Siebel

Performance Tuning Guide

August 2021

Siebel
Performance Tuning Guide

August 2021

Part Number: F12770-09

Copyright © 2021, Oracle and/or its affiliates.

Authors: Siebel Information Development Team

This software and related documentation are provided under a license agreement containing restrictions on use and disclosure and are protected by intellectual property laws. Except as expressly permitted in your license agreement or allowed by law, you may not use, copy, reproduce, translate, broadcast, modify, license, transmit, distribute, exhibit, perform, publish, or display in any part, in any form, or by any means. Reverse engineering, disassembly, or decompilation of this software, unless required by law for interoperability, is prohibited.

The information contained herein is subject to change without notice and is not warranted to be error-free. If you find any errors, please report them to us in writing.

If this is software or related documentation that is delivered to the U.S. Government or anyone licensing it on behalf of the U.S. Government, the following notice is applicable:

U.S. GOVERNMENT END USERS: Oracle programs (including any operating system, integrated software, any programs embedded, installed or activated on delivered hardware, and modifications of such programs) and Oracle computer documentation or other Oracle data delivered to or accessed by U.S. Government end users are "commercial computer software" or “commercial computer software documentation” pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, the use, reproduction, duplication, release, display, disclosure, modification, preparation of derivative works, and/or adaptation of i) Oracle programs (including any operating system, integrated software, any programs embedded, installed or activated on delivered hardware, and modifications of such programs), ii) Oracle computer documentation and/or iii) other Oracle data, is subject to the rights and limitations specified in the license contained in the applicable contract. The terms governing the U.S. Government’s use of Oracle cloud services are defined by the applicable contract for such services. No other rights are granted to the U.S. Government.

This software or hardware is developed for general use in a variety of information management applications. It is not developed or intended for use in any inherently dangerous applications, including applications that may create a risk of personal injury. If you use this software or hardware in dangerous applications, then you shall be responsible to take all appropriate fail-safe, backup, redundancy, and other measures to ensure its safe use. Oracle Corporation and its affiliates disclaim any liability for any damages caused by use of this software or hardware in dangerous applications.

Oracle and Java are registered trademarks of Oracle Corporation and/or its affiliates. Other names may be trademarks of their respective owners.

Intel and Intel Xeon are trademarks or registered trademarks of Intel Corporation. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. AMD, Opteron, the AMD logo, and the AMD Opteron logo are trademarks or registered trademarks of Advanced Micro Devices. UNIX is a registered trademark of The Open Group.

This software or hardware and documentation may provide access to or information about content, products, and services from third parties. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to third-party content, products, and services unless otherwise set forth in an applicable agreement between you and Oracle. Oracle Corporation and its affiliates will not be responsible for any loss, costs, or damages incurred due to your access to or use of third-party content, products, or services, except as set forth in an applicable agreement between you and Oracle.

The business names used in this documentation are fictitious, and are not intended to identify any real companies currently or previously in existence.

Contents

Preface i

1 What's New in This Release 1

What's New in Siebel Performance Tuning Guide, Siebel CRM 21.8 Update 1
What's New in Siebel Performance Tuning Guide, Siebel CRM 21.5 Update 1
What's New in Siebel Performance Tuning Guide, Siebel CRM 21.4 Update 1
What's New in Siebel Performance Tuning Guide, Siebel CRM 20.12 Update 2
What's New in Siebel Performance Tuning Guide, Siebel CRM 20.3 Update 2
What's New in Siebel Performance Tuning Guide, Siebel CRM 20.2 Update 3
What's New in Siebel Performance Tuning Guide, Siebel CRM 20.1 Update 3
What's New in Siebel Performance Tuning Guide, Siebel CRM 19.7 Update 3
What's New in Siebel Performance Tuning Guide, Siebel CRM 19.1 Update 4

2 Siebel CRM Architecture and Infrastructure 5

Siebel CRM Architecture and Infrastructure 5
About Performance and Scalability 5
About Siebel Architecture and Infrastructure 6
Siebel Architecture and Infrastructure Areas for Tuning 8
About Siebel User Request Flow 9
Performance Tuning Terminology 11
Sizing Considerations for Siebel CRM Version 17.0 and Later 12

3 Tuning the Siebel Application Object Manager for Performance 17

Tuning the Siebel Application Object Manager for Performance 17
About the Siebel Application Object Manager 17
Siebel Application Object Manager Infrastructure 18
Performance Factors for Siebel Application Object Manager Deployments 19
Topology Considerations for Siebel Application Object Manager Deployments 22
Guidelines for Siebel Application Object Manager Tuning 22
Configuring Database Connection Pooling for Siebel Application Object Managers 29
Using Thread Pooling for Siebel Application Object Managers 36

4 Tuning the Siebel Server Infrastructure for Performance 39
Tuning the Siebel Server Infrastructure for Performance 39
Configuring SISNAPI Connection Pooling for Siebel Application Object Managers 39
Tuning Server Request Broker (SRBroker) 41
About Synchronous and Asynchronous Requests Forwarded by SRBroker to Batch Components 41

5 Tuning Siebel Web Client for Performance 43
Tuning Siebel Web Client for Performance 43
About Siebel Web Clients 43
Performance Factors for Siebel Web Clients 44
Guidelines for Siebel Web Client Tuning 45

6 Tuning Siebel Communications Server for Performance 49
Tuning Siebel Communications Server for Performance 49
About Siebel Communications Server 49
Session Communications Infrastructure 50
Performance Factors for Session Communications 51
Topology Considerations for Session Communications 52
Guidelines for Session Communications Tuning 53
Siebel Email Response Infrastructure 58
Performance Factors for Siebel Email Response 59
Topology Considerations for Siebel Email Response 59
Guidelines for Siebel Email Response Tuning 60

7 Tuning Siebel Workflow for Performance 63
Tuning Siebel Workflow for Performance 63
About Siebel Workflow 63
Monitoring Workflow Policies 63
Tuning Workflow Policies for Performance 66
Tuning Workflow Processes 68
Tuning Workflow Process Manager for Performance 70

8 Tuning Siebel Product Configurator for Performance 73
Tuning Siebel Product Configurator for Performance 73
Siebel Product Configurator Infrastructure 73

Performance Factors for Siebel Product Configurator 74
Topology Considerations for Siebel Product Configurator 75
Guidelines for Siebel Product Configurator Tuning 77
About Siebel Product Configurator Caching 80
Administering the Siebel Product Configurator Cache 88

9 Tuning Siebel EAI for Performance 91
Tuning Siebel EAI for Performance 91
About Siebel Enterprise Application Integration 91
Guidelines for Siebel EAI Tuning 91

10 Tuning Siebel EIM for Performance 99
Tuning Siebel EIM for Performance 99
About Siebel EIM 99
Siebel EIM Architecture Planning Requirements 100
Siebel EIM Usage Planning 102
General Guidelines for Optimizing Siebel EIM 105
Recommended Sequence for Implementing Siebel EIM Processes 105
Troubleshooting Siebel EIM Performance 108
Database Guidelines for Optimizing Siebel EIM 117
Data Management Guidelines for Optimizing Siebel EIM 126
Run Parameter Guidelines for Optimizing Siebel EIM 126
Monitoring the Siebel Server During a Siebel EIM Task 127

11 Tuning Siebel Remote for Performance 129
Tuning Siebel Remote for Performance 129
About Siebel Remote 129
Tuning Siebel Remote Server Components 130
Tuning the Siebel Mobile Web Client in a Siebel Remote Deployment 133

12 Tuning Customer Configurations for Performance 135
Tuning Customer Configurations for Performance 135
General Performance Guidelines for Customer Configurations 135
Analyzing Generated SQL for Performance Issues 137
Guidelines for Siebel Scripting 145
Guidelines for Data Objects Layer 149

Guidelines for Business Objects Layer 154
Guidelines for User Interface Objects Layer 158

13 Tuning Operating Systems and Databases for Performance 161
Tuning Operating Systems and Databases for Performance 161
Tuning Microsoft Windows for Enhanced Siebel Server Performance 161
Tuning the Siebel Server for All UNIX and Linux Operating Systems 162
Tuning the Siebel Application Interface Computer for All Applicable UNIX and Linux Operating Systems 164
Tuning the Siebel Application Interface for All UNIX and Linux Operating Systems 165
Tuning Siebel CRM for AIX 166
Tuning Siebel CRM for HP-UX 168
Tuning Siebel CRM for Oracle Solaris 169
Tuning Siebel CRM for IBM DB2 170

14 Monitoring Siebel Application Performance with Siebel ARM 173
Monitoring Siebel Application Performance with Siebel ARM 173
About Siebel Application Response Measurement 173
About Siebel ARM Parameters and Variables 174
Enabling and Configuring Siebel ARM 176
Guidelines for Converting Siebel ARM File 177

15 Analyzing Siebel ARM Data 179
Analyzing Siebel ARM Data 179
About Siebel ARM Files 179
Analyzing Siebel ARM Files Using the Siebel ARM Query Tool 180
Analyzing Siebel ARM Files Using the Siebel ARM Analyzer Tool 196

Preface

This preface introduces information sources that can help you use the application and this guide.

Using Oracle Applications

To find guides for Oracle Applications, go to the Oracle Help Center at *http://docs.oracle.com/*.

Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the *Oracle Accessibility Program website*.

Contacting Oracle

Access to Oracle Support

Oracle customers that have purchased support have access to electronic support through My Oracle Support. For information, visit *My Oracle Support* or visit *Accessible Oracle Support* if you are hearing impaired.

Comments and Suggestions

Please give us feedback about Oracle Applications Help and guides! You can send an email to: *oracle_fusion_applications_help_ww_grp@oracle.com*.

1 What's New in This Release

What's New in Siebel Performance Tuning Guide, Siebel CRM 21.8 Update

The following table lists the changes in this revision of the documentation to support this release of the software.

Topic	Description
Cache Management for Siebel Product Configurator (modified) *Configuring Session Pooling for Siebel Product Configurator* (new)	Modified and new topics. As of Siebel CRM 21.8 Update, the following product changes apply to Siebel Product Configurator: • The user interface in the Cache Administration view is enhanced to define routing of customizable products within a bundled promotion to remote instances of the Siebel Product Configurator Object Manager. • Administrators can configure session pooling for Siebel Product Configurator. This feature is intended to provide better performance when users or agents work with customizable products that are routed to remote instances of the Siebel Product Configurator Object Manager.

What's New in Siebel Performance Tuning Guide, Siebel CRM 21.5 Update

The following table lists the changes in this revision of the documentation to support this release of the software.

Topic	Description
Siebel Scripting Guidelines for Optimal Performance	Modified topic. Setting the following environment variables can improve performance when you use scripting: • USE_NEW_MM • USE_NEW_RM

What's New in Siebel Performance Tuning Guide, Siebel CRM 21.4 Update

The following table lists the changes in this revision of the documentation to support this release of the software.

Topic	Description
Sizing Considerations for Siebel CRM Version 17.0 and Later	Modified topic. As of Siebel CRM 21.2 Update, the **`applicationcontainer`** directory has been replaced by two directories, as follows: • **`applicationcontainer_external`** (for Siebel Application Interface) • **`applicationcontainer_internal`** (for all other Siebel Enterprise components)
Parameters for Configuring Siebel Product Configurator Caching	Modified topic. Updated the description of the eProdCfgNumbOfCachedCatalogs parameter.
Tuning Kernel Settings	New topic. Moved the information about configuring **`ulimit`** parameter values out of *Tuning Kernel Settings for AIX* and into *Tuning the Siebel Server for All UNIX and Linux Operating Systems*. This information applies to all UNIX or Linux operating systems, not just AIX.
Tuning Siebel CRM for IBM DB2	New topic. Provides instructions for tuning your Siebel database on IBM DB2. These steps improve the performance of Siebel workspace queries on IBM DB2, for Siebel developers using Siebel Tools or Siebel Web Tools.

What's New in Siebel Performance Tuning Guide, Siebel CRM 20.12 Update

The following table lists the changes in this revision of the documentation to support this release of the software.

Topic	Description
Siebel Architecture and Infrastructure Areas for Tuning (modified) Administering Content-Based Routing (removed)	Modified and removed topics. Siebel Constraint Engine (also known as Advanced Constraint Engine) is no longer supported for Siebel Product Configurator, as of Siebel CRM 20.12 Update. Installation and Siebel Management Console configuration options for Siebel Constraint Engine no longer apply. Siebel Product Configurator uses and will continue to use the existing iLog integration. For more information, see *Siebel Installation Guide* .

What's New in Siebel Performance Tuning Guide, Siebel CRM 20.3 Update

The following table lists the changes in this revision of the documentation to support this release of the software.

Topic	Description
Administering Content-Based Routing (deleted in 20.12)	New topic. Provides information about content-based routing, which provides a load balancing configuration for your customizable products. This feature uses Siebel Constraint Engine, which is part of the Siebel Product Configurator integration with Oracle Advanced Constraint Technology.

What's New in Siebel Performance Tuning Guide, Siebel CRM 20.2 Update

The following table lists the changes in this revision of the documentation to support this release of the software.

Topic	Description
Tuning the Anonymous Pool	New topic. Provides information about sizing considerations for the anonymous pool on the Siebel Application Interface.

What's New in Siebel Performance Tuning Guide, Siebel CRM 20.1 Update

No new features have been added to this guide for this release. This guide has been updated to reflect only product name changes.

What's New in Siebel Performance Tuning Guide, Siebel CRM 19.7 Update

The following table lists the changes in this revision of the documentation to support this release of the software.

Topic	Description
Sizing Considerations for Siebel CRM Version 17.0 and Later	New topic. Provides information about sizing considerations for Siebel CRM version 17.0 and later releases, relative to prior releases.

What's New in Siebel Performance Tuning Guide, Siebel CRM 19.1 Update

No new features have been added to this guide for this release. This guide has been updated to reflect only product name changes.

2 Siebel CRM Architecture and Infrastructure

Siebel CRM Architecture and Infrastructure

This chapter provides an overview of Oracle's Siebel CRM architecture and infrastructure and provides introductory information about tuning the Siebel CRM applications for performance and scalability. It contains the following topics:

- *About Performance and Scalability*
- *About Siebel Architecture and Infrastructure*
- *Siebel Architecture and Infrastructure Areas for Tuning*
- *About Siebel User Request Flow*
- *Performance Tuning Terminology*
- *Sizing Considerations for Siebel CRM Version 17.0 and Later*

Related Books

Siebel Deployment Planning Guide

Siebel Installation Guide

Siebel System Administration Guide

Using Siebel Tools

Configuring Siebel Business Applications

About Performance and Scalability

Every implementation of Siebel CRM is unique. Your Siebel application architecture, infrastructure, and configurations might differ depending on your business model.

Performance and scalability are defined as follows in the context of this guide:

- **Performance.** A Siebel application's ability to function, generally measured in response time or throughput.

  For example, measures of performance might include the time required to log in to the Siebel application or to display a Siebel view in the Siebel Web Client, or the volume of transactions (sometimes referred to as requests) that a server component can process in a given time period.

  Some typical inhibitors of performance are inadequate hardware, excessive network round trips, heavy customizations, and poor networking infrastructure.

- **Scalability.** A Siebel application's ability to continue to perform well as volumes increase.

