increases, partitioning is necessary.

If the Database Administrator requests that partitioning be done, contact TIBCO MDM engineering for information on partitioning.

**JMS Management**

The JMS server plays a very small part in the overall application performance.

You should not tune the JMS server unless it is absolutely necessary.

**EMS**

The default prefect is five but in most cases, you should change the prefect to None.

You should only use a prefect of five if the overall throughput is the main requirement and the system is always heavily loaded.

**WebSphere MQ**

For large volumes, size the logs correctly to avoid runtime errors.

For more information on how to configure log files, refer to the *TIBCO MDM Installation and Configuration* guide.

**Caches**

For development and most functional testing, a single-node installation with cache is sufficient. You need not set up a central cache server.

To view the cache status and characteristic, use ActiveSpaces tools. Additional information is available through JMX console. See Mbean for Cache.

**Sequence Numbers and Caches**

Sequence numbers are cached by TIBCO MDM and, you should not change the sequence numbers (alter or read) directly using any custom SQL code.

A sequence number changes when you use the sequence in an SQL statement.

**Objects and Caches**

TIBCO MDM caches a large number of objects. If the database tables are updated using scripts, check with TIBCO support to see if the cache is impacted.

If the cache is impacted, clearing the affected cache using JMX or the provided scripts is usually sufficient. In rare cases, you may need to restart the whole cluster.

**Management of Multiple TIBCO MDM Instances with Caches**

Multiple instances of TIBCO MDM use cache for exchanging job status and for distributed locks.

If this information is lost due to the abnormal shut down of TIBCO MDM instances or cache instances, the following events may occur:

- Distributed locks may not be released.
- Distributed locks may be released prematurely.
- Jobs which are processing data batches may hang.

When you observe this type of abnormal behavior, shut down and then restart the whole cluster including all TIBCO MDM instances and cache instances.
To avoid such unacceptable situations:

- Allow TIBCO MDM instances to shut down gracefully.
- Set up replication for some caches.

**Cache Configuration**

Correct cache configuration is important for optimal throughput. TIBCO support provides a cache computation sheet to help you calculate the required memory for various caches.

The most common installation error is incorrect cache configuration. Some other common errors are:

- Not sizing the cache for expected data volume
- Incorrectly setting up the cache cluster so that each instance is independent rather than joined.

To avoid such errors, keep as much as data in memory as possible to reduce evictions caused by insufficient memory. Ensure that memory allocated for caching is never more than the free physical memory.

**Security**

Security best practices discusses security considerations associated with installation and configuration. The best practices include:

- Manage users and roles
- Security models
- Lightweight Directory Access Protocol (LDAP) integration
- Data encryption
- Security auditing

**Managing Users and Roles**

Currently there is no UI function to create or modify roles. Instead, roles are managed by importing role metadata.

**Procedure**

1. Export the role data using **Export metadata**.
2. Edit the exported XML to change the roles (including new roles, removal of roles).
3. Import the role data using **Import metadata**.

**Result**

Instead of assigning permissions to users directly, assign them to roles. This is easier to understand and manage.

When a new enterprise is created, immediately create an admin user for the new enterprise and use this admin for the remainder of the setup.

**Work Supervisor Role**

The Work Supervisor role has some specific permissions that you can assign to any other role.

To simplify things, limit the supervisor functions to one role and do not assign the Work Supervisor role to too many users because it dilutes control and may negatively impact performance.
Support Engineer Role
The Support Engineer role should be restricted to technical support engineers.
Delegate at least one user to the Support Engineer role so that person can obtain details from the configuration and database when the need arises.

Security Models
Value-based security for record lists can result in a filter query every time the record list is shown, which may negatively affect the performance.
To avoid degrading performance, do not use a complicated filtering criterion.

Lightweight Directory Access Protocol (LDAP) Integration
LDAP integration and single sign-on is driven by configurations. Generally, you could change the behavior using the configuration and do not need to make any code changes.
However, because LDAP policies vary greatly, you may need to write a Java program to customize the behavior for multiple directories or to extract headers that cannot be mapped using configuration properties.

Data Encryption
TIBCO MDM does not require and is not affected by data encryption while data is on the wire or stored in a database.
A column, table, or tablespace in a database can be encrypted without any impact on the application. TIBCO MDM itself does not provide any tools to encrypt data (except for passwords stored in configuration files, which are always encrypted by TIBCO MDM).
You can encrypt data in the cache. However, TIBCO MDM does not currently support named user to cache connection.
Encryption of data on the wire is configured using SSL for all data transfers, including:
- Browser-to-web server
- TIBCO MDM-to-JMS
- TIBCO MDM-to-database
Patterns does not provide any encryption for the data it stores. To secure the connection, specify the IP addresses of an TIBCO MDM server to Patterns and specify this list when a Pattern server is started. When you specify this list, it accepts the connections only from specified servers (connections from local hosts are always allowed). Although the data transmission is not encrypted, you can encrypt data stored in a database by using the features provided by the database.
You can also encrypt data stored on JMS by using the features provided by the JMS vendor.

Security Auditing
TIBCO MDM tracks and maintains all changes to permissions assigned to roles and users, and roles assigned to users.
The history of changes for audit is stored in the database table which you can access directly using database tools.

Analytics
You should consider and plan for TIBCO MDM Analytics in every project.
The Spotfire limited license bundled with TIBCO MDM analytics will help you consider and plan for TIBCO MDM Analytics in every project. You can only use the Spotfire license with TIBCO MDM. You
can also use it to view the operational statistics and data quality. Contact the TIBCO Professional Services Group to obtain sample projects.

**Synchronization**

Synchronizing a large number of records (in the millions) takes a considerable amount of time.

If you have millions of records, consider whether the history of synchronization is really needed and if the history is not required, use the database dump workflow with the ExtractDataToDelimitedFile activity. Alternatively, use the data extraction framework provided through DBExtractor.

You can read data directly from the database tables using an ETL tool such as Kettle.

**Rulebases**

You can use rulebases within TIBCO MDM to complete data validation, record modification, and data lookups.

Rulebases although powerful are prone to misuse. The mentioned best practices need to be considered when working with rulebases.

All rulebases are evaluated top down. Any rules that make changes to fields at the top of the rule may get overwritten by rules evaluated later on.

You should not make changes directly to rulebase files after they have been deployed in TIBCO MDM because this can lead to lost changes.

Always use Business Studio to develop and deploy the rules. The only exception to this is rulebases called during workflows to validate data being worked on. In such cases, you should manually place the rules in the correct folder and point the workflow to this rule.

**Validation Rules**

Use the validation rule to confirm that the data presented to TIBCO MDM is correct and in-line with the business rules.

You can also specify validation messages.

**Validation Rules for Data Imports**

You can write validation rules for data imports from data sources. If you have a large number of data sources with many rules, write separate rules for each data source. You can then write a simple rule to identify which data source is being imported and then call the correct rulebase. This process eliminates writing one monolithic rulebase which takes time to evaluate.

**Initialization Rulebase**

You can use an initialization rulebase to default initial values creating a new record. Use this rule type to default initial values for fields.

All the rules which apply when data is saved or modified should be defined in the validation rule file. Initialization of data when creating a record should be placed in an initialization rulebase. The initialization rulebase is run only once before a record is initialized and it is not run at all after the record has been initialized. Use initialization rulebase to default initial values while creating a new record. All the rules which apply when data is saved or modified should be defined in the validation rule file.
Use Enumerations with Rulebases

If you have a frequently changing list of enumerations a user can select, consider using data source as a lookup table.

If the enumerations themselves are reference data, use a repository to store the values and a user link to get the values. This eliminates the need to change the rulebase when a valid value list changes, and lets TIBCO MDM users manage the values themselves.

Security and Rulebases

You can optimize the implementation of security by checking against roles by evaluating the role list at the start of the rulebase and setting a flag.