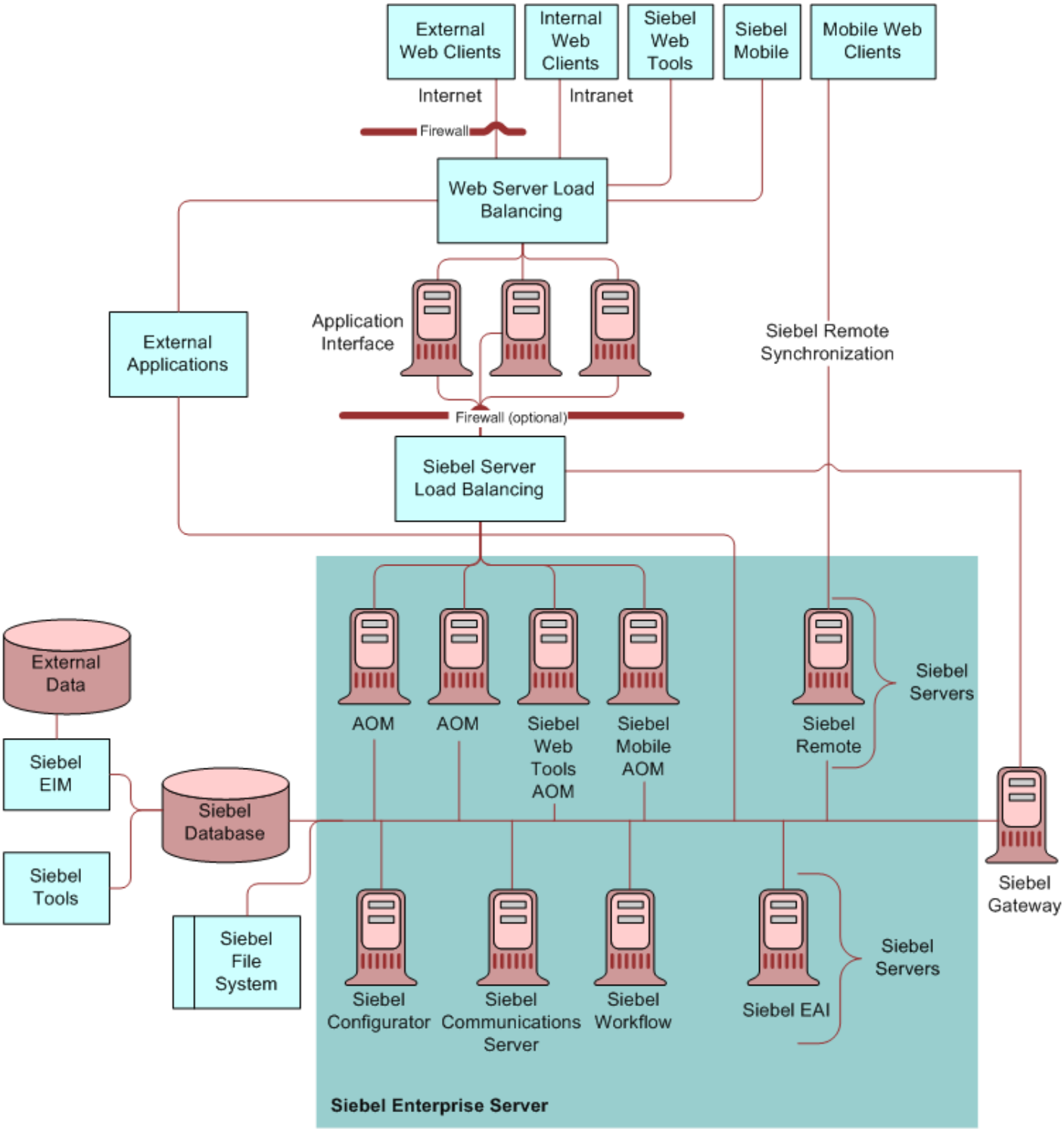
  Scalability is generally measured in hardware terms; for example, maintaining acceptable performance after adding new processors on existing computers (vertical scalability) or new Siebel Server computers (horizontal

scalability) to process an increased number of users. Some typical inhibitors of scalability are an inflexible application module structure and an inability to run parallel processes.

For more definitions of terminology related to performance and scalability, see *Performance Tuning Terminology*.

About Siebel Architecture and Infrastructure

This topic shows a generic representation of the architecture and infrastructure of a Siebel CRM deployment. Your Siebel applications might be deployed differently. For descriptions of individual entities included in this illustration, see *Siebel Deployment Planning Guide* , *Siebel System Administration Guide* , and *Siebel Installation Guide* .


External Web Clients
Internal Web Clients
Siebel Web Tools
Siebel Mobile
Mobile Web Clients
Internet
Intranet
Firewall
Web Server Load Balancing
External Applications
Application Interface
Siebel Remote Synchronization
Firewall (optional)
Siebel Server Load Balancing
External Data
Siebel EIM
Siebel Database
Siebel Tools
Siebel File System
AOM
AOM
Siebel Web Tools AOM
Siebel Mobile AOM
Siebel Remote
Siebel Servers
Siebel Gateway
Siebel Configurator
Siebel Communications Server
Siebel Workflow
Siebel EAI
Siebel Servers
Siebel Enterprise Server

Siebel Architecture and Infrastructure Areas for Tuning

The following list provides information about tuning some specific areas of the Siebel applications architecture and infrastructure.

Performance in many of these areas can be monitored and analyzed using Siebel Application Response Measurement (Siebel ARM), which is described in *Monitoring Siebel Application Performance with Siebel ARM* and *Analyzing Siebel ARM Data*.

- **Siebel Application Object Managers.** Siebel Application Object Managers are Siebel Server components that reside on a Siebel Server and support users accessing Siebel applications through the Siebel Web Client and the Siebel Application Interface, or through external applications.

  Running Siebel Application Object Manager components has significant performance and scalability implications. In general, the goal for tuning a Siebel Application Object Manager is to maximize scalability with little or no performance degradation as more users use Siebel applications.

  Although Siebel Application Object Manager components can be tuned for optimal performance, capacity for this and all other Siebel Server components is ultimately limited by Siebel Server computer resources such as CPU and memory. For more information about tuning this area, see *Tuning the Siebel Application Object Manager for Performance*.
- **Siebel Web Client.** The means for end users to access Siebel application features and data. Siebel Web Client uses a Web browser. Siebel applications like Siebel Call Center or Siebel Self Service, Siebel Web Tools, and Siebel Mobile applications on mobile devices all use the Siebel Web Client.

  The response time experienced by the Siebel Web Client end user is subject to the configuration and tuning of Siebel Enterprise elements such as the Siebel Application Object Manager, network bandwidth and latency, the Siebel Application Interface, the Siebel database, and the Siebel application configuration (represented in the Siebel runtime repository). It is also subject to local computer resources and settings, including browser settings such as those for caching. For more information about tuning this area, see *Tuning Siebel Web Client for Performance*. See also *Tuning Customer Configurations for Performance*.
- **Siebel Communications Server.** Siebel Communications Server provides an infrastructure to support several kinds of communications activities for Siebel CRM users, including session communications (such as voice calls) and inbound and outbound communications (such as email).

  Siebel Communications Server processing can affect end user response time, and might demand additional Siebel Application Object Manager resources to support user sessions. Performance and scalability is subject to third-party server configuration and capacity and Siebel Server computer resources and configuration. For more information about tuning this area, see *Tuning Siebel Communications Server for Performance*.
- **Siebel Workflow.** Siebel Workflow is an interactive environment that automates business processes such as automating escalation of events and notification of appropriate parties; routing and assigning work; processing work; and enforcing authorization and transition rules.

  Siebel Workflow processing can affect end user response time (for synchronous requests), and might demand additional Siebel Application Object Manager resources to support user sessions. Performance and scalability is subject to Siebel Server computer resources and configuration. For more information about tuning this area, see *Tuning Siebel Workflow for Performance*.
- **Siebel Product Configurator.** Siebel Product Configurator supports order management and product configuration functions for Siebel applications.

Siebel Product Configurator processing can affect end user response time (for configuration sessions), and might demand additional Siebel Application Object Manager resources to support user sessions. Performance and scalability is subject to Siebel Server computer resources and configuration. For more information about tuning this area, see *Tuning Siebel Product Configurator for Performance*.

- **Siebel Enterprise Application Integration (Siebel EAI).** Siebel EAI provides components for integrating Siebel CRM with external and internal applications, and provides inbound and outbound interfaces to and from a Siebel application. Siebel RESTful interfaces are part of the EAI framework and are also used extensively in contexts such as Siebel Management Console configuration and internal communications between Siebel CRM modules.

  Siebel EAI processing can affect end user response time (for real-time interfaces), and might demand additional Siebel Application Object Manager resources to support user sessions. Performance and scalability are subject to Siebel Server computer resources and configuration. For more information about tuning this area, see *Tuning Siebel EAI for Performance*.
- **Siebel Enterprise Integration Manager (Siebel EIM).** Siebel EIM provides components that transfer data between the Siebel database and other corporate data sources. For more information about tuning this area, see *Tuning Siebel EIM for Performance*.
- **Siebel Remote.** Siebel Remote provides components that allow Siebel Mobile Web Clients (typically operating remotely, in disconnected mode on a laptop) to connect to a Siebel Server and exchange updated data and files, a process known as synchronization. For more information about tuning this area, see *Tuning Siebel Remote for Performance*.
- **Siebel Tools.** Siebel Tools and Siebel Web Tools are integrated development environments for configuring aspects of a Siebel application, including elements in the data objects, business objects, and user interface objects layers. Siebel scripting languages are also managed in the Siebel Tools environment.

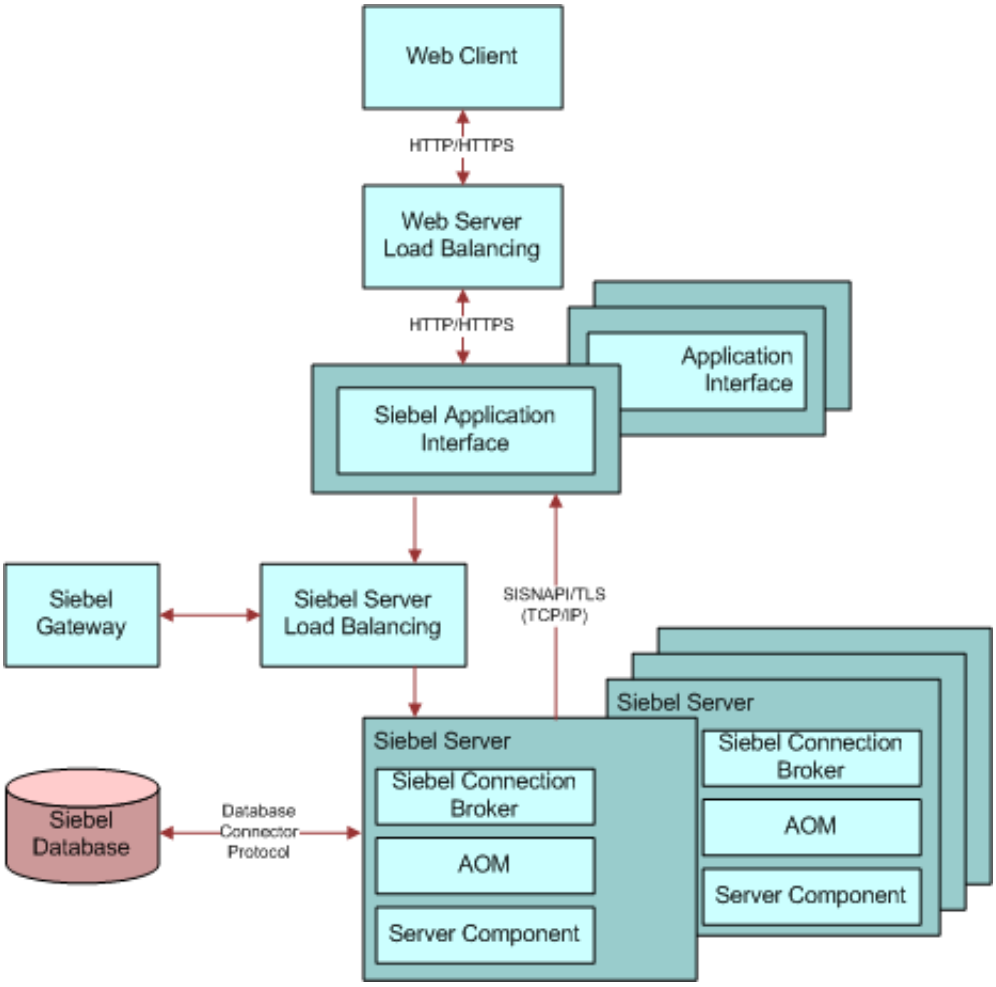
  Siebel Tools configurations and scripting play a critical role in the performance and scalability of a configured Siebel application. Customizations made through Siebel Tools partly determine the degree to which performance and scalability of a particular deployment differs from the original installation.

  Appropriate configuration optimizes operations in the Siebel database and does not add unnecessary overhead to supporting user sessions. (Siebel Tools itself does not play a role in the Siebel applications at run-time.)

  For more information about tuning this area, see *Tuning Customer Configurations for Performance*.
- **Operating systems and databases.** For more information about tuning your Microsoft Windows or UNIX operating system or an IBM DB2 database, see *Tuning Operating Systems and Databases for Performance*.

About Siebel User Request Flow

The following diagram illustrates how a user request is processed within the Siebel CRM architecture and infrastructure (generically presented), and shows potential areas for performance tuning. For a description of each portion of this data flow, see *Siebel System Administration Guide* and other relevant documents on the *Siebel Bookshelf* .



A typical Siebel CRM client request flows from the user's Siebel Web Client through the system, and back again, following the general flow outlined below.

1. A user performs an action that initiates a request. For example, the user clicks a link in the Site Map to navigate to a particular view. The request is generated by the Web browser and Siebel Web Client framework.
2. The request goes through the network, using a new or an existing HTTP connection, to the Siebel Application Interface. The request might go through a network router, proxy server, cache engine, or other mechanism.
3. The Siebel Application Interface receives the HTTP request and determines that it is a Siebel application request.
4. The Siebel Application Interface parses the HTTP message and generates a SISNAPI (Siebel Internet Session Network Application Programming Interface) message, based on the content of the HTTP message. Siebel Application Interface also parses the incoming cookie to obtain the user session ID.

   The Siebel Application Interface and Siebel Gateway work together to provide Siebel Server load balancing. When a user requests a new application connection, Siebel Application Interface sends a request to Siebel Gateway, which returns a connect string for the least-loaded Application Object Manager from among the Siebel Servers supporting that component. The user session will use this Application Object Manager.

SISNAPI is a messaging format that runs on top of the TCP/IP protocol. It is used for network communication between Siebel Servers and Siebel Application Interface. SISNAPI connections use encryption and authentication based on Transport Layer Security (TLS).

5. On the Siebel Server, the SCBroker component receives the initial request for a session and forwards it to a Siebel Application Object Manager process. Subsequent communication for the session does not use SCBroker. For more information, see *Siebel Application Object Manager Infrastructure* and related topics.
6. The Siebel Application Object Manager receives and processes the SISNAPI message sent from Siebel Application Interface.

   If a database query is needed to retrieve the information, the Siebel Application Object Manager formulates the SQL statement and sends the request to the Siebel database over a database connection. The database request goes through the database connection, using a protocol format that is specific to the database connector.
7. The database executes the SQL statement and returns data back to the Siebel Application Object Manager. The Siebel Application Object Manager forwards the message to the Siebel Application Interface that originated it.
8. The Siebel Application Interface receives the SISNAPI message, and translates it back to HTTP. The message is now in the form of Web page content.
9. The Siebel Application Interface load balancing configuration, if present, then forwards the Web page content through the original HTTP connection to the end user's Web browser.
10. The Web browser and the Siebel Web Client framework process and display the return message.

Performance Tuning Terminology

This topic provides definitions of specific terms related to performance and tuning Siebel CRM. For definitions of *performance* and *scalability*, see *About Performance and Scalability*.

For more information about some of these terms and concepts (including concurrent users and think time) in the context of tuning Siebel Application Object Manager components, see *Performance Factors for Siebel Application Object Manager Deployments*.

Term	Definition
Concurrent users	The number of application users actively using and accessing the Siebel application, or a particular element, such as a Siebel Application Object Manager process, at a particular time. It is useful to distinguish between concurrently connected users and concurrently active users: both sets of users consume memory, but only active users consume CPU (processor) resources.
Latency	Delay experienced in network transmissions as network packets traverse the network infrastructure.
Think time	The wait time between user operations. For example, if a user navigates to the Account screen and reviews data for 10 seconds before performing another operation, then the think time in this case is 10 seconds. Average think time is a critical element in performance and scalability tuning, particularly for Siebel Application Object Manager. When think time values are correctly forecasted, then actual load levels will be close to anticipated loads.
Process	An operating system (OS) process. For example, a Siebel Server component such as Siebel Application Object Manager consists of multiple OS processes, referred to as multithreaded processes.

Term	Definition
Multithreaded process (or MT server)	A process running on a multithreaded Siebel Server component that supports multiple threads (tasks) per process. Siebel Application Object Manager components run multithreaded processes that support threads.
Task	A concept for Siebel applications of a unit of work that can be done by a Siebel Server component. Siebel tasks are typically implemented as threads.
Thread	An operating system feature for performing a given unit of work. Threads are used to implement tasks for most Siebel Server components. A multithreaded process supports running multiple threads to perform work such as to support user sessions.
Response time	Amount of time the Siebel application takes to respond to a user request, as experienced by the end user. Response time is an aggregate of time incurred by all server processing and transmission latency for an operation. Response time is based on processing related to the request and to processing for other requests that might affect this user request.
Throughput	Typically expressed in transactions per second (TPS), expresses how many operations or transactions can be processed in a set amount of time.

Sizing Considerations for Siebel CRM Version 17.0 and Later

This topic provides information about sizing considerations applicable as of Siebel CRM version 17.0, relative to prior releases. The information in this topic supersedes or modifies similar information provided elsewhere in this guide.

Sizing for Siebel CRM Tier

Siebel CRM version 17.0 introduced new business agility and cloud support capabilities, including the following:

- Siebel Application Interface, a new module, replaces Siebel Web Server Extension. Siebel Application Interface is a Java module conforming to the Java EE Web Profile standard and can run on any compliant application container (Web container). It is currently certified on and includes Apache Tomcat.
- Siebel Gateway replaces the previous module by the same name. The Siebel Gateway registry, based on Apache ZooKeeper, replaces the siebns.dat file.
- Siebel CRM server modules now use application containers on Apache Tomcat and Siebel REST API calls as part of the framework for configuration using Siebel Management Console (which replaces the Siebel Configuration Wizard) and for regular operations.

Both Siebel Application Interface and Siebel Gateway support distributed topologies, using Siebel Application Interface load balancing or Siebel Gateway clustering. For more information about these and related product changes, see *Siebel Installation Guide* and *Siebel Deployment Planning Guide* .

For this release and subsequent Siebel CRM 18.x and 19.x Update releases, more memory and CPU are required compared to prior releases for Siebel Application Interface, Siebel Gateway, and Siebel Server. The following are observations made during the performance and scalability tests performed by Oracle:

- **Siebel Application Interface.** The minimum memory (RAM) required for the Java Virtual Machine (JVM) heap size was observed to be 4 GB. The CPU usage is about 1% per 1000 users and an additional 5% average CPU in case of a high login rate.
- **Siebel Gateway.** The minimum memory required for the JVM heap size was observed to be 4 GB. The CPU usage is about 1% to 2% after the Siebel Servers have started (4% maximum CPU) and an additional 5% average CPU in case of a high login rate. The same recommendations apply for each node. Example for configuration:
  - As of Siebel CRM 19.1 Update, it is recommended to set the JVM heap size minimum and maximum to 4 GB on Siebel Application Interface and Siebel Gateway.
  - On Microsoft Windows, you can set the JVM options using the tomcat8w utility. You must also specify the name of the service to which your setting applies. This utility is located in the `SIEBEL_ROOT\applicationcontainer_internal\bin` directory. Alternatively, you can set the heap size in setenv.bat to 4 GB (-Xms4096M -Xmx4096M). This batch file is located in the `SIEBEL_ROOT\applicationcontainer_internal\bin` directory.
  - On UNIX operating systems, set the heap size in setenv.sh to 4 GB (-Xms4096M -Xmx4096M). This shell script is located in the `SIEBEL_ROOT/applicationcontainer_internal/bin` directory.
- **Siebel Server.** The memory requirement has increased by 20%. CPU usage has increased by 25%, on average, across all Siebel CRM applications.

Several issues, fixes, and recommendations in these areas are reflected in corresponding alert articles available on My Oracle Support. For more information about configuring JVM options, see 2472920.1 (Article ID) on My Oracle Support. See also the Java documentation from Oracle. For information about stopping and starting the application container and related information, see *Siebel Installation Guide* and *Siebel System Administration Guide* .

The preceding information regarding CPU requirements assumes two sockets Hexacore Intel Xeon CPU X5670@2.93 GHz, and a total of 12 cores.

> **Note:** The sizing information in this topic is based on consolidation of several Siebel CRM performance and scalability tests. Actual results may vary, based on a broad range of implementation-specific factors, such as application configuration and customization, transaction mix, duration of each request, login storms, frequent incoming EAI requests requiring a new login, hardware or operating system platform, network parameters, database size, and so on. Oracle does not warrant or guarantee that customers will obtain the same or similar results. It is strongly recommended that you thoroughly test your Siebel CRM applications by performing performance and scalability testing with the expected volumes to determine your specific resource requirements. Oracle does not share the details of its internal tests. To address your requirements, it is strongly recommended that you engage Oracle Advanced Customer Support to do the sizing based on those needs.

Monitoring

The following information describes how to monitor the usage of JVM heap size and other important metrics, such as the number of concurrent threads or other performance factors. For the run-time heap usage and thread usage information, Oracle used the Java Mission Control utility and connected to the remote JVM. To enable JMX remote on UNIX operating systems, add the following arguments to the setenv.sh file for the Siebel Gateway and Siebel Application Interface installations. After enabling JMX remote, you can monitor heap usage, threads, and other metrics using the Java Mission Control dashboard (for example, you can add graphs).

```
-Dcom.sun.management.jmxremote
-Dcom.sun.management.jmxremote.port=7777
```

```
-Dcom.sun.management.jmxremote.ssl=false
-Dcom.sun.management.jmxremote.authenticate=false
-XX:+UnlockCommercialFeatures
-XX:+FlightRecorder
```

Setting Up Garbage Collection Logs

The table below describes parameters that you can optionally specify to capture garbage collection logs for troubleshooting purposes.

Parameter	Description	Example/Comments
-verbose:cg	Enables logging of garbage collection (GC) information.	N/A
-XX:+PrintGCDetails	Provides details about GC logs, such as: • Size of the young and old generation before and after garbage collection • Size of total heap • Time it takes for garbage collection to occur in young and old generation • Size of objects promoted at every garbage collection	N/A
-XX:+PrintGCDateStamps	Records each verbose GC cycle with date and timestamp format.	N/A
-Xloggc:path/filename	Specifies the file to which to redirect GC information for logging.	-Xloggc:/vol1/siebel/applicationcontainer_internal/logs/gc.log
-XX:+UseGCLogFileRotation -XX:NumberOfGCLogFiles=number_of_files -XX:GCLogFileSize=file_size	Specifies file rotation for GC logs. Using file rotation makes log analysis easier and protects the disk from space overconsumption.	-XX:UseGCLogFileRotation -XX:NumberOfGCLogFiles=10 -XX:GCLogFileSize=100M Optional; recommended
-XX:PrintHeapAtGC	Prints the heap layout before and after garbage collection is logged.	Optional
-XX:+HeapDumpOnOutOfMemoryError	Generates a heap dump when an allocation from the Java heap or the permanent generation cannot be satisfied. There is no overhead in running with this option, which is useful for production environments where the OutOfMemoryError exception takes a long time to surface.	Optional
-XX:HeapDumpPath=path	Specifies where to generate the heap dump.	-XX:HeapDumpPath=/vol1/siebel/applicationcontainer_internal/heapdumps Optional

Parameter	Description	Example/Comments

3 Tuning the Siebel Application Object Manager for Performance

Tuning the Siebel Application Object Manager for Performance

This chapter describes the structure and operation of Siebel Application Object Manager components and the tuning that might be required for optimal operation. It contains the following topics:

- *About the Siebel Application Object Manager*
- *Siebel Application Object Manager Infrastructure*
- *Performance Factors for Siebel Application Object Manager Deployments*
- *Topology Considerations for Siebel Application Object Manager Deployments*
- *Guidelines for Siebel Application Object Manager Tuning*
- *Configuring Database Connection Pooling for Siebel Application Object Managers*
- *Using Thread Pooling for Siebel Application Object Managers*

For more information about the Siebel Server and Siebel Application Object Manager infrastructure, and about the Siebel Web Client, see the following documents on the *Siebel Bookshelf* :

- *Siebel Deployment Planning Guide*
- *Siebel System Administration Guide*
- *Siebel Installation Guide*

About the Siebel Application Object Manager

The term *Siebel Application Object Manager* refers to any of several Siebel Server components that support users accessing Siebel applications through the Siebel Web Client and the Siebel Application Interface.

A different Siebel Application Object Manager component is provided for each base application for Siebel CRM, and for each installed and deployed language in which you can run your Siebel applications. For example:

- Call Center Object Manager (SSCObjMgr_enu) is the Siebel Application Object Manager for Siebel Call Center in a U.S. English environment.
- Sales Object Manager (SSEObjMgr_fra) is the Siebel Application Object Manager for Siebel Sales in a French environment.
- Self Service Object Manager (SServiceObjMgr_jpn) is the Siebel Application Object Manager for Siebel Self Service in a Japanese environment.

When configured appropriately, Siebel Application Object Manager components on your Siebel Servers can use memory and CPU resources efficiently, and can communicate efficiently with the Siebel database, the Siebel Application Interface, and other components in the Siebel Enterprise.

The multiprocess, multithreaded model for Siebel Application Object Manager components provides scalability to support deployments with a wide range of concurrent Siebel application users.

The overall performance of the Siebel Application Object Manager contributes significantly to the response time as experienced by your end users.

Siebel Application Object Manager Infrastructure

A Siebel Application Object Manager component is implemented as a *multithreaded process* on the Siebel Server. At runtime, a parent process starts one or more multithreaded processes, according to the Siebel Application Object Manager configuration.

Each Siebel Application Object Manager process can host multiple user sessions (as tasks), which in turn are implemented as threads within the process. These threads might be dedicated to particular user sessions, or they might serve as a pool that can be shared by multiple user sessions. (For each process, a few threads also start that are dedicated to performing core functions for the process.)

As more users log in to the system, additional processes can be instantiated to host these users.

- In this chapter, the term *thread* is often used interchangeably with *task*, except when you are using thread pooling. For details, see *Using Thread Pooling for Siebel Application Object Managers*.
- The terms *multithreaded server* or *MT server* are alternative terms for multithreaded process (a process that supports multiple threads). For example, the names of the Siebel Application Object Manager parameters `MaxMTServers` and `MinMTServers` refer to multithreaded processes.

Siebel Application Object Manager components, which run in interactive mode, handle processing for Siebel Web Client sessions, in which the application user interface (UI) resides. The Siebel Application Object Manager task manages Siebel business objects and data objects and performs business logic for the client session.

Generally, each Siebel Application Object Manager task starts in response to a request from a Siebel Web Client running in a Web browser, and ends when the client disconnects.

Siebel Application Object Manager Communications with Other Modules

Each Siebel Application Object Manager task uses Siebel Server infrastructure capabilities to communicate with the Siebel database, the Siebel Application Interface, and other Siebel Enterprise Server components.

The following are the major types of communication that the Siebel Application Object Manager has with other modules:

- **Communication with the Siebel database uses database connections.** Database connections can also be managed and tuned for optimal performance. You can optionally configure connection pooling for database connections.
  For information about configuring database connection pooling, see *Configuring Database Connection Pooling for Siebel Application Object Managers*.
- **Communication between the Siebel Connection Broker (SCBroker) and the Siebel Application Object Manager processes on the same Siebel Server uses mechanisms internal to the operating system.** SCBroker receives each SISNAPI (Siebel Internet Session Network Application Programming Interface) connection request from the Siebel Application Interface and forwards the connection request to a Siebel

Application Object Manager multithreaded process. Once the request has been forwarded, subsequent requests for the same user session flow directly from Siebel Application Interface to this Siebel Application Object Manager process. The request is forwarded using either a least-loaded or a round-robin algorithm, according to the setting of the SCBroker parameter `connForwardAlgorithm`.

For information about configuring SCBroker, see *Siebel Deployment Planning Guide* and *Siebel System Administration Guide* . See also *Siebel Installation Guide* .

- **Communication with the Siebel Application Interface uses SISNAPI, a messaging format that runs on top of the TCP/IP protocol.** SISNAPI connections use encryption and authentication based on Transport Layer Security (TLS). For information about tuning SISNAPI communications, see *Configuring SISNAPI Connection Pooling for Siebel Application Object Managers*.
- **Communication with other Siebel Enterprise Server components (including other Siebel Servers) also uses SISNAPI, going through Server Request Broker (SRBroker).** For information about tuning SRBroker, see *Tuning Server Request Broker (SRBroker)*.

About Tuning the Siebel Application Object Manager

Tuning activities directly or indirectly applicable to Siebel Application Object Manager components might involve any or all of the following:

- Initially configuring your system using the Siebel Management Console, as described in *Siebel Installation Guide* . You can configure the Siebel Enterprise, Siebel Server, and Siebel Application Interface, among other entities.
- Selectively enabling component groups and components on each Siebel Server. Only enable the component groups and components that you need.
- Using the Siebel Server Manager (GUI or command-line version) to tune parameters for the Enterprise Server, the Siebel Server, or the Siebel Application Object Manager component, as described in *Siebel System Administration Guide* . Alternatively, you can use using the Siebel Management Console.
- Tuning parameters for the Siebel Application Interface. You configure the Siebel Application Interface using the Siebel Management Console.

Configuration parameters are stored in the Siebel Gateway registry, as described in *Siebel Installation Guide* .

Some other chapters in this book discuss Siebel Application Object Manager tuning that relates to using other modules, such as Siebel Communications Server or Siebel Product Configurator.

Performance Factors for Siebel Application Object Manager Deployments

In planning to deploy Siebel Application Object Managers, or in troubleshooting performance for existing Siebel Application Object Manager deployments, you must consider several factors that determine or influence performance.

Factors that are central to the task of configuring the Siebel Application Object Manager are also called *performance drivers*. Performance drivers for Siebel Application Object Manager include concurrent users and average think time. Other important factors such as hardware resources will set limits on overall capacity or capacity per server.

Subsequent topics provide information and guidelines to help you achieve and maintain optimal performance and scalability.

These factors are critical in initially configuring your Siebel Application Object Managers, particularly when specifying values for the component parameters **MaxTasks**, **MaxMTServers**, and **MinMTServers**, which are discussed in *Tuning Siebel Application Object Manager Components for CPU and Memory Utilization*.

Concurrent Users

The number of concurrent users is the total number of user sessions supported at any one time. It also includes sessions supporting anonymous users. It is useful to distinguish between concurrently connected users and concurrently active users: both sets of users consume memory, but only active users consume CPU (processor) resources.

For planning and tuning purposes, you must consider concurrent users (and total users) at multiple levels:

- The entire deployment (enterprise)
- Each Siebel Server
- Each Siebel Application Object Manager component on each server
- Each multithreaded process for each Siebel Application Object Manager component

The maximum number of concurrent users per Siebel Server (assuming, for example, that a particular Siebel Server computer is dedicated to running Siebel Application Object Manager components) depends on the average think time, on your hardware resources, and on the nature of your Siebel applications deployment.

In terms of configuration, the maximum number of concurrent users for the Siebel Application Object Manager is limited by the value of the **MaxTasks** parameter. The effective maximum is also limited by the number of multithreaded processes for this Siebel Application Object Manager and by your hardware resources.