Use the flag for better performance, instead of checking it against the roles.

For more efficiency and easier management, use decision tables for simple rules for access control and attribute visibility.

Optimize Performance with Rulebases

If your rulebase has many conditional sections, separate them into smaller rulebases to provide better performance and modularize your code.

Even though making the rules conditional provides some performance benefits, writing them in separate rulebases and using include makes it even more efficient.

Use Lookups with Rulebases

Repository lookups (for example, a drop-down list to show records from another repository) can be very expensive if the lookup is performed against a large repository.

Such queries do not use cached data because it assumes that the data in the target repository could change any time. Such lookups have been found to be the primary cause of many slow user interface service requests. The same applies to any SQL based lookups.

TIBCO recommends that you initially design modular rulebases and separate the rules by usage based on conditions. Important conditions to consider are:

- CHANGED: this condition indicates whether the record data has changed. If the record data has not changed, you can skip some validations.
- RECORD_ACTION: this context var indicates the action being taken. that is ADD or EDIT (check with rulebase guide).
- RELATIONSHIP_CONTEXT: this condition indicates the direction of traversal.

Indexes and Rulebases

To determine which indexes are required, do a test run and extract the query from the debug log. Then run the results through the database tools.

You need to create only the required indexes because too many indexes slow down Data Manipulation Language (DML) operations.

Use Drop-Down Lists with Rulebases

The Drop-down lists that have a large number of entries slow down the application, especially in the UI.

If your data requires a long drop-down list, consider the following strategies:

- Create a more selective drop-down list by using the context. Consider using GROUP_ID to limit the entries for PARTICIPENT_TYPE.
• Create cascaded drop-down lists by introducing groups. For example, the entries in the city field would depend on the state field so that you do not have to list all cities in the drop-down list. The entries in the city field are populated after state field is selected.

• Redesign to ensure that you do not end up with fewer than 100 entries.

The Drop-down lists based on the data sources are cached during execution so they do not impact performance. Datasources with a large number of entries impact the UI performance. SQL-based datasources may provide the wrong results if data in the underlying table has changed but TIBCO MDM is not notified of the change by executing UPLOAD.

For usability and rendering purposes, limit the number of choices to fewer than 50 when defining drop boxes to represent a list of predefined choices.

Propagation and Rulebases
Propagation should be designed to work in one direction. If you have the following scenario, you may get unpredictable results:

• Propagate from Repo A to Repo B
• Propagate from Repo B to Repo C
• Propagate from Repo C to Repo A or B

Depending on the change and the use case when the propagation happens in both directions, the order of propagation may be different.

While propagating the entries to related records, note that updates are not allowed if the related records are already modified, unconfirmed and part of another workflow. In these cases, you must specify the target record state as Draft in the EvaluateRulebase activity to isolate the changes.

Constraints and Rulebases
Use the timing log to see which constraints are slow. This gives you a starting point for analyzing performance issues.

Use JMX to see how often constraints and rulebases are executed.
Design Considerations

Best practices are implemented during the design and development of a TIBCO MDM solution.

Repositories

A repository can have a large number of attributes and a well-designed repository keeps these attributes separate.

For example, the number of base attributes that apply to all records across all categories should not exceed 100.

Design of Data Models for Repositories

Use standard database modeling principles to identify attributes, objects, and relationships during the initial design of new data models.

After you have identified all the objects and relationships, consider the questions in the following table:

<table>
<thead>
<tr>
<th>Design Principles Checklist</th>
</tr>
</thead>
<tbody>
<tr>
<td>Are there frequent traversals to many levels of relationships? Should these objects be denormalized to reduce the depth of hierarchy?</td>
</tr>
<tr>
<td>Are there many small objects which can be denormalized into one bigger object? This may result in a sparse table with many null attributes. If you have many small tables which are associated with another table in a relationship, consider denormalizing it and merging them into one.</td>
</tr>
<tr>
<td>Are there too many relationships between objects? Could these relationships be reduced?</td>
</tr>
<tr>
<td>Is a relationship cardinality expected to be very high (for example, 100s)? If so, consider removing the relationship and using a softlink. Alternatively, you can use an intermediate object to group the child records.</td>
</tr>
<tr>
<td>Are the attribute sizes, especially for Strings correct? After the data is populated, you can change the attribute sizes, however it is difficult. Some of the attribute size changes are not allowed, so TIBCO recommends that you define the correct attribute size during the design stage.</td>
</tr>
<tr>
<td>Are the attribute data types correct? If in doubt, consider using String. After the data is populated, you can still change the data types but it is difficult. In some cases, change in data type is not possible due to stale data which causes the conversion to fail. In String datatype provided attributes are defined as String. In this case, TIBCO MDM skips the checks to see if the data is the appropriate data type. In such cases, use validation rules to validate the data.</td>
</tr>
<tr>
<td>Do you really need effective dates for repository and relationships? Effective dates impact performance and introduce complexity in rules, use this feature sparingly.</td>
</tr>
<tr>
<td>Can you present any perspectives in a simplified view of the data model?</td>
</tr>
<tr>
<td>Are there a large amount of specialized objects that result in small tables, each with little additional information? If so, combine them into repository.</td>
</tr>
<tr>
<td>Does the repository have a large number of attributes? If so, consider splitting it.</td>
</tr>
</tbody>
</table>
Design of Data Models for Repositories Using the Out-of-the-Box UI

If you are using a TIBCO MDM out-of-the-box UI, the complexity and depth of data model can have a significant impact on the UI.

Small, highly normalized data models, for example, result in poor UI usability and performance because you need to perform many clicks to view all data.

To optimize out-of-the-box UI performance and usability, consider the following points when designing a data model:

● Each attribute group is typically mapped as one tab on record UI. Attribute groups are arranged according to the sequence assigned and attributes are arranged in the order of position within the attribute group. You can provide additional UI specifications to combine data from more than one attribute group or adjust the positions of groups and attributes by changing the order.
● The relationships and related records show up in the navigation hierarchy.
● Effective dates introduce the drop-down boxes to enable users to navigate between different time periods.
● Using Perspectives you can select different views. By default, the UI provides a set of attributes in record header and in relationship hierarchy. You can change this by configuring the desired attributes through ConfigValues.xml.

Attribute Groups

Attribute groups in TIBCO MDM group together attributes that are similar or share security constraints.

You can use Attribute groups to:

● Display records in an automatically generated tabbed UI.
● Define security and data visibility settings.
● Assign data custodians for governance and route work items.
● Assign rejection comments for a group instead of individual attributes.

You can apply rules to attribute groups to manage and organize attributes and to perform the following actions:

● Hide entire attributes-based users, roles, or actions (for example, create and edit)
● Group together attributes that have similar behavior. For example, you can group together read-only attributes, or attributes that are hidden for certain users.
● Create a data entry wizard by hiding attribute groups until certain conditions are met. For example, you can hide subsequent attribute groups until the first attribute group has been populated.
● Hide attribute groups from specified users. Such rules must also be reflected in any validation rules. Otherwise, the rule may fail validation because a user cannot populate a field they cannot see.

Identifying Related Entities and Attributes for a Repository

After all the entities are identified, you need to identify all related entities and attributes.

Procedure

1. Normalize the resulting model.
2. Associate the subentities with entities through relationships or foreign key lookups.
3. Separate the entities from the list that store data coming from external sources. For example, if you are getting addresses from external systems to store in TIBCO MDM, the address may be mapped
to the repository ADDRESS and the data acquired from the external system is a data source DS_ADDRESS.

4. Denormalize the tables with access patterns in mind. For example, you can have a Customer entity where a customer has a phone number stored in Phone entity. However, if one of the phone numbers is the main phone that is always accessed along with the customer and is logically considered part of the customer details, you may want to store this phone number as an attribute in the Customer entity itself.

5. Group attributes logically and assign positions to the attributes within group so that they are sequenced in a logical order.

**Uses of TIBCO MDM Relationship Table in Repositories**

The TIBCO MDM relationship table is a generic association table that stores associations and relationships.

The types of associations and relationships are as follows:

- Relationships between records, defined in the repository model
- Association of repository with output maps and input maps
- Association between input maps for multirepository import and between output maps for multirepository export.
- Association between attributes and attribute groups.
- Hierarchy of classification codes and association with records.

Different types of relationships are identified by different relationship types. For example, a separate table is used for storing the associated data for each relationship associated with a repository that also has any relationship attributes.

More than one relationship of the same type is not possible between any two records. To create such relationships first create an intermediate association object.

**Sparse Repositories**

Sparse repositories have many columns that do not apply to all the records and may be null for most. Such null columns are efficiently handled by databases.

TIBCO MDM does not allow inheritance. This means that if the model requires you to model subobjects that vary slightly, model them in the same repository and use a record type to identify different types of objects. This is called denormalization of the data model.

Repositories have a limit of 1024 attributes. However, with category-specific attributes, you can define unlimited attributes as category-specific. You can use this method to exceed 1024 attributes in a repository.

It is a best practice to use shared tables whenever possible for multivalue and category specific attributes. Too many tables adversely affects performance. Use a separate table for a column which is likely to be present for most records and if number of records are large, resulting in a large table with count more than 50 million.

**Record Relationships**

When using record relationships in TIBCO MDM, you can bundle together and retrieve collections of related records.

When a parent record is queried, TIBCO MDM must evaluate all of the related records and potentially retrieve these and any related subrecords (if configured to do so). Relationships in TIBCO MDM can be a powerful tool. However, misused relationships can negatively impact performance.

Use relationships only when they are required to obtain bundles of data. Some examples include the following:
• Using the relationship between a rail line table and the track table you can query all of the tracks that make up a line.

• Using the relationship between CarModel and CarParts you can identify which car Models use which car parts. (You can use reverse relationships in TIBCO MDM, for example, which car models use a car part.)

**Identify the Record Relationships during the Design Phase**

Identify the relationships early in the design phase because relationships affect the whole solution.

The later you define or remove relationships the larger the impact it has on the design.

TIBCO MDM supports the concept of a many to many relationship. In Business Studio you can document different multiplicities, but these need to be enforced by using rulebases to travel up and down the relationship tree counting instances.

**Manage the Cardinality of Record Relationships**

TIBCO MDM manages all relationships as peer to peer and many-to-many. Therefore, you do not need to define cardinality.

However, it is better to define cardinality in the repository model for documentation. If cardinality has to be enforced, use the rulebase.

If the cardinality is expected to be more than 500, you encounter performance issues. For example, if a Car comprises over 500 parts, you should group the parts (door parts, cabin parts, suspension parts). This reduces the amount of ‘crawling’ over relationships that TIBCO MDM needs to complete. Larger cardinality results in performance degradation for all channels, especially on the UI.

Use the following strategies to keep cardinality manageable:

• Create an intermediate group object. For example, if a customer has more than 500 accounts, create an account group object to bunch accounts such that each group has no more than 100 accounts.

• Configure TIBCO MDM to exclude relationships from parent to child or reverse if the navigation is always in one direction.

• Define the relationships as a softlink. The relationships between records are explicitly maintained if defined as a softlink, and are not searched as version-specific.

**Management of Hierarchies**

If a data model is deeply nested or has a deep hierarchy, you have to configure the switches.

The switches are described in the following table to control the depth and optimize performance:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>com.tibco.cim.optimization.recordbundleview.optimaldepth</td>
<td>Defines the depth of the bundle to be loaded for view. This parameter is useful when you do not want to review the records from lower levels of the hierarchy.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------------------------------------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>com.tibco.cim.optimization.recordbundlevalidation.depth</td>
<td>Defines the depth of the bundle for validation. If none of the nodes change (at any level) in the hierarchy, the validation is not done for child records of the modified node at depths higher than this value. Changed data is always validated. Use this parameter when the validation of the child records are not dependant on the data of parent records. By indicating that the child records need not be validated unless changed, TIBCO MDM can optimize validations of the hierarchy.</td>
</tr>
<tr>
<td>tibco.optimization.recordbundle.excluderelationship</td>
<td>Specifies which relationships you can ignore for navigation through bundling. The list can include relationships defined for either direction (forward or backward). It is typically used to enforce one way navigation.</td>
</tr>
<tr>
<td>com.tibco.cim.optimization.recordview.skipcustomvalidation</td>
<td>Specifies that custom validation class specified for a record can be bypassed for viewing. The default value is true, unless you want to override the validation class with some custom code.</td>
</tr>
<tr>
<td>com.tibco.cim.ui.optimization.recordsearch.relationship.depth</td>
<td>Defines the depth of the hierarchy available for Configuration of the search panel. This switch determines the depth of search within the hierarchy.</td>
</tr>
<tr>
<td>com.tibco.cim.optimization.recordsearch.relationship.depth</td>
<td>Defines the depth of the hierarchy for search. It applies to web services.</td>
</tr>
</tbody>
</table>
**Softlinks**

Consider using a softlink if two records are related and referred to together but are not updated together.

Softlinked records are obtained whenever required and have the following attributes.

- You cannot propagate data down to softlinked records.
- Querying softlinked records using the SOAP GetRecord service does not return any soft linked records.
- Softlinked records are not validated when the root record changes.

Despite these limitations, softlinks are an effective way to relate records together in a fast, efficient manner.

**Effective Dates**

Future effective dates in TIBCO MDM are a means to complete and confirm records that become active at a specified date in the future.

This is useful for expected changes such as new product releases, changes to employment details, and price changes.

When using future effective dates, keep the following best practices in mind:

- Always try to use effective dates in a linear fashion between record versions. Ensure that the later version of a record has the later effective date than the earlier versions. If this is not the case, the earlier versions (for example, version 5 with a later effective date than version 6) can overwrite the golden copy, which can lead to data inconsistencies. To avoid this, use rulebases and approval cycles to confirm the dates being used are correct and that earlier dates are not used.
- Effective dates in repositories and records have an impact on both memory and performance in TIBCO MDM.
- Versions are only identifiers and do not imply ordering. A version 6 may become the golden copy before version 3.

**Data Source Identification and Design**

Developers spend a great deal of effort to ensure that the data from other systems is mapped correctly to TIBCO MDM. You can import such external data using data source.

Most data sources have already been identified for use in TIBCO MDM and new ones are built to fill in gaps or present the existing information better. The following best practices should help design data sources to gain maximum benefit:

- Whenever possible identify which data sources present the most accurate collection of data for any repositories and use it to populate your primary key fields and any other data fields.
- Use the same data source for any tables that store similar information. For example, if you have a data source that provides all customer ID information, use this for all tables (such as Business and Normal customers) where the ID is required. Splitting it over two or more data sources gives rise to inconsistencies and degrades the quality of the data within TIBCO MDM.
- You can join more than one data source to merge data into a single repository. You can then map different data source data to different parts of the repository in just one action.
- Do not transform the data while mapping data source to input map using input map expressions, which is slow and has limited functionality. Instead use rulebase during import to transform the data.
If a lot of data transformation and lookups are required, prepare the data before importing it into TIBCO MDM. While TIBCO MDM is able to complete lookups and change data, it may be computationally expensive and time consuming. For example, a simple data lookup where an ID is converted into a text value is acceptable within TIBCO MDM. However, if it has to look up a value and then execute a collection of rules based on this value, which then changes other attributes, TIBCO recommends performing this externally. TIBCO MDM executes these rules every time a record (in hundreds or thousands) is presented to it. There are a variety of ways to achieve this. For example, you can use the following:

- TIBCO Clarity for data discovery and data transformation.
- ActiveMatrix BusinessWorks (or similar) to access the data required from the source system. Process the data internally to produce an end result that adheres to the required business rules.
- Other ETL tools such as Kettle.