Depending on the average think time and other factors, each multithreaded process (that is, process within the Siebel Application Object Manager) typically supports a maximum of about 100 concurrent users. Configure enough multithreaded processes (using the **MaxMTServers** parameter) to support the maximum number of concurrent users required for your peak loads.

> **Note:** Some complex or specialized Object Manager components support fewer concurrent users. For example, Object Managers for Siebel eCommunications and Siebel Product Configurator typically support about 25 concurrent users. For more information about the Object Manager for Siebel Product Configurator (Siebel Product Configuration Object Manager), see *Tuning Siebel Product Configurator for Performance*.

Think Time

Think time is the average elapsed time between operations performed by users in a Siebel application. Think time includes the time required by users to conduct customer interactions, enter data into the application, and work in other applications.

The assumed think time has a direct relationship to the number of concurrent tasks that a multithreaded process can support. However, factors such as a high degree of customization or heavy use of scripting will reduce the number of tasks that each process can support.

Determine the average think time based on the usage patterns typical of your user base. After the application has been configured, perform a clickstream analysis for your key processes, and try to capture the time between the user actions

(operations) that are represented by the clicks. Also use the `list statistics` command in Siebel Server Manager to help you calculate average think time.

Consider the average time between each operation (such as clicking New) and between each overall transaction (such as performing all steps for creating a new contact). Mouse clicks do not equate to operations if they do not send a request to the Siebel application infrastructure. Calculate the overall average think time based on all of these factors.

The ratio of 100 (100 tasks per process), based on a 30-second think time, is assumed in the formula for setting the `MaxMTServers` parameter. This formula is presented in *Tuning Siebel Application Object Manager Components for CPU and Memory Utilization*.

The ratio of 100 is based on having approximately three users running operations at the exact same time (100 divided by 30 = approximately 3.3). It is generally observed that each multithreaded process can handle about three operations at the same time with minimal performance degradation.

With longer think times, one multithreaded process can support more than 100 concurrent tasks; with shorter think times, fewer tasks. For example, if the think time is 15 seconds between user operations, then about 50 tasks per process could be supported (15 times 3.3 = approximately 50, or 50 divided by 15 = approximately 3.3).

Nature of Siebel Application Deployment

Which Siebel applications and other modules that you are using, how you have configured your Siebel applications, how you have deployed your applications, and other such factors also affect Siebel Application Object Manager performance and how many concurrent users that you can support. Some of these factors include:

- Will you support employee applications (such as Siebel Call Center), customer applications (such as Siebel eService), partner applications (such as Siebel PRM), or some combination of these?
- Will you deploy your Siebel software in a global environment using multiple languages?
- What degree and what kind of application configuration changes have you made, such as those that you do using Siebel Tools? For more information, see *Tuning Customer Configurations for Performance*.

  The number of concurrent tasks that you can support varies based on the level of customization or the use of process automation for the application the Siebel Application Object Manager supports. Recommendations in this guide generally assume that operations performed are fairly standard or typical. Depending on your deployment and the modules used, some operations initiated by a single user action can be relatively complex and demand more resources than most other operations.
- Will you use specialized functionality such as offered by Siebel Product Configurator (for product configuration) or Siebel CTI (computer telephony integration for call center agents)? How will you deploy such functionality? What percentage of your user base will use such functionality? These are only examples of such specialized functionality.

Hardware Resources

Hardware resources for each Siebel Server computer, particularly the CPU (processors) and memory, are a factor in how many concurrent users can be supported for each Siebel Application Object Manager component. For example, a four-

processor computer has twice the resources of a two-processor computer and can potentially support twice as many concurrent users. Key hardware resources for Siebel Application Object Manager performance include:

- **CPU (processors).** The number of processor cores in the CPU for the server computer, and the rating of these processors.
- **Memory.** The amount of RAM, and whether it can accommodate users without excessive paging.

Disk I/O and network capacity are other important hardware factors, but they do not affect Siebel Application Object Manager tuning. They do significantly affect performance for the Siebel database and the Siebel File System, which can severely impact the overall user response time.

The total number of processor cores (alternatively, the total number of computers with a given number of processors) that you can devote to supporting Siebel Application Object Manager components will determine the total number of concurrent users.

Topology Considerations for Siebel Application Object Manager Deployments

Your Siebel applications can be deployed using a variety of topologies, or system layouts. Although Siebel Application Object Managers are only a part of the overall deployment, they play a direct and central role in supporting Siebel application users.

You must determine on how many computers you will run Siebel Server, and on how many of these you will run Siebel Application Object Manager components. In some cases, you might choose to run multiple components on the same Siebel Server. Among all of the other considerations, the resources of each computer will determine the number of computers that you require.

> **Note:** Siebel Application Object Manager components are typically the major resource consumers for your Siebel Server computers. The tuning considerations discussed in this chapter generally assume that you are not running additional components on a Siebel Application Object Manager computer that will significantly contend for available resources.

For more information about topology considerations, see *Siebel Deployment Planning Guide* .

Guidelines for Siebel Application Object Manager Tuning

Using your hardware resources optimally and configuring your system appropriately can help you to achieve your performance goals. Consider your resources and requirements carefully, and test and monitor system performance on a continual basis.

Review information presented in *Siebel System Administration Guide* and other sources. All tuning calculations must be done with some understanding of the overall system and the considerations described in *Performance Factors for Siebel Application Object Manager Deployments*.

Review the following for more information about Siebel Application Object Manager tuning:

- *Tuning Siebel Application Object Manager Components for CPU and Memory Utilization*
- *Tuning Parameters for Siebel Application Object Manager Caches*
- *Additional Parameters Affecting Siebel Application Object Manager Performance*
- *Memory Consumers in Siebel Application Object Manager*

Tuning Siebel Application Object Manager Components for CPU and Memory Utilization

This topic is part of *Guidelines for Siebel Application Object Manager Tuning*. It provides background information and guidelines for tuning your Siebel Application Object Manager components, particularly for setting values for the parameters `MaxTasks`, `MaxMTServers`, and `MinMTServers`.

Settings for these parameters determine how well the system performs under specific user load and operations. Parameter settings provide a means of controlling the server capacity through the Siebel Server infrastructure, and directly impact the overall capacity for each server.

How you set the `MaxTasks`, `MaxMTServers`, and `MinMTServers` parameters is a direct function of the factors described in *Performance Factors for Siebel Application Object Manager Deployments*, which determine the true capacity of the server.

The art of tuning Siebel Application Object Manager components is to come up with the right parameter settings that allow the server computers to host the largest number of users (scalability) with minimal impact on user response time (performance).

About MaxTasks, MaxMTServers, and MinMTServers

The Siebel Application Object Manager parameters `MaxTasks`, `MaxMTServers`, and `MinMTServers` are described below. You configure these parameters using Siebel Server Manager, which is described in detail in *Siebel System Administration Guide* .

For background information about multithreaded processes, threads, and related concepts, see *Siebel Application Object Manager Infrastructure*.

- **MaxTasks (Maximum Tasks).** Specifies the total number of tasks (threads) that can run concurrently on this Siebel Application Object Manager, for this Siebel Server. Beyond this number, no more tasks can be started to handle additional requests.
- **MaxMTServers (Maximum MT Servers).** Specifies the maximum number of multithreaded processes that can run concurrently on this Siebel Application Object Manager. Beyond this number, no more multithreaded processes can be started to handle additional requests.
- **MinMTServers (Minimum MT Servers).** Specifies the default minimum number of multithreaded processes that will start on this Siebel Application Object Manager when the parent process is started. The parent process can be started either explicitly (using Siebel Server Manager) or automatically (if the Siebel Server is started when the component state was last set to Running). Setting `MinMTServers` to 0 effectively disables the Siebel Application Object Manager component.

As more users log in, new tasks start to handle these sessions, and new multithreaded processes are started to support the additional tasks. The tasks and processes are added according to the Siebel Application Object Manager load-balancing behavior, up to the maximum number of tasks and maximum number of multithreaded processes. For more information, see *Effect of Siebel Application Object Manager Parameter Settings*.

Note: `MaxTasks`, `MaxMTServers`, and `MinMTServers` are generic parameters that apply to many different Siebel Server components. However, the specific behavior described in this chapter applies to Siebel Application Object Manager components. For more information, see *Siebel System Administration Guide* .

These parameters relate to one another in the following ways:

- `MaxMTServers` and `MinMTServers` are typically set to the same value. Doing this avoids any performance penalty for a user whose login causes a new multithreaded process to start. `MaxMTServers` *must* be equal to or greater than `MinMTServers`.

  Starting all multithreaded processes up front when the parent process is started is generally acceptable. The memory overhead for running a multithreaded process itself, apart from the overhead of its threads, is minimal.
- The ratio `MaxTasks` divided by `MaxMTServers` determines the maximum number of threads (tasks) that can run concurrently on a given multithreaded process. For more information, see the discussion of think time under *Performance Factors for Siebel Application Object Manager Deployments*.

Effect of Siebel Application Object Manager Parameter Settings

The following information illustrates how a Siebel Application Object Manager behaves given particular example settings for the `MaxTasks`, `MaxMTServers`, and `MinMTServers` parameters. More realistic examples can be found in *Formulas for Calculating Siebel Application Object Manager Parameter Values*.

For example, if `MaxTasks` = 500, and `MaxMTServers` = 5, then `MaxTasks` divided by `MaxMTServers` = 100. This means that, at most, 100 threads (tasks) can run in a multithreaded process on this Siebel Application Object Manager.

Typically, `MinMTServers` would be set the same as `MaxMTServers`. However, in this example, assume `MinMTServers` = 4. In this case, four multithreaded processes start by default, which can handle a total of 400 concurrent threads.

As users start the application on the server, the number of concurrent threads rises, and the following occurs:

- As the number of concurrent threads rises, but remains below 400, these threads are distributed among the four multithreaded processes that started by default for this Siebel Application Object Manager. This is a form of load balancing internal to the Siebel Application Object Manager component.
- If the number of concurrent threads reaches 400, and a new request is received, then a fifth multithreaded process starts for this Siebel Application Object Manager. The Siebel Application Object Manager now distributes threads among five multithreaded processes.
- If the Siebel Application Object Manager reaches 500 concurrent threads, then no more client session requests can be handled, because the existing multithreaded processes can start no more threads, and the Siebel Application Object Manager can start no more multithreaded processes. The Siebel Application Object Manager can be said to be "maxed out."

If Siebel Application Object Manager loads fall back, as users log out or session timeouts are enforced, then threads are freed up. In some cases, a multithreaded process whose threads have completed might also time out and stop running; this can happen only when `MaxMTServers` is greater than `MinMTServers`.

Guidelines for Configuring Siebel Application Object Manager Parameters

The following information provides formulas and guidelines for setting the `MaxTasks`, `MaxMTServers`, and `MinMTServers` parameters for your Siebel Application Object Manager components.

Note: All elements in the two formulas shown in *Formulas for Calculating Siebel Application Object Manager Parameter Values* vary according to your deployment. The number of concurrent users a Siebel Application Object Manager can support depends on factors such as the number of processors, session timeout settings, and the average think time.

Typically, the Siebel Application Object Manager is the only component using significant resources on the Siebel Server computer. If you run multiple server components, or run non-Siebel modules, then a Siebel Application Object Manager on this computer might support fewer concurrent threads.

Follow these general steps to determine how to set these parameter values:

- Determine the total number of concurrent users, based on the average think time and other factors discussed earlier.
- Determine the number of concurrent users that must be supported on a given Siebel Server computer running Siebel Application Object Manager. In the formulas outlined below, this is the target number of users plus the number of anonymous users, where applicable.
- Determine how many Siebel Server computers are needed to support your concurrent users. This is typically done by Oracle Advanced Customer Services or by platform vendors. Contact your Oracle sales representative to request assistance from Oracle Advanced Customer Services.

Note: Because each customer application implementation is unique, customers are encouraged to engage Oracle Advanced Customer Services for a detailed sizing review to best assess the hardware needs to support their deployment. Oracle Advanced Customer Services has experience from thousands of implementations and works closely with Oracle's Performance and Scalability team to provide sizing guidance.

- Plug your values into the formulas in this topic, then adjust the values to meet any additional criteria. In particular:
  - If your calculated value for **MaxMTServers** is not an integer, then round up the value to the nearest integer.
  - After you adjust the value of **MaxMTServers**, if your calculated ratio for **MaxTasks** divided by **MaxMTServers** is not an integer, then round up the value of **MaxTasks** until this ratio is an integer.
- Test your initial parameter settings, such as to gauge the actual number of anonymous users required, then adjust settings further as necessary.

Formulas for Calculating Siebel Application Object Manager Parameter Values

Use the following formulas for calculating parameter values for your Siebel Application Object Manager components:

- **MaxTasks** = target_number_of_users plus anon_users
- **MaxMTServers** = (target_number_of_users plus anon_users) divided by 100
- **MinMTServers** = **MaxMTServers**

As necessary, after making your initial calculations, round up **MaxMTServers** to the nearest integer, calculate the remainder (X) of **MaxTasks** divided by **MaxMTServers**, then increment **MaxTasks** by adding (**MaxMTServers** minus X). You do this to make sure that the ratio of **MaxTasks** divided by **MaxMTServers** is an integer.

The following descriptions apply to the variables and figures used in the formulas:

- target_number_of_users is the maximum number of concurrent user sessions that your Siebel Application Object Manager will support (for users who are logged in to the application).

The maximum number of concurrent users is limited by the value of the **MaxTasks** parameter for the Siebel Application Object Manager, by the number of multithreaded processes you are running (determined by **MaxMTServers** and **MinMTServers**), and, effectively, by your hardware resources.

- anon_users is the number of sessions on the Siebel Application Object Manager that are dedicated to anonymous users (threads that support users who do not log in).
  - For employee applications like Siebel Call Center, anonymous users are not supported, so this is not a factor.
  - For customer applications like Siebel eService, anonymous users might be approximately 25% of the target number of users. However, the actual figure depends on the business requirements for a specific deployment, and could be much higher, for example.
- In the **MaxMTServers** formula, the figure of 100 is the approximate maximum number of concurrent threads that each multithreaded process on the Siebel Application Object Manager can support. The number 100 is a rule of thumb. Use the number that is appropriate for your deployment.

Note: A ratio of 100 for threads per multithreaded process works for most Siebel Application Object Manager usage scenarios. However, if your deployment involves a shorter think time than 30 seconds, or a heavier than average load per thread, then each multithreaded process will support fewer concurrent threads. Conversely, a longer think time or a lighter average load will support more concurrent threads. For more information about how the ratio of threads per multithreaded process relates to think time, see *Performance Factors for Siebel Application Object Manager Deployments*.

Example Settings for Siebel Application Object Manager Parameters

Along with other factors such as think time, the calculation of **MaxTasks**, **MaxMTServers**, and **MinMTServers** depends on your assumptions for target_number_of_users and anon_users, as previously described. Example settings follow for Siebel Call Center and Siebel eService.

Example Settings for Siebel Call Center

For Siebel Call Center, assume (for example) a think time of 30 seconds, and assume that target_number_of_users = 500. For this application, anon_users is not a factor. Your parameter values would be like the following:

```
MaxTasks = 500
MaxMTServers = 500 divided by 100 = 5
MinMTServers = MaxMTServers = 5
```

Example Settings for Siebel eService

For Siebel eService, assume (for example) a think time of 30 seconds, and assume that target_number_of_users = 500. Depending on your implementation, anon_users might be about 25% of target_number_of_users (or 125). Your preliminary parameter values would be like the following:

```
MaxTasks = (500 plus 125) = 625
MaxMTServers = (500 plus 125) divided by 100 = 6.25 = 7 (round up)
MinMTServers = MaxMTServers = 7
```

Adjust the value of **MaxTasks**. The variable X = the remainder of (625 divided by 7) = 2. Increment **MaxTasks** by (**MaxMTServers** minus X): 625 plus (7 minus 2) = 625 plus 5 = 630. Therefore, the final calculations for parameter values would be like the following:

```
MaxTasks = 630
MaxMTServers = MinMTServers =7
```

Note: The settings of `MaxTasks` and `MaxMTServers` also help determine the setting of the parameter `SessPerSisnConn`, which sets the number of sessions per SISNAPI connections. For more information, see *Configuring SISNAPI Connection Pooling for Siebel Application Object Managers*.