**Conventions of Naming**

TIBCO MDM provides a large number of resources, including data repositories, data sources, synchronization profiles, and synchronization formats.

TIBCO recommends using a naming convention for each type of object. The following are the recommended suffixes for each type of object:

- **Repositories**: None
- **Back end Systems**: BS
- **Data sources**: DS
- **Input Maps**: IM
- **Output Maps**: OP
- **Classifications**: CS
- **Synchronisation Formats**: SF
- **Synchronisation Profiles**: SP
- **Subsets**: SB

TIBCO recommends providing a descriptive name. The name of data sources, for example, should reflect where the data source is getting its data and from what type of data it contains. Studio projects can be assigned a prefix based on the project’s containing enterprise. For example, PR1 for a project from enterprise 1 and PR2 for a project from enterprise 2.

Ensure that all repositories are given a table name instead of using generated table names. Specifying table names makes the names consistent across different environments (development, test, production). Using generated table names can lead to portability issues. The table name you specify must be unique in the database schema. The naming convention for table names is to prefix it with a project acronym. For example, a customer table name can be E2_CUSTOMER where E2 is the project name.

By following these it should be possible to give unique names to each resource in TIBCO MDM that reflects its type, function, and the enterprise to which it belongs.

**User-defined Table and Column Names**

TIBCO MDM enables user-defined names for database entities such as tables and columns. TIBCO MDM automatically generates names with internal conventions if no table or column names are provided.

These automatically-generated names can be cryptic or generic. Automatically-generated names are likely to change if the table is ported from one repository to another. Metadata associated with automatically-generated names can also change. In most cases, TIBCO recommends specifying the
database tables and columns. Automatically-generated names are useful in development environments, where users do not want to name tables and columns and do not want to expose database details.

User-defined tables and columns make it easier for external tools like TIBCO Spotfire to access the database. User-defined names are also more easily understood by SQL programmers.

User-defined table names must be unique within the same database instance. You cannot assign the same table name to more than one repository even if they are defined in different enterprises as long as these enterprises are deployed in the same TIBCO MDM instance. You cannot import metadata into another enterprise within the same instance because attempts to create a duplicate table.
Import

Best practices are also implemented for the import feature of MDM.

Initial Loads

Database Loader quickly imports very large data provided the data is clean.

Master data stored in TIBCO MDM goes through validations and cleansing, but when the project is initiated a large amount of existing data is loaded into TIBCO MDM that may have to skip such validations. Common practice is to externally clean the data before importing it using Database Loader.

In many cases, cleaning the existing data per validation rules is not possible. For example, existing data may be missing some required attributes. To import this data into TIBCO MDM, there are two strategies:

- Current Data Acceptance
- Initial Load With Cleansing

Both strategies are situational, and the best choice depends on business needs.

Current Data Acceptance

Businesses typically choose this strategy when they are replacing an existing master data management solution (such as SAP) with TIBCO MDM or when a company uses legacy source systems to create a valid snap shot for master data.

The company may decide that the data being presented was good for their master data needs until this point. This effectively creates two classes of master data: before and after TIBCO MDM go-live (stricter data rules are used to validate data after going live).

You can use Database Loader to load the before TIBOC MDM go-live data. Database Loader skips the validations and imports the data. Regular import of after migration data triggers the new validation rules when accessed for the first time. You may need to modify this type of data to bring it into alignment.

Use the current data acceptance approach if the before TIBCO MDM data is uplifted over the time, giving you more time to work on the data.

The potential drawback is that it may leave records in a state which cannot be updated later as the validation rules may have several preconditions that cannot be met. These cases need manual intervention to correct the data.

Initial Load with Cleansing

With this strategy all the data is cleaned completely before importing it into TIBCO MDM.

Although you can acquire clean data quickly using this strategy, you need a reasonable amount of effort to prepare the ETL (extract, transform, and lookup) solution. Typically this is a one-time activity.

To clean the data:

- Use an extract, transform, load (ETL) tool to transform, lookup, and enhance data as required. Kettle is a free ETL tool that can quickly process a large amount of data.
- Write this data to a text file and then import it into TIBCO MDM using Database Loader.
**Input Maps**

Instead of using input maps for data transformation, keep the input maps simple with no expressions. Use rulebases during the import step to transform the data.

Expressions entered in the input map are limited to simple expressions that work on one or more attributes. Expressions cannot include procedures and they are limited to the specific database being used. To specify sequences in the expression, you must implement an SQL function.

**Importing Nested Data Mapped to Multiple Repositories**

Provide information regarding the join keys on the datasource to determine the relationship while importing multiple repository and multiple data sources that have multilevel joins.

You can do this by creating a property file which has information about the joins on the staging tables.

**Procedure**

1. Identify the relationship for which you need additional joins to determine the relationship.
2. Next, identify the parent InputMap and get the ID from the database. Review the staging table with STG_<inputmapid>.
3. These additional join keys should be part of both parent and child repositories and if they are not, create one. Make sure their Database Column names match.
4. Ensure you map these attributes to the correct attributes in the datasource, so that the join is correct.
5. Create a folder COMMONDIR\<enterprise internal name>\inputmap
6. Create 'import.prop' under the 'inputmap' folder which is created in the previous step.
7. Edit the property file and provide additional join keys for all your inputmaps where additional joins are needed.

- The properties file is a name value pair.
- Name part - Syntax - JOINKEY_<CATALOG NAME>_<INPUTMAP
- <NAME>_<RELATIONSHIP NAME>
- Value part - Database Column name from stg_inputmap table. Additional join keys (multiple columns can be specified separated by comma [,])

For example:
Repository Name - B
Inputmap Name - BParentChild
Relationship - B2C
Additional keys - CDIVISIONID,CA
JOINKEY_B_BPARENTCHILD_B2C=CDIVISIONID,CA

The property name has to be specified in upper case.

**Avoid Failures During Import**

Errors are common during import. Avoid failures by following the guidelines that help you to avoid common errors.

It involves the following practices:

- **Import of large batches**
• Changes in the input map during the import
• Imports in the approval stage
• Use of the CONTAINS attribute for imports
• Date loading

Import of Large Batches
TIBCO does not recommend importing large batches and recommends an optimal size of 200K to 500K.

Larger chunks of data require bigger demands on the cache and database so you should split batches into smaller chunks to improve performance. A chunk size of 500K, requires hardware setup and tuning.

When importing large batches, remember that new records take a longer time to import than existing records, and that records with relationships take a longer time than those without relationships.

As more data is imported into a repository, subsequent imports are slower. Collect database statistics whenever row count changes by about 10%.

Changes in the Input Map During Import
Making changes to the input map when the import is running results in import failure or incorrect results.

Imports in the Approval Stage
Do not leave imports in the approval stage for a long period. Imported records are saved as draft, and the visibility of drafts is limited to the process which created it.

Such records are not visible to other processes including other imports. The longer the records stay in the draft state, the higher the chance of conflicts.

Multiple pending imports and conflicts usually lead to different outcomes depending on which records are approved first. Once the records enter conflicts, it is difficult to manage the conflicts and to predict the outcome.

Use of CONTAINS Attribute for Imports
The CONTAINS attribute is deprecated except for specifying the DELETE or the DELETEALL commands.

You should not use this attribute for creating or modifying any relationships, rather use input map hierarchies and explicitly map related records to ID/Ext or related repositories.

Date Loading
Date loading is always tricky because loaders support many variations of date formats.

You must specify the date format for the datasource. Define this attribute for the datasource as String and then map it to a date repository attribute.

All dates in a data source should be in the same date format. All dates in all datasources used in one import should have the same date format.

Use of SQL-Based Datasources
Avoid using SQL based datasources for data that changes often. TIBCO MDM can cache the data contained in the SQL datasource.

Unless an explicit upload is done when the underlying data changes, TIBCO MDM does not know that the data has changed. The data caching happens when the SQL data sources are used in rulebases.

Avoid importing multiple repositories and hierarchies based on the view and SQL datasource.
Import of Control Switches

Many different performance switches can be combined to create a fine-tuned performance during import.

Some of the more relevant switches are discussed in the following list:

- **Cyclic relationship during import (ConfigValues.xml)** — If you do not expect data to be cyclic, keep this value as False. This test is quite expensive. (com.tibco.cim.optimization.import.cyclictest).
- **ProcessOption** — Use this input parameter to override defaults.

Creation of Indexes for Data Sources for Database Loader

The Database Loader is primarily designed for initial data load and it does not validate data.

To ensure that data is clean before uploading it, use Data Quality and ETL tools to process the data file and clean it before uploading the data.

To quickly load larger data greater than 50K that join more than one datasource create indexes for the datasources. To do this, create the file $MQ_COMMON_DIR\<enterprise internal name>\datasource\datasourcename.idx.

**Example 1**

If the CID column of the data source DF_33969_37793_TAB table is mapped to productid and productidext is not mapped, create the index file as follows:

```
UPPER("CID")
```

**Example 2**

If the CID column of data source DF_33969_37793_TAB table is mapped to productid and CEXT is mapped to productidext, create the index file as follows:

```
UPPER("CID"),UPPER("CEXT")
```
Web Services

Best practices should also be implemented when working with the web services of TIBCO MDM. This involves the following practices:

- **Optimization of performance with web services**
- **Single sign-on and web services**
- **Use of cache with web services**
- **Concurrent web services**
- **Synchronous web services**

**Optimization of Performance with Web Services**

Reading a large number of entries, such as records and work items, in one call eventually fails with an out-of-memory error because the payload size increases.

The performance also deteriorates if a large payload needs to be transported. Such failures are sporadic, irregular and unpredictable. To create predictable and reliable performance, always build the client to scroll through the result set and set the startCount correctly. Scrolling through 100 records at a time should result in predictable and reliable performance.

**Single Sign-On and Web Services**

Be aware that a regular login takes about 30 minutes, but it could take much longer if Single Sign-on is implemented.

If you want to eliminate this overhead, you can use the Login service to login and pass the session ID in the subsequent web service calls. However, ensure that you also use the Logout service to log out after the work is complete. If you do not log out, the system eventually logs out based on inactivity timeout configuration but until that happens, a session remains open consuming memory. Left over login sessions also result in denial of service once the maximum allowed login count is exceeded.

**Use of Cache with Web Services**

Specifying ID or EXT in the UI record uses cache to find the records (find by primary).

This is significantly faster than search without both values.

**Concurrent Web Services**

Configure the number of concurrent web services allowed.

Use the com.tibco.cim.webservicelistener.maxcount parameter for this.

**Synchronous Web Services**

Synchronous web services are a good choice when a caller needs to verify the operation was successful.

It can be coupled with the in-memory workflow to provide a lightweight execution. Synchronous web services return only when the workflow (if any) fired by the web service is completed.

Do not use synchronous web services if it takes more than a few seconds to execute the service and any associated workflow. For example, do not use the import or mass update workflows which process a batch of records.
Workflows and Synchronous Web Services

Workflows initiated by the synchronous workflows do not go through the workflow queue. It means that they do not have wait periods and are not assigned to the workflow queue listener. Such workflows are in addition to the number of simultaneous workflows that can be fired through the workflow queue. Sizing a machine based on the number of workflow queue listeners creates an additional load and may negatively affect the performance.

Synchronous web services that fire workflows take more time than asynchronous workflows because the web service returns after the workflow completion. During this time, HTTP threads are held for a longer duration and the maximum concurrent HTTP listener count may be reached.

Synchronous workflows fired through the workflow may return the control to web service before the workflow has completed. This happens when large batches of data are processed by the activity and the activity splits the processing into multiple batches. Once the activity splits the batches, the activity suspends and the web service completes with workflow status as INPROGRESS.
Workflows

The Workflow topic deals with best practices implemented for workflows. It involves the following practices:

- Customization of workflows
- Error Handling
- Activities tuning
- Comparison between subflows and spawned workflows

Customization of Workflows

Customization of workflows deals with the best practices that are implemented in customizing workflows. This involves the following practices:

- Use of Java with customized workflows
- Work with transitions in customized workflows
- Work with caches in customized workflows
- Use of rules in customized workflows
- Performance considerations in customized workflows

Use of Java with Customized Workflows

Instead of using Java code in transition to perform a task that is not possible using predefined activities, create a custom workflow activity. Custom activities perform much better, are tracked using the event log, and have well-defined interfaces. Activity performance statistics are reported in JMX and the timing log.

Work with Transitions in Customized Workflows

Java class-based transitions perform much better than Java code added directly to transitions. You can experiment with inline transitions using Bsh, but you should create and compile a Java class after you have finalized the transition code.

Do not perform any database update in the transitions. Transitions do not support failover and restart. Any updates performed on the transition are not confirmed until the next activity is complete.

Work with Caches in Customized Workflows

Do not update any row in the database from the custom code because this can cause the cache to be out-of-sync.

The caching algorithm and objects stored in the cache often change to fine tune the cache. Additionally, updates may create deadlocks.

Use of Rules in Customized Workflows

Using rules to select input values for the workflow activities instead of hard-coding them makes it easier to change the inputs.

You can use the business process rules to select the rulebase to be defined in the EvaluateRulebase activity.
Performance Considerations in Customized Workflows

If you have a workflow containing more than 20 steps, break it up into smaller subflows to reduce the completion time. Longer workflows take more time to complete.

Design workflows to create work items for exceptions. Creating thousands of notification work items leads to poor performance and excessive workloads.

Use work item time out judiciously. For example, you should not typically configure Notification workflows to timeout because these type of work items, do not provide input parameters ExpiryType.

Error Handling

Use the Rollback functionality to roll back the changes made to the master data.

Activities Tuning

Activities tuning deals with the different activities that is used for tuning workflows.

Use of GetRecord for Tuning

When tuning workflows you can use the GetRecord activity which controls the depth of related records by specifying the depth and relationship names of records.

Use of Rulebases for Tuning

Use the EvaluateRulebase activity with ExecutionMode set to NoRecords if you want to execute the rulebase only once.

Review the inputs of the activities and apply inputs which restrict the scope of activities. For example, for EvaluateRulebase, parameter EvaluateChildren set as false will restrict the rulebase to root record.

Subflows

You can create subflows within a workflow. Subflows do not create new events. This means that you can freely share context, status, outputs, and errors between subflows and parent workflows.

Subflows should not set the status of the event; the event is created for the parent workflow because subflows do not have separate events. The status change is reflected in the parent workflow and the subflow should return to the parent workflow. You must complete any status changes in the parent workflow.

Subflows enable you to limit the context by explicitly mapping the context from the parent workflow to the input and output parameter of the subflow. This creates a cleaner workflow free of side effects.

Subflows can have work items and suspend the workflow. Subflow (applies to synchronous workflow only) suspension suspends the parent workflow as well. However, when such work items time out, the subflow is restarted and completed. Unless the subflow is suspended again, the parent workflow assumes that the subflow has completed and resumes. To avoid this behavior, you need to suspend the subflow by calling out suspend in the transition.

Synchronous and Asynchronous Subflows

There are two types of subflows, synchronous subflows and asynchronous subflows.

Synchronous subflows are a good choice when the subflow must return some data to the parent workflow and the parent workflow needs to wait for the subflow to finish. In subflows, errors are propagated to the parent workflow.

Asynchronous subflows are not recommended. Instead, either use synchronous subflows or spawn a new workflow. Use asynchronous subflows only when you do not need any status or data back from the subflow and when the parent does not have to wait for the subflow to complete.
Error Handling in Subflows

Any error generated in the subflow is propagated to the parent workflow if it is not handled (no error transition) in the subflow.