Tuning Parameters for Siebel Application Object Manager Caches

This topic is part of *Guidelines for Siebel Application Object Manager Tuning*.

The Siebel Application Object Manager uses several caches, which affect memory usage for the Siebel Application Object Manager. Tuning Siebel Application Object Manager caches affects Siebel Application Object Manager performance and memory usage. The following are two of the major caches used by Siebel Application Object Manager that can be configured:

- SQL cursor cache
- SQL data caches

SQL Cursor Cache

The SQL cursor cache is configured using the `DSMaxCachedCursors` parameter. This cache can be enabled on multithreaded components (such as Siebel Application Object Manager) with database connection pooling. The value represents the number of SQL cursors per database connection.

For a Siebel Application Object Manager for which the Siebel Server computer is more likely to reach its memory capacity before it reaches its CPU capacity (for example, for Siebel Call Center), you can set `DSMaxCachedCursors` to a low value, even to 0. (Such an application is sometimes referred to as *memory-bound*.)

For a Siebel Application Object Manager for which the Siebel Server computer is more likely to reach its CPU capacity before it reaches its memory capacity, the default value of 16 for the `DSMaxCachedCursors` parameter might be appropriate. (Such an application is sometimes referred to as *CPU-bound*.)

In general, the value must reflect the CPU and memory resource availability on the Siebel Server computer running a particular Siebel Application Object Manager component. The trade-off in setting this parameter is that allocating memory to caching SQL cursors means that they would need to be created less often, but at a cost in memory.

The memory requirement per cursor depends on factors such as the size of the query, type of database connection, row size, and number or rows returned by the query. The utility of these cached cursors depends on the uniqueness of the queries that they represent. In general, most Siebel application queries are unique and would not benefit from reusing a cached cursor.

Generally, when more users share a database connection, through connection pooling, you would increase the number of cursors cached, provided that the required memory is available. For more information about database connection pooling, see *Configuring Database Connection Pooling for Siebel Application Object Managers*.

SQL Data Caches

The SQL data caches are configured using the `DSMaxCachedDatasetsPerProcess` and `DSMaxCachedDataSets` parameters. Two types of data caches are guided by these parameters:

- Global data cache, which is useful in most cases. This cache is governed by `DSMaxCachedDatasetsPerProcess`. The default value is 16.
- Per-connection data cache (which can be enabled with, or without, database connection pooling). This cache is governed by `DSMaxCachedDataSets`. The default value is 16.

For a CPU-bound Siebel Application Object Manager, the default values for `DSMaxCachedDatasetsPerProcess` and `DSMaxCachedDataSets` might be appropriate.

For a memory-bound Siebel Application Object Manager (for example, for Siebel Call Center), you can set `DSMaxCachedDatasetsPerProcess` and `DSMaxCachedDataSets` to a low value, even to 0.

In general, the values must reflect the CPU and memory resource availability on the Siebel Server computer running a particular Siebel Application Object Manager component. The trade-off in setting these parameters is that allocating memory to caching SQL data sets means that they would need to be created less often, but at a cost in memory. See also the discussion of the SQL cursor cache.

Additional Parameters Affecting Siebel Application Object Manager Performance

This topic is part of *Guidelines for Siebel Application Object Manager Tuning*. It provides guidelines for setting additional parameters that affect Siebel Application Object Manager performance.

- **MemProtection.** Setting the `MemProtection` parameter to `FALSE` for your Siebel Application Object Manager component might improve performance.

  When this parameter is `TRUE` (the default), each transaction issues a large number of serialized mprotect statements, the total effect of which can degrade performance on your Siebel Server computers. Setting `MemProtection` to `FALSE` can improve performance and also improve scalability.

  The `MemProtection` parameter is hidden. Click Hidden in the Component Parameters view tab to display it. Alternatively, you can set it using the command-line version of the Siebel Server Manager, as shown:

  ```
  change param MemProtection=False for comp component_alias_name server siebel_server_name
  ```

  where:

  - component_alias_name is the alias name of the Siebel Application Object Manager component you are configuring, such as SCCObjMgr_deu for the German version of Call Center Object Manager.
  - siebel_server_name is the name of the Siebel Server for which you are configuring the component.
- **DSMaxFetchArraySize.** This is a named subsystem parameter that controls the maximum number of records that can be returned by a business component in ForwardBackward mode. It does not restrict the number of records returned for ForwardOnly cursors. By default, `DSMaxFetchArraySize` has a value of 0. When this parameter is set to 0, the Siebel Application Object Manager initializes the parameter to 10,000. This means that a maximum of 10,000 records can be returned by a business component in ForwardBackward mode.
- **DSPreFetchSize and DSMaxCursorSize.** Set these parameters *only* for Siebel implementations on IBM DB2 for z/OS. For more information about setting these parameters, see *Implementing Siebel Business Applications on DB2 for z/OS* . For all other databases, these parameters must be set to -1.

Memory Consumers in Siebel Application Object Manager

This topic is part of *Guidelines for Siebel Application Object Manager Tuning*.

In addition to the caches described earlier, this topic discusses major memory consumers in Siebel Application Object Manager components. For more information about some of these topics, see *Tuning Customer Configurations for Performance*.

Note: Specific performance and scalability testing is required to determine the size of Siebel Application Object Manager memory required.

- **Database client libraries.** Database client libraries have their own caches, caching metadata, connections, cursors, and data. Some of these caches can be reduced in size by using Siebel database connection pooling, described in *Configuring Database Connection Pooling for Siebel Application Object Managers*.
- **Scripts.** A script defined on a business component, applet, or business service is loaded into Siebel Application Object Manager memory when the script is first invoked.

  For Siebel eScript, garbage collection is performed according to settings that are optimized for each release in order to use server memory and other resources appropriately.
- **Heavy configurations.** Performance is affected when an application is heavily configured, because the memory for the Siebel Application Object Manager increases proportionally in this case.
- **Session timeouts.** Higher session timeout values mean more active sessions on the server at a time, therefore more memory being used. Lower session timeout values can mean more frequent logins.
- **Navigation pattern.** Numerous scenarios that can be used to navigate in the application can make using global caches ineffective.
- **Users per Siebel Application Object Manager.** More users per Siebel Application Object Manager means more sharing of global resources between the users. While the amount of memory used *per user* on this Siebel Application Object Manager is less, more memory is used overall.
- **Number of applets on views.** More applets configured on views means more business components will be needed at a time, hence higher overall memory usage.
- **PDQ size.** The list of items in the PDQ (predefined queries) list are maintained on the server for the current business object. The higher the number of items in this list, the more memory it consumes. The size of PDQ strings also determines memory usage.

Configuring Database Connection Pooling for Siebel Application Object Managers

This topic describes database connection configuration options for Siebel Application Object Managers, particularly database connection pooling. It contains the following information:

- *About Database Connections for Siebel Application Object Managers*
- *Database Connection Pooling Usage Guidelines*
- *Configuring Pooling for Default Database Connections*
- *Configuring Pooling for Specialized Database Connections*

Note: Each customer must determine whether their RDBMS has a sufficient total number of database connections for their needs. The total number of available connections is subject to limitations deriving from RDBMS and operating system platforms and other factors. Before you configure connection pooling, verify how many database connections are available for use by the Siebel Application Object Manager. RDBMS performance and usage of database connections by non-Siebel components are outside the scope of this guide. In general, if the available memory resources for database connections for Siebel CRM are sufficient without your using database connection pooling, then it is recommended not to use this feature.

About Database Connections for Siebel Application Object Managers

This topic is part of *Configuring Database Connection Pooling for Siebel Application Object Managers*. It provides an overview of database connections for Siebel Application Object Manager components, including nonpooled connections and pooled connections. Subsequent topics provide guidelines and instructions for configuring different types of database connection pooling.

About Nonpooled Database Connections

If you do not pool database connections, then the number of database connections corresponds to the number of Siebel Application Object Manager sessions; that is, database connections are not pooled. No special Siebel Application Object Manager configuration is required for using nonpooled database connections. When no pooling is configured, database connections are closed when the user session terminates.

- **Nonpooled default database connections.** With nonpooled database connections, during session login, a database connection is established, using the user's database credentials. (When an external authentication system is used, such as LDAP, the user's database credentials might not be the same as the user's Siebel credentials.)

  This database connection becomes bound to the session, and is the default database connection used for read, write, update, and delete operations.

  In this book, such connections are called *default database connections*. These connections can alternatively be pooled, as described later in this topic.
- **Nonpooled specialized database connections.** If, during a session, specialized functionality is invoked that uses the external transaction management capabilities of the Siebel Application Object Manager, then a second database connection is opened for this specialized use.

  This database connection is also bound to the session, and is used for all externally controlled transactions performed by the session. Siebel EAI components are an example of specialized code that does external transaction management.

  In this book, such connections are called *specialized database connections*. These connections can alternatively be pooled, as described later in this section.

About Pooled Database Connections

Optionally, you can configure your Siebel Application Object Manager components to support pooling for the same two types of database connections described previously for nonpooled database connections:

- **Pooled default database connections.** These database connections can be pooled to support sharing (multiplexing), persistence, or both features.

- Shared connections support multiple user sessions at the same time, by multiplexing (sharing) database operations for multiple SQL statements over the same database connection. Using shared connections can support more users with a given number of connections.
- Persistent connections are pooled, but are not necessarily shared. Using persistent connections can enhance performance by avoiding the cost of creating database connections. All shared connections are also persistent connections.

For more information, see *Database Connection Pooling Usage Guidelines* and *Configuring Pooling for Default Database Connections*.

- **Pooled specialized database connections.** These database connections are dedicated to a single session at a time, and serve a specialized purpose. Pooling such connections provides persistence, but such connections are never shared. By persistently pooling these connections, you enhance performance by avoiding the cost of creating connections.

Note: If you configure pooling for default database connections, but not for specialized database connections, then each specialized database connection is closed when the transaction that required it completes.

For more information, see *Database Connection Pooling Usage Guidelines* and *Configuring Pooling for Specialized Database Connections*.

Database Connection Pooling Usage Guidelines

This topic is part of *Configuring Database Connection Pooling for Siebel Application Object Managers*.

Observe the following guidelines to help you determine if you must use database connection pooling, or to guide your deployment of connection pooling.

For more information about configuring the Siebel Application Object Manager parameters for database connection pooling mentioned in this topic, see *Configuring Pooling for Default Database Connections* and *Configuring Pooling for Specialized Database Connections*.

When to Consider Using Database Connection Pooling

Consider implementing database connection pooling if, and *only if*, one or more of the following is true for your deployment:

- The RDBMS cannot support the number of dedicated user connections that you would require if using nonpooled database connections. Pooling default database connections for shared use can reduce the number of connections you require.
- Memory resources are scarce on the Siebel Server computer on which the Siebel Application Object Manager is running. Pooling default database connections for shared use can reduce Siebel Application Object Manager memory requirements per concurrent user.
- Your deployment uses external authentication such as LDAP (that is, authentication other than database authentication), and creating new connections is slow on the database server. Pooling database connections can speed up login or other operations by providing persistent pooling, whether or not connections are also shared.
- You use a Siebel Server component that requires frequent logins for special-purpose processing. Pooling database connections to provide persistent connection pooling (not sharing) can provide a significant benefit for such components.

- For Siebel Product Configurator, if you are using the component Siebel Product Configuration Object Manager (alias eProdCfgObjMgr_jpn in a Japanese locale), then it is highly recommended to configure persistent connection pooling. For more information about Siebel Product Configurator, see *Tuning Siebel Product Configurator for Performance*.

  > **Note:** A separate Application Object Manager component is provided for each installed and deployed language in which you can run your Siebel applications. For example, Call Center Object Manager for French is SCCObjMgr_fra.

- For some other components, such as EAI Object Manager (when run using intermittent sessions, where SessionType in the SOAP header = None), it might also be helpful to configure persistent connection pooling

> **Note:** If session caching is configured for a component (by setting the parameter `ModelCacheMax`), then persistent connection pooling might provide little benefit. For example, session caching is typically configured for Workflow Process Manager. For more information about session caching for Siebel Workflow, see *Caching Sessions*.

Guidelines for Using Database Connection Pooling

If you decide to use database connection pooling, then observe the following guidelines:

- Start with a low ratio of `MaxTasks` divided by `MaxSharedDbConns`, such as 2:1.

  > **Note:** If you plan to use a ratio higher than 3:1, then it is recommended that you consult Oracle Advanced Customer Services. Contact your Oracle sales representative to request assistance from Oracle Advanced Customer Services.

- If you have short (aggressive) average think times in your user scenarios, then use a smaller ratio of `MaxTasks` divided by `MaxSharedDbConns`. Longer think times can support larger ratios.

  For a 30-second think time, do not use a ratio higher than 10:1. For a 15-second think time, do not use a ratio higher than 5:1. Other factors discussed in this topic will also determine practical limits. For more information about think time, see *Performance Factors for Siebel Application Object Manager Deployments*.

- Minimize long-running queries. Long-running queries can affect overall performance and must be minimized or avoided. If the current connection pool is blocked by a long-running query from one user session, then other user sessions will use a different pool. However, a long-running query can continue to run on the RDBMS even if the user who initiated it has stopped the browser.

  When you are using database connection pooling, it is critical to optimize database access in your environment. If long-running queries occur, then monitor the overall database response time. To achieve a satisfactory response time, use a small ratio of `MaxTasks` divided by `MaxSharedDbConns`.

  You can minimize or avoid long-running queries by providing the Cancel Query option for users, by adjusting indexes to include fields that can be sorted or queried by end users, by configuring the application user interface so that nonindexed fields are not exposed, and by training users to avoid operations that would perform long-running queries. For more information about how indexing can affect Siebel application performance, see *Managing Database Indexes in Sorting and Searching*. For more information about configuring the Cancel Query option, see *Siebel Applications Administration Guide* .

- Consider the cost of creating database connections. This cost differs based on your authentication method.

If your deployment uses database authentication, then a database connection is created for each login, for authentication purposes. Afterwards, this connection is released to the shared connection pool, if the total number of connections is less than the maximum. Or, if the pool is full, then the connection is closed (terminated). Therefore, even when the pool is full and connections are available, new connections are still created temporarily for each new session login. These connections must be accounted for in determining the allocation of database connections.

With external authentication, however, you can use persistent connection pooling to reduce the cost of creating database connections. With persistent connection pooling, database connections, once created, are persistent, though such connections might or might not be shared. For pooled default database connections where connections are persistent but not shared, set `MaxSharedDbConns` to equal `MaxTasks` minus 1. For more information about authentication options, see *Siebel Security Guide* .

- In order to configure connection pooling for specialized database connections, you must also configure pooling for default database connections, as follows:
  - If you do not configure connection pooling for shared database connections (`MaxSharedDbConns` equals -1 or 0), then each specialized database connection, once created, is dedicated to the user session. The value of `MinTrxDbConns` is ignored.
  - If you configure connection pooling for shared database connections (`MaxSharedDbConns` has a value greater than 0, and less than `MaxTasks`), then specialized database connections are not dedicated to user sessions. Such connections are handled according to the setting of `MinTrxDbConns`:
    - If `MinTrxDbConns` equals -1 or 0, then, after the transaction that required it has ended, each specialized database connection is closed (deleted).
    - If `MinTrxDbConns` has a value greater than 0, then, after the transaction that required it has ended, each specialized database connection can return to the connection pool.
- Siebel CRM does not support MTS or Oracle multiplexing features.

Configuring Pooling for Default Database Connections

This topic is part of *Configuring Database Connection Pooling for Siebel Application Object Managers*.