This is useful if you need to prevent the parent from continuing execution in these cases. However, if you have an error transition, the error is handled and not automatically propagated to the parent workflow. (Java try and catch paradigm).

Spawned Workflows

Spawned workflows are separate events and do not share the context with parent workflows.

Use spawned workflows when the parent workflow is independent of any child processes.

Spawned workflows are a good choice when the parent workflow continues after the child workflow has been initiated and does not need any feedback. TIBCO recommends using spawned workflows if you do not know how many children workflows to start or if you need to start a large number of child workflows.

Best Practices for Common Workflows

- Instead of one big workflow, split your workflow into smaller workflows and use subflows to manage smaller workflows.
- Use Java class based transitions instead of BASH based transitions.
- Always include error transitions.
- Use transitions for decisions, and custom activities for performing tasks.
- Use the TIBCO MDM Process Designer to define and maintain workflows instead of editing the XML directly. If you edit XML, always validate it using workflow schema.
- If there is more than one transition between a pair of activities, order your transitions so that the most likely one is at the top.
- Reuse state variables; keep state size small.

Best Practices for In-memory Workflows

- Processes that execute in-memory mode should be very optimal.
- Do not run long-running processes using the in-memory mode.
- Do not use activities that take a long time to execute or that import a large number of records for in-memory workflows.
- Ideally, activities and subflows for in-memory workflows should be executed synchronously.
- Typically, in-memory workflows complete in a single transaction.
- Adding subflows in async mode, spawned workflows in async mode, and checkpoints lead to the persistence of intermediate states of the workflow.
- Use the Checkpoint activity sparingly, for example, use it only when you need to check the status of the workflow at important milestones.
Deployment

The Deployment topic deals with the best practices that are implemented during TIBCO MDM deployment.

It involves the following practices:

- Task prioritization through isolation
- TIBCO MDM memory utilization
- Multiple TIBCO MDM instances
- Fallover
- Capacity planning
- Use of TIBCO MDM
- File system management
- Database maintenance
- Backup strategies
- Reduction of disk space requirements

Task Prioritization through Isolation

You can configure different priorities for different tasks based on task type and input channel.

If you want to separate different types of processing, you may want to consider the deployment illustrated in Message Prioritization to ensure that UI performance does not degrade during a batch load.

Before you consider this deployment, review how the message prioritization can be configured to change the priority of certain tasks. The deployment architecture illustrated by Message Prioritization separates the UI server but shares the rest of the components to ensure that data consistency and availability is intact.

Message Prioritization

This architecture guarantees that the UI performance is not affected when there is a large load of import or backend messages.

TIBCO MDM Memory Utilization

The TIBCO MDM JVM requires very little heap memory for the execution of the software.

The memory required for the JVM heap depends on the following:

- Number of async threads (AsynCall queue). Each thread holds many records in the heap for processing. The number of records can be as high as 5000.
• Row size of the biggest repository. The memory required for records to be held in memory depends on the row size of the largest repository.

• Directory information and metadata objects. All directory information (for example users, roles, permissions, and enterprises) and all metadata objects (for example repository, input maps, output maps, synchronization profile) are cached in a heap.

For most development environments, 1 GB of heap and 256 MB of perm size is sufficient.

For a large installation with many users and repositories, use the cache computation worksheet provided by TIBCO Support to calculate the memory requirements. As a rule of thumb, 1.5 GB heap plus 256 perm size is sufficient for most QA instances. For production instances, 2 GB to 3 GB is usually sufficient.

Use the cache computation work sheet to compute additional memory which is required for distributed cache.

Multiple TIBCO MDM Instances

Configure multiple instances with the load balance of TIBCO MDM whenever possible, until the CPU usage reaches approximately 75% with peak load.

Failover

TIBCO MDM implements a wait and retry mechanism to handle subsystem (database, JMS, file) failures.

This failover is configured based on the error codes that are returned. If the failover does not occur for a certain failure it may be due to one or more of the following reasons:

• The error is not configured for failover. You need to add the error messages to ConfigValue.xml file.
• The subsystem version is different (new Oracle version) and the error description may have changed.
• The error description may be presented in a different language.

Capacity Planning

TIBCO Engineering provides a free service to review your capacity requirements and to make recommendations on hardware.

TIBCO encourages you to use this service by contacting TIBCO Support.

Planning worksheets are available from TIBCO Support to help you size the following:

• Cache memory
• Database

Use of TIBCO MDM Studio

This topic deals with the best practices for using TIBCO MDM Studio.

Using Version Control

You can configure TIBCO MDM Studio to save models in a version control system. After making changes, check in the changed models to your version control system. If more than one developer is working on the same project, lock the project while checking out to avoid conflicts.
Partial Deployment

TIBCO MDM Studio also supports partial deployment of the model. If any changes are made to the input map or relationship, you must redeploy the entire model.

With partial deployment, you can deploy the following:

- A single repository with no changes in related components
- A single input map, output map, or synchronization format
- A single relationship and no changes in parent repository

File System Management

Beginning with TIBCO MDM 8.0 release version, file system management is now simplified.

Most data that was previously stored on the file system is now stored in a database. File system is used primarily for two important sets of files:

- Files uploaded to TIBCO MDM for attributes of type = FILE
- Temporary files generated during the processing of master data.

Temporary Directory

On the file system, most of the generated files are temporary files. These are stored under $MQ_COMMON_DIR/Temp.

The files stored under $MQ_COMMON_DIR/Temp are not required after they are used up by workflow.

Reclaiming Space by Deleting Temporary Files

In emergency situations, all the files and sub-directories under the $MQ_COMMON_DIR/Temp directory can be removed to reclaim space. Ensure that no workflows are running when deleting all files from this directory. Running workflows may need recently created temporary files. To ensure no workflows are running, shut down all TIBCO MDM instances or retain the files generated in the last 10 days.

Sent and Received Directories

While sending messages in or out of TIBCO MDM through JMS, TIBCO MDM also creates sent and received directories in $MQ_COMMON_DIR.

These files are used for tracing, reconciliation, and debugging only. The sent and received directories keep a copy of the sent or received messages and you can remove these directories without affecting any process. These files are used for tracing, reconciliation, and debugging only.

Work Directory

Do not delete files stored under the $MQ_COMMON_DIR/work folder. This folder does not grow significantly and is not expected to have many files.

If you are using a lot of attributes for which data type is defined as FILE, the files are stored under $MQ_COMMON_DIR/work. As these files are versioned, the space required to store data will increase proportional to the number of attributes defined as file and versions of such records.

Scheduling Database Maintenance

TIBCO recommends scheduling a file clean-up job to run at periodic intervals. To do this, use the tibcrontab.sh/.bat sample script found in $MQ_HOME/bin. This job deletes temporary files created by TIBCO MDM. The sample script can be edited for different retention periods.
Database Maintenance

TIBCO recommends setting up a maintenance schedule for TIBCO MDM to clear older logs and temporary files, and keep the database clean and manageable.

For more details about the purge and tibcrontab functions, see *TIBCO MDM System Administration*.

Database Purge Scheduling

The purge operation removes older record versions, changes history and workflow trace data, and helps maintain a clean database.

You can invoke the purge operation as a workflow, a command-line tool, or as a scheduled job. TIBCO recommends scheduling the data purge to run regularly.

Beginning with TIBCO MDM 8.3.1 release version, you can use interval-based purge to target records updated in a specified interval. Schedule history purges, for example, to run with an interval equal to 7 days to keep the history from growing too large. You can also use purge to remove data which is no longer needed. For example, test data which you can delete using the command-line purge.

If invoked as a workflow, the purge operation can only be run in two modes: deleting older record versions and deleting history. If invoked as a command-line tool, use the various options to trim unused test data, data sources, and repositories.

See Chapter 9, Configuring Purge in the *TIBCO MDM System Administration* for more details about the purge operation.

Performance Considerations While Running Purge

While running, purge uses significant resources from the database, cache, and text index.

While you can continue to use the system, the TIBCO MDM may appear sluggish. When data is removed from the database, the database updates all indexes.

Purge consumes a large amount of database resources and therefore may negatively affect performance. To avoid this, schedule purges to coincide with periods of low or no activity and also schedule only one purge job at a time. If you run a purge for the first time or after a long period after the last purge, it may take more time to complete than a regularly scheduled purge.

If you are running a purge for a repository (to clean up all records), performing an import for the same repository can generate resource contention and deadlocks. This can significantly slow the purge and import process.

Performance Impact of Synchronization on Purge

Synchronization of master data with external systems generates a large amount of track and trace data.

Even if the amount of incremental data being synchronized is small, TIBCO MDM tracks of the decision to skip all the records which were not synchronized. If you are doing excessive synchronization, purge the data to maintain the database size. TIBCO provides a purge job to trim the synchronization data.

Synchronization

Consider the impact of synchronization on capacity planning. The amount of synchronized data can be substantial and often requires significant disk space.

When you synchronize data (except DBDump), TIBCO MDM maintains a detailed history. This history includes the following:

- **Synchronization status**: When and with which system or partner.
- **Synchronized data**: The data for each record that was synchronized. The changed data is recorded because data can be transformed during synchronization.
Change Management

Use a version control system to track all configurations and customizations. Instead of deploying the files directly to TIBCO MDM instances, check them into the version control system, then use scripts to deploy them to target servers.

TIBCO provides a sample script to automate deployment from version control system.

To customize the components provided in the $MQ_COMMON_DIR/Work/standard folder, make a copy of the component in an enterprise-specific folder. TIBCO MDM prioritizes files within enterprise-specific folders even if they have the same name as the files under the standard folder. You can change artifacts under the standard folder with a new version or hot fix, however they are always overwritten.

Backup Strategies

TIBCO MDM data is stored in two separate data stores: files and a relational database.

Configurations

TIBCO recommends keeping all configuration files in a version control system to track changes.

Take regular backups whenever these components are changed. When creating backups, be sure to do the following:

- Ensure that all configuration files in $MQ_HOME/Config are backed up. These files affect the behavior of the TIBCO MDM application.
- Back up the ECM.ear file. This is the TIBCO MDM application that gets unpacked and deployed into the application server. TIBCO recommends backing up this file for every change made to its contents, such as visual customization, custom function, custom workflow activities and custom web services.
- Check in all the modified application server configuration files into your version control system. The configuration files for the supported application servers are:
  - Standalone.xml, for JBoss Application Server
  - /Config files for WebLogic Application Server and WebSphere Application Server
- For more information on configuration files, refer to the Installing on Application Server chapter in the TIBCO MDM Installation and Configuration Guide.
- Store all the enterprise-specific configurations in $MQ_COMMON_DIR. Back up and check these configurations into your version control system on a regular basis.

Data files

All TIBCO MDM data files are in the $MQ_COMMON_DIR/work directory which you should back up.

You do not need to back up files under the following directories:

- /temp
- /sent
- /received

Database

In addition to your company’s general backup strategy, TIBCO recommends that you perform additional steps.

The additional steps to be performed for TIBCO MDM are as follows:

- Always back up the database first before backing up $MQ_COMMON_DIR.
- Back up the entire schema for TIBCO MDM.
Reduction of Disk Space Requirements

You can reduce disk space requirements by employing a few strategies.

The strategies to reduce disk space requirements are as follows:

- Use in-memory workflows.
- Reduce the amount of data collected for each workflow step, including the intermediate documents.
- Remove the output parameter if the output document of a step is not needed.
- Modify the out-of-the-box workflow to remove parameters you do not need to extract the attributes.
- Tune the in-memory workflow by specifying appropriate values for track and trace levels.
- Review the options and configurations for workflow and rulebase tracing. You can eliminate some of these if you do not need to specify log options of EvaluateRulebase to generate error report file.
- Review the configuration of your production environment to ensure that request and response files for web service requests are not generated.
Performance Management

The Performance Management topic deals with the best practices for performance management in TIBCO MDM.

Performance within TIBCO MDM is based upon the proposed solution as well as the underlying technology. TIBCO MDM provides you with a large collection of interoperable tools that can join in chains to create processes.

If these tools are misused, performance and functional issues are likely to arise. Ensure that performance tuning with any enterprise application is always an iterative cycle. The following suggestions may assist with performance.

UI Performance

UI performance degrades if the cache is incorrectly sized. If the cache is small, the user and authentication information may be evicted, which causes reloads.

Use JMX to check the cache statistics to see how the various caches are performing. Use the appropriate cacheConfig.xml file. Three samples are supplied.

You can customize the search but adding too many search attributes to the screen clutters up the screen and make it slow to draw. UI performance can degrade and client CPU consumption can increase to 100%. Configure only the attributes that matter instead of adding all fields.

Optimizing rulebases is the key to record UI performance. Use context variables, CHANGED and RECORD_ACTION to limit the rules applicable.

Event log and inbox performance depends on default filters, including rows per page. Reduce the rows per page to mitigate the issue.

Control what you can see, especially the hierarchy and the relationships. Showing less data is better.

UI performance also depends on the browser version and network load. Your browser’s version has a huge impact on performance and note that older browsers tend to run slower. Upgrade to the latest version or use another browser to mitigate the issue.

Search Optimization

In the out-of-the-box (OOB) configuration, you do not get indexes on all the data you want to search.

After determining the common searches, determine which searches consume the maximum time. Capture the debug log for such searches and extract the query. Run the query through the database tools to determine which indexes should be created. Case-sensitive and case-insensitive searches need different indexes. You must create functional indexes to support case-insensitive searches in databases. However, indiscriminately creating indexes slows down new data acquisition.

Rulebases

Optimize the rulebases as much as possible and ensure that the rulebases are used frequently and for every record.

You need not validate all the constraints in the validation and initialization rulebases for every record. Make use of the context variables to reduce the number of validations required for a situation.

You should also remove the unused constraints.
Database Optimizations

Periodically, run the database maintenance and optimization tools to identify queries that take a longer time and add indexes to these queries.

If one repository is being used extensively, place indexes in that repository against the most searched attributes.

To reduce the database overhead, remove indexes that do not provide any value. Do not remove product out-of-the-box indexes.

Use the TIBCO MDM Purge feature to purge all the older versions of the records that are being stored but never used.

Timing Logs

If you suspect performance degradation, review timing logs for various components.

Default configurations already captured timings for slow components and you can check the log directory for a timing log file.

Timing logs capture action, SQL, activity, and rulebase which exceed the time limit defined in the configuration. If it detects a slow SQL, it is registered in the timing log. If simple SQLs start appearing in this log, it is an indication that the database is not performing well. If a particular activity shows up in the log, focus on that specific activity.

You can load timing logs to a database table using the sample script provided under bin or viewed through the sample projects in Spotfire. TIBCO MDM also generates alerts that you can view from the JMX console.

Preload

Default preload configuration preloads most of the metadata and record data. You do not need to change this for small and medium sized systems, for example, for a few 100,000 records.

Regardless of the record count, you should always preload CATALOG, ENTERPRISE.

Compute the requirement memory for preloading all the record data and if memory is sufficient, load all the record data by specifying RECORD, RECORDMAXMODVERSION, PRODUCTKEY, SYNCLOG, and SYNCRECORD.

However, if you have insufficient memory to accommodate all objects, consider preloading in the following priority:

PRODUCTKEY
RECORD
RECORDMAXMODVERSION
SYNCLOG (if synchronization performance needs to be improved)
RECORD
SYNCRECORD

Do not set up preload to a large number of records without sufficient cache because the records are evicted as the cache fills up. Using the JMX console, check the record count in cache to ascertain whether the records are being evicted. If the record count in cache is less than the number of records preloaded, stop the preload.