Default database connections can be used by most Siebel Application Object Manager operations.

Configuring Parameters for Pooling Default Connections

The following information describes how to enable or disable pooling for default database connections using the parameters `MaxSharedDbConns` (`DB Multiplex - Max Number of Shared DB Connections`) and `MinSharedDbConns` (`DB Multiplex - Min Number of Shared DB Connections`).

- To enable connection pooling, set `MaxSharedDbConns` and `MinSharedDbConns` to positive integer values (at least 1) that are no higher than `MaxTasks` minus 1. A connection will be shared by more than one user session once the number of sessions within the multithreaded process exceeds the maximum number of shared connections allowed per process.
  - `MaxSharedDbConns` controls the maximum number of pooled database connections for each multithreaded process.
  - `MinSharedDbConns` controls the minimum number of pooled database connections that the Siebel Application Object Manager tries to keep available for each multithreaded process.

The setting of `MinSharedDbConns` must be equal to or less than the setting of `MaxSharedDbConns`. Depending on your Siebel Application Object Manager usage patterns, set these to the same value or set `MinSharedDbConns` to a lower value (if you determine this to be helpful in conserving database connection resources).

- To configure persistent and shared database connection pooling, set `MaxSharedDbConns`, using the appropriate ratio of `MaxTasks` divided by `MaxSharedDbConns`. Depending on the ratio, a greater or lesser degree of sharing will be in effect. Start with a 2:1 (or smaller) ratio for `MaxTasks` divided by `MaxSharedDbConns`. With this example ratio, two user tasks will share the same database connection.
- To configure persistent but nonshared database connection pooling, set `MaxSharedDbConns` = `MaxTasks` minus 1.
- To disable connection pooling, set `MaxSharedDbConns` and `MinSharedDbConns` to -1 (this is the default value).

`MaxSharedDbConns` and `MinSharedDbConns` are defined per Siebel Application Object Manager component, on an enterprise basis (these parameters are included in named subsystems of type InfraDatasources). The database connections these parameters control are not shared across multithreaded processes. The actual maximum number of database connections for each multithreaded process is determined by the ratio `MaxSharedDbConns` divided by `MaxMTServers`.

Note: `MaxSharedDbConns` and `MinSharedDbConns` work differently than `MinTrxDbConns`, which specifies the number of shared specialized database connections available for each multithreaded process. For details, see *Configuring Pooling for Specialized Database Connections*.

Example Configuration for Pooling Default Connections

Assume, for example, the following parameter settings:

```
MaxTasks = 500
MaxMTServers = 5
MinMTServers = 5
MaxSharedDbConns = 250
MinSharedDbConns = 250
```

With these settings, the Siebel Application Object Manager component can support a maximum of 500 tasks (threads). Those 500 tasks would be spread over five multithreaded processes, each having 100 tasks. Each multithreaded process would have a maximum of 50 shared database connections, each of which would serve up to two tasks.

How Pooled Default Connections Are Assigned

When a user logs into the Siebel Application Object Manager, a database connection is established to authenticate the user, then discarded (closed) once the database or external authentication system authenticates the user. After successful authentication, the Siebel Application Object Manager's connection manager checks the connection pool for SQL statements. If this connection pool is empty, then the connection manager adds a connection.

Each time a user initializes an SQL statement, the connection manager checks the connection pool for available connections. The connection manager reserves a connection for the SQL statement in one of the following ways:

- If a connection is available to handle the SQL statement, the connection manager assigns this connection to execute the SQL statement.

  The connection manager reserves this connection until execution of the SQL statement terminates. On termination of the SQL statement, the connection manager releases the connection. At the end of a user session (due to a user logging out or a session timeout), the connection manager checks the connection used by the user session. If this connection is not referenced by other user sessions and the number of connections available in the pool of database connections exceeds the value specified by `MinSharedDbConns`, the connection manager closes this connection and releases it from the pool of database connections.

- If no connection is available in the connection pool, then the connection manager creates a new connection and assigns it to execute the SQL statement.

  The connection manager continues to add connections to the connection pool until the number of connections in the connection pool equals `MaxSharedDBConns`.

Configuring Pooling for Specialized Database Connections

This topic is part of *Configuring Database Connection Pooling for Siebel Application Object Managers*.

Specialized database connections, which are not shared, are used primarily by specialized Siebel components such as Siebel EAI that need transactions to span multiple Siebel Application Object Manager operations. These connections are used for operations that use `BEGIN TRANSACTION` and `END TRANSACTION`.

Configuring Parameters for Pooling Specialized Connections

The following information describes how to enable or disable specialized connection pooling using the parameter `MinTrxDbConns` (`DB Multiplex - Min Number of Dedicated DB Connections`).

- `MinTrxDbConns` controls the minimum number of specialized database connections for each multithreaded process. The connections are not created until they are needed. The minimum value is thus the minimum size of the pool of specialized connections once all of the connections in the pool have been created.
  - To enable specialized connection pooling, set `MinTrxDbConns` to a positive integer value (at least 1). You must also configure pooling for default database connections.
  - To disable specialized connection pooling, set `MinTrxDbConns` to minus 1 (this is the default value).
- There is no explicit limit to the maximum number of specialized connections. However, effectively, there cannot be more specialized connections than sessions. On average, there will be many fewer connections than sessions.

`MinTrxDbConns` is defined per Siebel Application Object Manager component, on an enterprise basis (this parameter is included in named subsystems of type InfraDatasources). The database connections that this parameter controls are not shared across multithreaded processes. The actual minimum number of specialized database connections for each multithreaded process is what you specify as the value for `MinTrxDbConns`.

> **Note:** `MinTrxDbConns` works differently than `MaxSharedDbConns` and `MinSharedDbConns`. `MaxSharedDbConns` and `MinSharedDbConns` specify the number of shared database connections available for all Siebel Application Object Manager processes, while `MinTrxDbConns` specifies the number of specialized database connections per Siebel Application Object Manager process. For more information, see *Configuring Pooling for Default Database Connections*.

Example Configuration for Pooling Specialized Connections

Assume, for example, the following parameter setting, in addition to those described in *Example Configuration for Pooling Default Connections*:

```
MinTrxDbConns = 5
```

With this setting, each multithreaded process would have a minimum of five specialized database connections. If all five multithreaded processes are running on this Siebel Application Object Manager, then there would be a minimum of 25 specialized connections for this Siebel Application Object Manager.

How Pooled Specialized Connections Are Assigned

When the Siebel Application Object Manager starts up, the specialized connection pool is empty. When a request is made to start a transaction, the Siebel Application Object Manager requests a database connection from the specialized connection pool. If one is available, then it is removed from the pool and given to the session for that session's exclusive use.

When the transaction completes (such as by being committed or canceled), the session returns the specialized connection to the pool. If the pool already contains more than the number of connections specified by `MinTrxDbConns`, then the specialized connection is closed; otherwise, it is retained in the pool.

Scenario for Assigning Pooled Specialized Connections

Assume, for example, that `MinTrxDbConns` is set to 2. Specialized connections will be handled as follows:

- User 1 starts Transaction 1. The specialized connection pool is empty, so a new connection is created. Once Transaction 1 completes, this connection is returned to the pool.
- User 2 starts Transaction 2. If Transaction 1 is still running, then a new specialized connection is created. If Transaction 1 completed, then Transaction 2 uses the first database connection.
- When two specialized connections have been created, they will remain in the pool until the Siebel Application Object Manager terminates.

Using Thread Pooling for Siebel Application Object Managers

Optionally, you can configure your Siebel Application Object Manager components to use thread pooling. Enabling Siebel Application Object Manager thread pooling as described in this topic both pools and multiplexes (shares) multiple tasks across threads.

Using Siebel Application Object Manager thread pooling can improve performance and scalability on multiple-processor computers that are under heavy load; for example, computers using eight or more processors that are running at more than 75% capacity.

> **Note:** Siebel Application Object Manager thread pooling is not recommended for smaller server computers or for computers that run under a relatively lower capacity.

About Thread Pooling for Siebel Application Object Manager

The pool size per multithreaded process for a Siebel Application Object Manager is determined by the combined settings of the parameters `UseThreadPool`, `ThreadAffinity`, `MinPoolThreads`, and `MaxPoolThreads`.

Siebel Application Object Manager thread pooling reduces some of the system resource usage devoted to creating and closing session threads, as users log in and log out or are timed out. As when you are not using thread pooling, session threads are created as needed as session requests demand. However, instead of being closed when a session terminates, they are released to a pool, where they become available for use by a subsequent session.

When `ThreadAffinity` is `FALSE` (the default), threads are multiplexed, as can be done with certain types of database connections or SISNAPI connections. At any given time, each thread can be dedicated to one or more user session (task).

> **Note:** Using thread pooling introduces its own overhead, such as in task context-switching. For this reason, it is strongly recommended not to try to pool threads without also multiplexing them. That is, do not set both `UseThreadPool` and `ThreadAffinity` to `TRUE`.

Configuring Siebel Application Object Manager Thread Pooling

To use thread pooling, you set the following parameters on the Siebel Application Object Manager:

- `UseThreadPool` equals `TRUE` (default is `FALSE`)
- `ThreadAffinity` equals `FALSE` (default is `FALSE`)
- `MinPoolThreads` equals min_number_threads_in_pool (default is 0), where min_number_threads_in_pool represents the minimum number of threads in the Siebel Application Object Manager thread pool.
- `MaxPoolThreads` equals `MinPoolThreads` (default is 0)

> **Note:** You must specify a value for `MaxPoolThreads` that is equal to or greater than the value of `MinPoolThreads`.

To determine an appropriate value for `MinPoolThreads` and `MaxPoolThreads`, start slowly, monitor system performance, then introduce more multiplexing, as appropriate for your deployment. For example, start with a formula like this (based on two tasks per thread):

```
MinPoolThreads = MaxPoolThreads = (MaxTasks/MaxMTServers) divided by 2
```

Subsequently, you can increase this to five tasks per thread, using this formula:

```
MinPoolThreads = MaxPoolThreads = (MaxTasks/MaxMTServers) divided by 5
```

For example, if `MaxTasks` equals 525, and `MaxMTServers` equals 5, this would be:

```
MinPoolThreads = MaxPoolThreads = (525 divided by 5) divided by 5 = 105 divided by
5 = 21
```

Or, if `MaxTasks` equals 725, and `MaxMTServers` equals 5, this would be:

```
MinPoolThreads = MaxPoolThreads = (725 divided by 5) divided by 5 = 145 divided by
5 = 29
```

> **Note:** Adjust values for think time, as necessary. If you cut your think time value in half, then double the number of threads in the pool.

4 Tuning the Siebel Server Infrastructure for Performance

Tuning the Siebel Server Infrastructure for Performance

This chapter describes the structure and operation of Siebel Application Object Manager components and the tuning that might be required for optimal operation. It contains the following topics:

- *Configuring SISNAPI Connection Pooling for Siebel Application Object Managers*
- *Tuning Server Request Broker (SRBroker)*
- *About Synchronous and Asynchronous Requests Forwarded by SRBroker to Batch Components*

For more information about the Siebel Server and Siebel Application Object Manager infrastructure, and about the Siebel Web Client, see the following documents on the *Siebel Bookshelf* :

- *Siebel Deployment Planning Guide*
- *Siebel System Administration Guide*
- *Siebel Installation Guide*
- *Siebel Security Guide*

Configuring SISNAPI Connection Pooling for Siebel Application Object Managers

This topic provides information about how to manage SISNAPI (Siebel Internet Session application programming interface) connections for your Siebel Server.

SISNAPI, a messaging format that runs on top of the TCP/IP protocol, is used for network communication between Siebel Application Object Manager and Siebel Application Interface. SISNAPI connections use encryption and authentication based on Transport Layer Security (TLS).

Each multithreaded process for a Siebel Application Object Manager component uses a pool of SISNAPI connections managed by the Siebel Application Interface. The process multiplexes (shares) many client sessions over each connection.

Each client session request opens a new connection and adds it to the pool, until the number of connections defined by the value of the parameter `SessPerSisnConn` have been created. Subsequent requests are then multiplexed over the existing pooled connections. SISNAPI connections persist until one of the following events occur:

- The Siebel Application Interface process terminates.
- The Siebel Application Object Manager component terminates.
- The value of the parameter `SISNAPI Connection Maximum Idle Time` (alias `ConnIdleTime`) is reached. For more information on this parameter, see *Siebel System Administration Guide* .
- Your firewall times out the connection.

Multiplexing traffic over a set of SISNAPI connections helps to reduce the number of open network connections.

The SISNAPI connection pool size per multithreaded process for a Siebel Application Object Manager is determined by the combined settings of **MaxTasks**, **MaxMTServers**, and **SessPerSisnConn**. In general, the maximum number of SISNAPI connections created would approximately equal **MaxTasks** divided by **MaxMTServers**, then divided by **SessPerSisnConn** (resolved to an integer).

SessPerSisnConn determines how many sessions can be multiplexed over a single SISNAPI connection. By default, **SessPerSisnConn** is set to 20 for Siebel Application Object Manager components. This figure is suitable for most deployments and generally does not need to be changed. You might set this differently, depending on how you have calculated think time. For example, for Siebel EAI, this value must be set to a much lower value, such as in the range of 3 to 5. For more information, see *Performance Factors for Siebel Application Object Manager Deployments*. For more information about configuring **MaxTasks** and **MaxMTServers**, see *Tuning Siebel Application Object Manager Components for CPU and Memory Utilization*.

> **CAUTION:** If you set **SessPerSisnConn** to a value of 1, then you must set the parameter **ConnIdleTime** to a value other than -1 (its default value). Otherwise, this combination of settings might cause the Siebel Application Interface to stop running.

The number of actual SISNAPI connections per multithreaded process for the Siebel Application Object Manager is determined by the following formula:

```
(MaxTasks divided by MaxMTServers) divided by SessPerSisnConn = SISNAPI_conn_per_process
```

In this formula, SISNAPI_conn_per_process represents the number of SISNAPI connections per multithreaded process.

Assume, for example, the following parameter values:

```
MaxTasks = 600
MaxMTServers = 5
SessPerSisnConn = 20
```

In this case, the formula would be:

```
(600 divided by 5) divided by 20 = 120 divided by 20 = 6
```

Or, if **MaxTasks** = 540 and **SessPerSisnConn** is set to 18, then the formula would be:

```
(540 divided by 5) divided by 18 = 108 divided by 18 = 6
```

In each case, six SISNAPI connections will be created and pooled for each Siebel Application Object Manager multithreaded process, from each Siebel Application Interface. Each Siebel Application Interface can potentially have its own set of six connections, so the maximum total number of connections into a Siebel Application Object Manager process is six times the number of Siebel Application Interface instances. In the first example above, 20 sessions would be multiplexed over each connection. In the second example, 18 would be multiplexed over each connection.

> **Note:** In general, it is recommended that the figures used for the above formula be tailored to produce an end result that is an integer. To achieve this, you might need to modify how you define **MaxTasks**, **MaxMTServers**, and **SessPerSisnConn**.

Some Object Manager components are not Siebel Application Object Manager components. Configuration issues for such components differ from those applicable to Siebel Application Object Manager components. For information about the EAI Object Manager, see *Tuning Siebel EAI for Performance*.

Tuning Server Request Broker (SRBroker)

The Server Request Broker (SRBroker) component routes requests between Siebel Server components, such as from a Siebel Application Object Manager to a batch component. SRBroker also handles requests between batch components. SRBroker is used whether the components run on the same computer or on different computers.

Server requests originating with a Siebel Application Object Manager component always go to the SRBroker component to determine what to do with the request:

- If the destination component is running on the same Siebel Server, then SRBroker passes the request to this component. If multiple instances of the destination component are running, then SRBroker passes the request to each component instance in a round-robin fashion.
- If the destination component is not running on the same Siebel Server, then SRBroker passes the request to SRBroker running on another computer. If the destination component runs on multiple Siebel Servers, then SRBroker passes the request to each server in round-robin fashion.

The default parameter values for SRBroker work well for most deployments. If necessary, adjust the value of the `MaxTasks` parameter (the default value is 100). `MaxTasks` determines the maximum number of SRBroker threads (tasks) that can run on the Siebel Server. As necessary, set `MaxTasks` to a value equal to the number of batch components running on the Siebel Server, plus the number of Siebel Servers in the enterprise, plus 10 (for overhead).

`MaxMTServers` and `MinMTServers` determine the maximum and minimum number of SRBroker multithreaded processes that can run on the Siebel Server. Each multithreaded process can run a maximum of `MaxTasks` divided by `MaxMTServers` threads. Default values for `MaxMTServers` and `MinMTServers` must be set to 1. Increasing this value will not increase performance, and will not have any benefit.

CAUTION: Setting `MaxTasks` parameter values for SRBroker components in such a way that does not meet the above guidelines might result in request failures.

For more information about the SRBroker and SRProc components, see *Siebel System Administration Guide* .

About Synchronous and Asynchronous Requests Forwarded by SRBroker to Batch Components

The SRBroker component forwards synchronous and asynchronous requests to batch components like Workflow Process Manager, running on the same server or on a different server. Asynchronous requests can use either of two modes, Asynch mode or DirectDb mode. (Using DirectDb mode is generally recommended.) A target component must be available in order for the request to succeed. Load balancing and appropriate sizing of batch components can help ensure that your server resources are effectively used and that all requests are fulfilled.

The Server Request Processor (SRProc) component, running on each applicable server, is also used in handling some requests. For requests made using DirectDb mode, SRProc writes the requests into the `S_SRM_REQUEST` table. SRProc running on the same or a different server also forwards requests to target batch components.

Requests might in some cases cause a target batch component to reach its maximum number of tasks, as set using `MaxTasks`. In such cases, the requests are handled as follows, depending on the type of request and on the setting of the `HonorMaxTasks` parameter on the target batch component:

- For synchronous requests or asynchronous requests using Asynch mode, if the target batch component has reached the maximum number of tasks, then the component queues the request in memory for processing when tasks become available. Queuing such tasks minimizes the potential of request failure on the batch component due to the `MaxTasks` value having been reached. However, in some cases, requests might remain in the queue in the process memory, delaying processing.
- For an asynchronous request made using DirectDb mode (which is generally recommended), the SRProc component on the same or a different server forwards the request from the `S_SRM_REQUEST` table to the target batch component. If the target batch component has reached the maximum number of tasks, and the parameter `HonorMaxTasks` is `FALSE` (the default setting), then the component queues it in memory for processing when tasks become available. Queuing such tasks minimizes the potential of request failure on the batch component due to the `MaxTasks` value having been reached. However, in some cases, requests might remain in the queue in the process memory, delaying processing.
- For an asynchronous request made using DirectDb mode, the SRProc component on the same or a different server forwards the request from the `S_SRM_REQUEST` table to the target batch component. If the target batch component has reached the maximum number of tasks, and the parameter `HonorMaxTasks` is `TRUE`, then it is unable to accept the request. However, SRProc on the same or a different server can forward the request from the `S_SRM_REQUEST` table to a target batch component for processing when tasks become available. Requests that are queued in the database are subject to less risk due to process failure or hangs than requests that are queued in memory.

> **Note:** Settings for `MaxTasks`, `MaxMTServers`, `MinMTServers`, and `HonorMaxTasks` are used by the Application Object Manager components for Siebel Mobile applications, as with other Siebel applications. Load balancing is supported for Siebel Mobile applications. To support synchronization for a Siebel Mobile disconnected application that participates in load balancing, you must set `HonorMaxTasks` to `TRUE` for the component the application uses for synchronization.

Set `MaxTasks` and `HonorMaxTasks` as appropriate for each batch component, according to your usage cases, request volumes, and overall topology. Consider that, if batch components sometimes experience hang issues, then it might be undesirable to queue requests in component memory.

For help resolving any component failure or hang issues, create a service request (SR) on My Oracle Support. Alternatively, you can phone Global Customer Support directly to create a service request or get a status update on your current SR. Support phone numbers are listed on My Oracle Support.

See also *Tuning Workflow Process Manager for Performance*.

Related Books

Siebel Deployment Planning Guide

Siebel System Administration Guide

Siebel Business Process Framework: Workflow Guide

Siebel Mobile Guide: Connected

Siebel Mobile Guide: Disconnected

Article 477818.1 (Article ID) on My Oracle Support

5 Tuning Siebel Web Client for Performance

Tuning Siebel Web Client for Performance

This chapter describes configuration options that affect the performance and throughput of the Siebel Web Client and provides guidelines for tuning the client to achieve and maintain optimal performance and scalability. It contains the following topics:

- *About Siebel Web Clients*
- *Performance Factors for Siebel Web Clients*
- *Guidelines for Siebel Web Client Tuning*

The following chapters in this guide also relate to Siebel Web Client performance:

- For performance considerations for Application Object Manager, see *Tuning the Siebel Application Object Manager for Performance*
- For performance considerations related to configuring Siebel applications, see *Tuning Customer Configurations for Performance*
- For performance considerations related to Siebel Application Interface and to server operating systems, see *Tuning Operating Systems and Databases for Performance*

For more information, see the following documents on the *Siebel Bookshelf* :

- *Siebel Deployment Planning Guide*
- *Siebel Installation Guide*
- *Deploying Siebel Open UI*
- *Siebel System Administration Guide*
- *Siebel Security Guide*
- *Siebel Remote and Replication Manager Administration Guide*
- *Desktop Integration Siebel Agent Guide*
- Certifications tab on My Oracle Support

About Siebel Web Clients

A Siebel Web Client is a browser-based client for Siebel CRM applications, which access data and services by way of Siebel modules running on server computers. The Siebel Web Clients allow users to access information managed by Siebel applications.

The main Siebel CRM client type covered in this book is the Siebel Web Client. This client runs in a browser on the end user's client computer, and provides the Web-based user interface of the Siebel CRM architecture, Siebel Open UI. The browser must support HTML5 and be W3C-compliant. Additional client functionality is supported through software such as Desktop Integration Siebel Agent.

Using HTTP, the browser connects to a Siebel Application Interface over the Internet. Through the Siebel Application Interface, the Siebel client connects to a Siebel Application Object Manager component on a Siebel Server, which

executes Siebel application business logic and accesses data. Data is accessed from the Siebel database and can also be accessed from other data sources using virtual business components and a variety of integration methods.

For more information about the Siebel Web Client and about Desktop Integration Siebel Agent, see *Siebel Installation Guide* , *Siebel System Administration Guide* , and *Desktop Integration Siebel Agent Guide* . For information about deploying Siebel applications in the browser, see *Deploying Siebel Open UI* . See also the Certifications tab on My Oracle Support.

Performance Factors for Siebel Web Clients

Some performance considerations for Siebel applications involve processing or tuning activities on servers only, and are described elsewhere in this guide. Some of these factors do not directly affect Siebel client performance, while others, such as the SQL queries in use, do affect Siebel client performance. This chapter highlights some of the factors most directly related to the performance of the Siebel Web Client. For additional information about these topics, see applicable documents on the *Siebel Bookshelf* .

About Supporting Multiple Siebel Modules

Depending on the specific Siebel CRM applications that you are using and how you deploy them, these applications have different requirements and characteristics.

- Employee applications, such as Siebel Call Center.
- Customer applications, such as Siebel Self Service or Siebel eSales.

All Siebel CRM applications use Siebel Open UI. All Siebel applications have many architectural elements in common. Multiple applications can use the same Siebel runtime repository. Each application uses its own Siebel Application Object Manager component. You might need to define, configure, and test multiple instances of each application to verify that your requirements are met in each usage scenario.

The performance of your Siebel applications will vary according to how you have configured the applications using Siebel Tools, custom browser scripts, or other methods. See *Following Configuration Guidelines*.

Client performance will also vary according to which Siebel modules that you deploy. The performance of the Siebel client is highly dependent on feature functionality. Therefore, performance characteristics of Siebel modules will differ.

Some modules add special processing requirements. For example, Siebel CTI uses the Communications Session Manager (CommSessionMgr) component, and supports the communications toolbar and displaying screen pops in the client. Server and local resources support this functionality.

Supporting users who are dispersed in offices around the country or around the world introduces special deployment factors that can affect performance.

About Local Computer Resources

The resources available on each user's local computer must meet or exceed the recommendations outlined in the Certifications tab on My Oracle Support. Some performance enhancement measures depend directly on the available resources.

Guidelines for Siebel Web Client Tuning

Consider your hardware resources and requirements carefully prior to rolling out configuration changes, to make sure that business requirements have been met and the client performs as anticipated in the design phase.

Review guidelines presented elsewhere in this book, particularly in *Tuning Customer Configurations for Performance* and in other relevant documents on the *Siebel Bookshelf* .

Ongoing testing and monitoring of your system performance is strongly recommended, because database characteristics change over time. To maintain an optimally performing system over time, you must plan for growth or other changes in your deployed application.

Activities that you perform to achieve performance and scalability goals might include the following:

- Adjusting your system topology
- Configuring the Siebel application in Siebel Tools or through other methods
- Configuring Siebel Server components, particularly the Siebel Application Object Manager
- Adjusting hardware resources available on local computers
- Adjusting operating system settings on server or client computers
- Adjusting Siebel Application Interface settings or network settings
- Setting user preferences for Siebel applications

This topic contains the following information:

- *Providing Sufficient Capacity for Siebel Application Interface and the Network*
- *Testing Performance for Web Clients*
- *Providing Sufficient Client Hardware Resources*
- *Tuning System Components*
- *Following Configuration Guidelines*
- *Specifying Static File Caching on the Siebel Application Interface*
- *Configuring the Data Block Size of HTTP Requests for the Siebel Developer Web Client*
- *Managing Performance Related to Message Notification*

Providing Sufficient Capacity for Siebel Application Interface and the Network

This topic is part of *Guidelines for Siebel Web Client Tuning*.

Make sure that your Siebel Application Interface is appropriately configured to meet your performance requirements. See also *Specifying Static File Caching on the Siebel Application Interface*.

Make sure that your network capacity (bandwidth) meets your performance requirements, and that your environment supports full duplex Ethernet. In addition, it is highly recommended that you install all Siebel Servers on the same subnet. For more information, review the general requirements for Siebel Enterprise Server installation and configuration in *Siebel Installation Guide* .

The following factors impact decisions regarding network bandwidth:

- **Application configuration.** Large, complex views will require bigger templates, more controls, and more data to be sent from the Siebel Application Interface to the client. If bandwidth is an issue, then it is important to consider user scenarios to determine the optimal size and layout per view.

  For example, for views used frequently, reduce the number of fields displayed. Rather than assuming a specific set of columns to display in list applets, provide the minimal number of columns required in the base configuration and let users decide which columns to display. For more information, see *Tuning Customer Configurations for Performance*.
- **Login.** The first login is generally the most expensive operation for the Siebel client. The client infrastructure caches the main components of the application on first login. Subsequent logins require far fewer resources. Cached objects remain on the client computer until the cache is cleared or a new version of the application configuration is available.

Testing Performance for Web Clients

This topic is part of *Guidelines for Siebel Web Client Tuning*.

Oracle Advanced Customer Services offers general guidance based on information known about the characteristics of the configured Siebel application. Contact your Oracle sales representative to request assistance from Oracle Advanced Customer Services.

Customer testing is advised in any case, because assumptions are based on general data. Actual experience can vary due to use-case scenarios. Select a few of the most common scenarios: those that represent the highest percentage of activity. Collect the overall bandwidth used.

Make sure that you are testing with warm views (those already visited and cached) if this is how the application will be used most of the time, assuming that users log in and use the application for four to eight hours at a time before logging off and starting a new session.

When you estimate bandwidth required for several users sharing a low-bandwidth connection, consider use cases carefully and plan accordingly. Rather than planning for worst-case network-performance scenarios (such as all users simultaneous pressing the Enter key or visiting a new view), it is likely that very few users are actually using the network at the same moment. For more information about performance monitoring, see *Monitoring Siebel Application Performance with Siebel ARM*.

Web client performance also depends on browser performance. For general information about this issue, see *Deploying Siebel Open UI* .

Providing Sufficient Client Hardware Resources

This topic is part of *Guidelines for Siebel Web Client Tuning*.

For best client performance for employee applications, provide sufficient or generous hardware resources to your end users. Requirements can vary according to your deployment.

The more memory that is available on your client computers, the greater the number of views that can be cached. For more information, see *Specifying Static File Caching on the Siebel Application Interface*.

The speed of the processors (CPU) on your client computers will affect how quickly the Siebel application user interface is rendered.

For best performance for Siebel applications, it is generally recommended to use recent versions of browsers that support all applicable standards in your testing and in your user deployments. More recent versions often include fixes and performance enhancements.

For Siebel client hardware and other platform requirements and recommendations, see the Certifications tab on My Oracle Support. For information about deploying Siebel applications in the browser, see *Deploying Siebel Open UI* .

Tuning System Components

This topic is part of *Guidelines for Siebel Web Client Tuning*.

Overall end user performance is affected by the processing on the client as well as by everything from the Siebel Application Interface to the Siebel database and back. Explore all applicable areas for opportunities to improve overall performance.

Most performance tuning involving Siebel Server components focuses on the Siebel Application Object Manager. For more information, see *Tuning the Siebel Application Object Manager for Performance*.

You can use Siebel ARM to monitor transactions through the Siebel infrastructure. Note areas which require substantial time and resources, and investigate them further for tuning opportunities.

For example, a custom configuration might have resulted in an unintentionally complex SQL statement for which the database instance has not been optimized. Small configuration adjustments in Siebel Tools, or database tuning, can improve both client performance and application scalability on Siebel Servers. For more information about Siebel ARM, see *Monitoring Siebel Application Performance with Siebel ARM*.

Following Configuration Guidelines

This topic is part of *Guidelines for Siebel Web Client Tuning*.

For best performance by the Siebel client, carefully assess all customer configuration initiatives. All configuration changes must be justifiable in terms of the cost of configuration itself, and in terms of possible impact on performance.

Some application administration tasks might also affect application performance, and must also be carefully assessed.

Follow guidelines in *Tuning Customer Configurations for Performance* or in other guides on the *Siebel Bookshelf* .

Specifying Static File Caching on the Siebel Application Interface

This topic is part of *Guidelines for Siebel Web Client Tuning*.

Siebel application elements are stored in the browser cache, to improve performance when users log in or access Siebel views.

Note: When measuring performance, take caching into account. For example, performance is better where application elements are retrieved from the browser cache than when the same elements are not cached and must be retrieved from the system.

Browser caching behavior is also subject to Siebel Application Interface settings for static file caching. Appropriate settings allow files that are rarely updated, such as image files, JavaScript files, or style sheet files, to be cached on the browser. Caching static files reduces network utilization and enhances Siebel Web Client response time.

For more information about tuning Siebel Application Interface, see *Tuning Operating Systems and Databases for Performance*.

For Siebel client hardware and other platform requirements and recommendations, see the Certifications tab on My Oracle Support. For information about deploying Siebel applications in the browser, see *Deploying Siebel Open UI* .

Configuring the Data Block Size of HTTP Requests for the Siebel Developer Web Client

This topic is part of *Guidelines for Siebel Web Client Tuning*.

For the Siebel Developer Web Client, Siebel CRM uses HTTP requests and responses to exchange information between the browser and the internal Web server that is part of the siebel.exe executable program. You can change the maximum length of the HTTP request data that is passed. By default, the maximum limit for the HTTP request is 524288 bytes (512 KB), which is typically sufficient. However, you can change this limit by configuring the `DataBlockSize` parameter in the `[Siebel]` section of the application configuration file, such as uagent.cfg for Siebel Call Center. If you change the value of this parameter, and restart the Siebel Developer Web Client, then the siebel.exe memory usage will reflect whatever value you set.