Import and export performance is better if data is preloaded. Database Loader does not use cached data and is not impacted.

Preload automatically preloads all the children repository metadata and records even if only the parent repository is specified for preload.
For every record, approximately three entries are created in the RECORD and RECORDMAXMODVERSION caches while two entries are created in the PRODUCTKEY repository.

**Use of JMX Console for Monitoring Performance**

Use the JMX console to review various statistics, such as workflow activities, rulebase constraint performance, and cache usage.

When the hit ratio is low for a cache, and you see a large number of misses, the cache is adequately sized.
Miscellaneous

The Miscellaneous topic deals with best practices not discussed elsewhere in this document.

It covers the following practices:

- **Debugging**
- **TIBCO ActiveMatrix BusinessWorks Integration**
- **Recover Failed Messages and Events**

**Debugging**

You can use several tools and techniques to debug TIBCO MDM.

Use the following tools and techniques to debug TIBCO MDM:

- **Workflow Trace File** The workflow trace file records the workflow path, including all input and output parameters for the activity and evaluation of transitions. You can download this file through Event Log.
- **Rulebase Trace File** The rulebase trace files record the data used for each condition evaluation and its results. The trace files are generated in the Temp directory and the file name patterns are rb*.
- **Cache and Security Debugging** Cache debugging logs all the calls made to the cache subsystem while security debugging logs all the calls to determine entitlements. Security debugging applies only when the application is running in the debug mode. Cache debugging is made when the property `com.tibco.cim.cache.debug` is set to `true` in the `ConfigValues.xml` file.
- **Support Engineer Role** This role provides tools for downloading log files and view data in TIBCO MDM tables.
- **Configurator** You can change almost all the configuration through the Configurator. Some property changes require the JMX Console to reconfigure the server. For example, use the JMX console to restart the JMS channel if the pool size of a workflow queue is changed after a configuration is deployed.
- **Logging Level** You can change the logging level without a restart. You can change most properties without restarting the server. Properties can be hot-deployed, including JMS pool sizes, scheduled job configurations, database name, and password (as configured in configValues). Use JMX for greater control over changes to properties.
- **Timing Log** The timing log records the components that are running slow. You can use the timing log provided to load this data to the database. TIBCO has developed Spotfire projects to analyze this data.
- **Data Model** The data model for TIBCO MDM data tables is available from TIBCO Support.
- **Generate Report** Use the **Generate Report** action in the Event Log screen to generate a report for TIBCO Support.
- **Documentation** All documentation is available on [https://docs.tibco.com](https://docs.tibco.com) and is updated periodically.
- **Purge** Use the purge operation to delete data and restore the state so that new tests can be run again.
- **Rollback** Use Rollback to revert the partial or complete changes to master data so that new tests can be run again. For example, you can use Rollback along with Resubmit to retest the Add Record or Modify Record action.
**TIBCO ActiveMatrix BusinessWorks Integration**

The TIBCO ActiveMatrix BusinessWorks MDM Plug-in encapsulates the TIBCO recommended connection and interaction method between TIBCO ActiveMatrix BusinessWorks and TIBCO MDM. It also provides a wrapper to the TIBCO MDM schema so that data can be easily mapped to TIBCO MDM. The plug-in provides the same functionality as the SOAP services that it wraps with some additional functionality. The plug-in presents a consistent interface that reduces the possibility of usage and configuration errors typical of SOAP service requests.

The plug-in does not implement all the SOAP services supported by TIBCO MDM. For the remaining services, use the native ActiveMatrix BusinessWorks capabilities.

The TIBCO MDM plug-in is a statement of direction for future releases of TIBCO MDM so early adoption is recommended. The plug-in is licensed along with TIBCO MDM.

**Use of TIBCO ActiveMatrix BusinessWorks with TIBCO MDM**

You can integrate TIBCO ActiveMatrix BusinessWorks with TIBCO MDM in a number of ways. For example, you can set it up to react to changes made in TIBCO MDM or so that it is triggered from external stimuli to query and make changes to TIBCO MDM as required.

**TIBCO MDM Triggering Active Matrix BusinessWorks**

Within workflows, you can send JMS messages into an Enterprise Service Bus (ESB) layer using the SendProtocolMessage activity and sending the generated inDocs. You can parse these messages and act upon them within ActiveMatrix BusinessWorks as required. This is a good method for creating a pipeline of records. You can then synchronizing those records out to partners through the ESB in a location-agnostic, standards-agnostic, and implementation-agnostic manner.

Make the call to SendMessageProtocol after a record has been confirmed, to ensure you are distributing the latest golden copy of that record. Inform the ESB when the following occurs:

- Synchronizations have occurred.
- Certain records have been changed. You can complete actions based on record states by using guarded transition states.
- Records have been deleted. Their references are removed in other external systems not controlled by TIBCO MDM.

**Active Matrix BusinessWorks Triggering TIBCO DM actions**

Use the TIBCO MDM Adapter to pick up messages from the ESB and actions performed in TIBCO MDM.

Such messages and actions include:

- Adding new records from other connected ESB systems into TIBCO MDM.
- Reading data from an external database and formatting this data for dates, names, and lengths before adding it to TIBCO MDM using the Add Record palette.
- Running periodic data cleanup in TIBCO MDM (without TIBCO Data Quality) to pull data from TIBCO MDM, make changes, and return the data. If you do not want such ActiveMatrix BusinessWorks changes to be subject to approvals, configure the Business Processes to ignore any actions that are raised by using the User Is feature.
Example Scenarios

The example mentioned explains how you can combine TIBCO ActiveMatrix BusinessWorks and TIBCO MDM using the TIBCO MDM Adapter.

The data model is not important in this situation. The following example shows how ActiveMatrix BusinessWorks, TIBCO MDM, and the TIBCO MDM Adapter work together to add and process a new record in TIBCO MDM.

- A user adds a new record in TIBCO MDM representing a new model of car.
- TIBCO MDM processes and approves this action according to the Business Requirements.
- The last action of the workflow triggers a message that is sent through JMS to the Enterprise Messaging Bus.
- This message is posted onto the EMS server of TIBCO MDM which is bridged out (using EMS scripts) to the data highway.
- An ActiveMatrix BusinessWorks service (using a JMS topic Subscriber) picks the message up and processes it.
- The ActiveMatrix BusinessWorks service adds the new car Model to the SAP-based system. The SAP system requires you to specify the car parts required for the new model.
- After you specify the car parts required for the new model, the bundle of parts is sent back to ActiveMatrix Business Works for insertion into TIBCO MDM.
- The ActiveMatrix BusinessWorks service takes the records from SAP, using the TIBCO MDM Adapter and queries MDM to ensure that the car parts are not already specified.
- The ActiveMatrix BusinessWorks service then formats the SAP-based data so that it fits better with the TIBCO MDM data model. This may involve changes such as naming conventions or combining two data collections together to form a single view.
- The ActiveMatrix BusinessWorks service then uses the Add Record Palette to add the new record into TIBCO MDM. This creates a relationship back to the new car model record.
- TIBCO MDM does not use Business Processes to raise the new work items for new records added from the ActiveMatrix BusinessWorks user. Instead, the ActiveMatrix BusinessWorks service guarantees that all business rules are met.

Recover Failed Messages and Events

If workflows started through JMS messages fail due to subsystem failures, TIBCO MDM attempts to redeliver the messages.

If JMS also fails, the message cannot be delivered and TIBCO MDM writes the message to a local disk. The messages stored in this file can be resubmitted using the message recovery script located at $MQ_COMMON_DIR/bin/msgRecovery.bat (msgRecovery.sh for Linux). Each server has its own log.