Managing Performance Related to Message Notification

This topic is part of *Guidelines for Siebel Web Client Tuning*.

Employee applications such as Siebel Call Center include the message notification feature. The display of messages in the Siebel client requires network resources and local resources on the client computer to continually update the displayed text.

- If your deployment does not require it, then turn off the message notification feature to save processing resources.
- If some of your users require message notification, then you can specify that users will be able to turn it off by choosing Tools, then User Preferences, and then Message Notification.

For more information about message notification, see *Siebel Applications Administration Guide* .

6 Tuning Siebel Communications Server for Performance

Tuning Siebel Communications Server for Performance

This chapter describes some issues that affect the performance and throughput of selected functionality for Siebel Communications Server modules, and provides guidelines for tuning these modules to achieve and maintain optimal performance and scalability. Functionality described in this chapter includes session communications (typically, Siebel CTI or Siebel Chat) and Siebel Email Response. Other communications-related modules are not described.

This chapter contains the following topics:

- *About Siebel Communications Server*
- *Session Communications Infrastructure*
- *Performance Factors for Session Communications*
- *Topology Considerations for Session Communications*
- *Guidelines for Session Communications Tuning*
- *Siebel Email Response Infrastructure*
- *Performance Factors for Siebel Email Response*
- *Topology Considerations for Siebel Email Response*
- *Guidelines for Siebel Email Response Tuning*

For more information about Siebel Communications Server, see the following documents on the *Siebel Bookshelf* :

- *Siebel CTI Administration Guide*
- *Siebel Chat Guide*
- *Siebel Email Administration Guide*
- *Siebel System Administration Guide*
- *Siebel Deployment Planning Guide*

About Siebel Communications Server

Siebel Communications Server provides an infrastructure to support several kinds of communications activities for Siebel application users. For session communications performance tuning information, see *Session Communications Infrastructure* and subsequent topics. For Siebel Email Response performance tuning information, see *Siebel Email Response Infrastructure* and subsequent topics.

- **Session communications.** Supports interactive (session) communications for contact center agents who use the multichannel communications toolbar to make or receive voice calls using computer telephony integration supported by CTI middleware or other third-party products or who use Siebel Chat.
- **Inbound communications.** Supports integrating with third-party email servers and processing inbound email (when using Siebel Email Response).

- **Outbound communications.** Supports integrating with a variety of third-party communications systems, such as email servers or wireless messaging providers, to send outbound communications.

  Supports agents sending email replies using Siebel Email Response.

  Supports the Send Email and Send Fax commands for Siebel application users. (Send Page is also available, but uses the Page Manager server component.)

  Supports users sending outbound communications content (email, fax, or page) using communication requests. Requests can be created and submitted either programmatically or manually through a user interface described in *Siebel Email Administration Guide* .

  Many Siebel modules invoke business service methods through workflows to send outbound communications.

Session Communications Infrastructure

Session communications refers to using Communications Server components to enable contact center agents or other users to handle interactive communications work items. For example, Siebel CTI supports this capability, enabling agents to handle voice calls using the communications toolbar.

It is important to understand the infrastructure that supports session communications in order to prevent or address performance issues in this area.

Session communications performance is addressed in this topic and in:

- *Performance Factors for Session Communications*
- *Topology Considerations for Session Communications*
- *Guidelines for Session Communications Tuning*

Key Siebel Server Components

Session communications are supported in the Siebel Server environment primarily by the following components:

- **Communications Session Manager (CommSessionMgr).** This server component manages interactive communications work items such as voice calls.
- **Siebel Application Object Manager.** This server component manages application sessions for end users who use the Siebel Web Client, including users who handle communications work items (agents). Interactive communication requests from agents typically go through Siebel Application Object Manager. For more information about Siebel Application Object Manager, see *Tuning the Siebel Application Object Manager for Performance*.
- **Server Request Broker (SRBroker).** This server component handles communications between the Siebel Application Object Manager and certain other Siebel Server components, including CommSessionMgr.

  For example, when a call center agent using Siebel CTI makes a call through the communications toolbar, the request goes from Siebel Application Object Manager to CommSessionMgr by way of SRBroker.

  SRBroker is used whether CommSessionMgr runs on the same computer as the Siebel Application Object Manager, or on a different computer. For more information about such scenarios, see *Topology Considerations for Session Communications*. For more information about SRBroker, see *Tuning Server Request Broker (SRBroker)*.

Other Siebel Server Components

You might also be using the following components in your Siebel Server environment and communications infrastructure:

- **Communications Configuration Manager (CommConfigMgr).** Optionally used to cache communications configuration data.
- **Communications Inbound Receiver (CommInboundRcvr).** For more information, see *Siebel Email Response Infrastructure*.
- **Communications Inbound Processor (CommInboundProcessor).** For more information, see *Siebel Email Response Infrastructure*.
- **Communications Outbound Manager (CommOutboundMgr).** Sends outbound email or other types of messages.

Third-Party Product Modules

You might be using third-party product modules such as CTI middleware, communications driver, and communications configuration; routing products; predictive dialers; interactive voice response modules; email servers; fax servers; and so on. For more information about working with third-party CTI middleware products, see *Siebel CTI Administration Guide* . For more information about working with third-party email servers, see *Siebel Email Administration Guide* .

Performance Factors for Session Communications

This topic describes factors that drive or affect performance for session communications deployments.

Depending on your deployment, your agents can be handling phone calls (Siebel CTI), email messages (Siebel Email Response), chat messages, work items of other communications channels, or a combination of these.

- **Inbound calls processed per hour.** The number of inbound calls (or other types of work items) processed per hour (or some other time period) by your communications infrastructure.
- **Outbound calls processed per hour.** The number of outbound calls processed per hour (or some other time period) by your communications infrastructure. (For outbound predictive dialer calls, only the calls that are answered and processed by Communications Server are relevant here.)
- **Number of user communications actions per minute (load).** The average number of communications-related user actions per minute, and the average think time between such user actions. Communications-related actions typically refers to actions performed using the communications toolbar.

  Longer think times mean less load on the Siebel database and the Siebel Server. Think time is an important factor in overall system load. Estimation of think time must approximate actual user usage. For more information about think time and Siebel Application Object Manager tuning, see *Tuning the Siebel Application Object Manager for Performance*.
- **Number of concurrent communications users (agents).** The number of concurrent users of session communications features (typically, contact center agents). This figure will be some percentage of the total number of concurrent users on the Siebel Application Object Manager.

You also need to understand how agents work with these features, the average number of inbound and outbound work items per agent, and how these factors relate to your organization's service goals. Some agents receive a large number of work items from ACD queues or initiate a large number of work items. Supervisors or other users can be defined as agents but might receive only escalated work items, for example.

For more information about concurrent users and Siebel Application Object Manager tuning, see *Tuning the Siebel Application Object Manager for Performance*.

- **Volume of customer data.** The total volume of customer data. Data volume affects how quickly data can be retrieved for various purposes, such as to perform lookups for screen pops, route work items, or populate the customer dashboard. In many cases, data volume directly affects response times seen by agents. The volume of data must be realistic and the database must be tuned to reflect real-world conditions.

  These and many other factors (such as the average call time, average time between calls for an agent, and so on) will affect system performance as experienced by contact center agents. An agent will be concerned with general response time, screen pop response time, and other perceived measures of performance.

Third-Party Product Considerations

Review information presented in applicable third-party documentation for any requirements that affect your deployment. For example:

- Some CTI middleware software might place limitations on the number of agents that can be served at a single contact center site.
- Integration with ACD queues, predictive dialers, or other modules can affect your configurations, affect network traffic, or have other impacts.
- The capacity of your telephony link (between the ACD switch and the CTI middleware) can affect performance.

Topology Considerations for Session Communications

Generally, Siebel Communications Server components for session communications, such as CommSessionMgr, run on the same Siebel Server computers as those running Siebel Application Object Managers. In some cases, however, you must run CommSessionMgr on a different computer than the Siebel Application Object Managers. These options are described in detail in this topic.

CTI middleware generally runs on servers located at each contact center facility.

Running CommSessionMgr on Siebel Application Object Manager Computers

Generally, Siebel Communications Server components for session communications must be run on the same Siebel Server computers as those running Siebel Application Object Managers. Such a topology allows the Siebel Application Object Manager load-balancing mechanism to indirectly balance Communications Server load. CommSessionMgr loads are fairly light and do not, in themselves, present a reason to run this component on dedicated computers.

Set the `Enable Communication` parameter to `TRUE` for all Siebel Application Object Managers to which your agents will connect. If you are using Siebel Server load balancing, then all Siebel Application Object Managers to which requests are distributed must be configured the same way.

Running CommSessionMgr on Dedicated Computers

Sometimes you *must* run CommSessionMgr on a different computer than the Siebel Application Object Manager components.

CommSessionMgr must run on the same computer where the communications driver for your CTI middleware is running. If your driver requires a particular operating system, then you must install Siebel Server and run CommSessionMgr on a computer of this operating system. Communications drivers are required to be able to run on one of the operating systems supported for Siebel Server, as described in the Certifications tab on My Oracle Support.

If your Siebel Application Object Manager components (such as Call Center Object Manager) run on computers using a different operating system, then you set several parameters in the communications configuration, including `CommSessionMgr` and `RequestServer`, in order to designate the computer where CommSessionMgr is running. All communications session requests from a Siebel Application Object Manager supporting users for this communications configuration will be routed to the CommSessionMgr component on the dedicated computer. For related information, see *Tuning the CommSessionMgr Component*. For more information about these parameters, see *Siebel CTI Administration Guide* .

Guidelines for Session Communications Tuning

Using your hardware resources optimally, and configuring your system appropriately, can help you to achieve your performance goals. Consider your resources and requirements carefully, and test and monitor system performance on a continual basis.

Review information presented in *Siebel CTI Administration Guide* , *Siebel System Administration Guide* , relevant third-party documentation, and other sources.

Activities that you can perform to achieve performance and scalability goals include, but are not limited to, the following:

- Adjusting your system topology. For more information, see *Topology Considerations for Session Communications*.
- Configuring the Siebel Application Object Manager component.
- Configuring CommSessionMgr and related components.
- Modifying communications configurations, communications driver settings, and so on. Many of the activities described in the topics that follow are of this nature.

To maintain an optimally performing system over time, you must plan for changes in the volume of incoming communications, number of users, and so on. Verify that your CTI middleware can support an anticipated increase in the volume of incoming communications and in the number of users. Then additional hardware might be required to run more Siebel Application Object Manager components and CommSessionMgr components to support the increase in volume of communications and in number of users.

This topic contains the following information:

- *Tuning the Siebel Application Object Manager Component*
- *Tuning the CommSessionMgr Component*
- *Conserving Siebel Application Object Manager Server Resources Through Caching*

- *Improving Performance for Communications Configurations*
- *Configuring Logging for Session Communications*
- *Improving Availability for Session Connections*
- *Improving Screen Pop Performance*
- *Reviewing Performance Impact of Activity Creation*

Tuning the Siebel Application Object Manager Component

This topic is part of *Guidelines for Session Communications Tuning*.

In general, CommSessionMgr and CommConfigMgr components use a small percentage of the resources of the Siebel Server on which they run, while Siebel Application Object Manager performance has the greatest effect on overall system performance, even when the communications components are present. (However, the resources used by CommSessionMgr depends on call center activity and on the third-party communications driver in use.)

Siebel Application Object Manager memory requirements for agent sessions depend on many factors. Siebel Application Object Manager memory usage for an agent using session communications is greater than for other users (those who are not defined as agents in a communications configuration).

Siebel Application Object Manager tuning also depends on your communications configuration caching methods. See also *Conserving Siebel Application Object Manager Server Resources Through Caching*. For more information about Siebel Application Object Manager tuning, see *Tuning the Siebel Application Object Manager for Performance*.

Tuning the CommSessionMgr Component

This topic is part of *Guidelines for Session Communications Tuning*.

For the CommSessionMgr component, the `MaxTasks` parameter determines the maximum number of communications events that can be processed at one time.

Generally, the default values are appropriate for the `MaxTasks`, `MinMTServers`, and `MaxMTServers` parameters, particularly if CommSessionMgr runs on each Siebel Application Object Manager computer.

If you use a dedicated Siebel Server computer to run the CommSessionMgr component, then it might be appropriate to set these parameters to higher values to optimize usage of server resources such as CPU and memory. See also *Topology Considerations for Session Communications*.

Conserving Siebel Application Object Manager Server Resources Through Caching

This topic is part of *Guidelines for Session Communications Tuning*.

You can use two caching mechanisms to make communications configuration data load faster for each agent session and to reduce demand on server resources on the Siebel Application Object Manager. These caching mechanisms can be used together or separately. For more information, see *Siebel CTI Administration Guide* .

- **CommConfigCache parameter (Siebel Application Object Manager).** Setting the `CommConfigCache` parameter on the Siebel Application Object Manager to `TRUE` caches communications configuration data when the first

agent logs in. Configuration data is cached until the Siebel Application Object Manager is restarted. For agents associated with the same communications configuration, each agent session uses the same cached data. See also *Tuning the Siebel Application Object Manager Component*.

Performance is improved for subsequent agent logins, because the configuration data is loaded from the cache rather than from the database.

Siebel Application Object Manager scalability is also improved because configuration data is shared in Siebel Application Object Manager memory across agent sessions, therefore conserving server resources even as the number of agent sessions increases.

- **CommConfigMgr server component and CommConfigManager parameter (Siebel Application Object Manager).** The CommConfigMgr server component caches communications configuration data when the first agent logs in. Setting the `CommConfigManager` parameter on the Siebel Application Object Manager to `TRUE` enables this server component.

  Performance is improved for subsequent agent logins, because the configuration data is loaded from the cache rather than from the database.

  Using the CommConfigMgr component to cache data speeds up the login process and reduces memory usage per agent session because the component uses configuration data that was already cached on the Siebel Application Object Manager component.

  Although it is not required to use the CommConfigMgr component in conjunction with the `CommConfigCache` parameter for Siebel Application Object Manager, if you use them together, then communications configuration data gets cached at the enterprise level rather than only for the Siebel Application Object Manager. Overall performance can be enhanced compared to using each of these mechanisms separately.

Improving Performance for Communications Configurations

This topic is part of *Guidelines for Session Communications Tuning*.

When you deploy session communications, you create communications configurations, define employees as agents, and associate each agent with one or more configurations. How you do these things affects performance and scalability.

In a deployment supporting a large number of agents across multiple physical sites, you must determine criteria for grouping your agents within configurations.

For example, some dialing filters that you define, using the parameter `DialingFilter.RuleN` , might be appropriate for agents at a specific place, such as within the same country or area code. Other dialing filters might be suitable for a different set of agents.

In addition, some switch, teleset, or CTI middleware settings are reflected in your communications configuration, and can differ between physical locations.

It might be helpful to define a communications configuration to apply to users at a single location only. In addition to simplifying the process of defining communications configurations, telesets, or other elements, this approach can help you reduce demand on server resources such as Siebel Application Object Manager memory or CPU.

If call transfers or similar functions are to be supported between contact centers, then additional configuration issues apply.

For more information about defining communications configurations and agents, see *Siebel CTI Administration Guide* .

Configuring Logging for Session Communications

This topic is part of *Guidelines for Session Communications Tuning*.

Logging data can be analyzed as part of performance monitoring or tuning, as described in *Monitoring Siebel Application Performance with Siebel ARM*.

Higher levels of logging provide more data to help you resolve system errors or performance issues; this is appropriate for system testing. For production systems, however, logging levels must be reduced to improve performance.

Log-related parameters applicable to session communications are summarized in this topic. The Siebel Application Object Manager component logs activity related to the user's client session, including usage of the communications toolbar, screen pops, and so on. The CommSessionMgr logs activity related to this component, such as commands and events for the communications driver.

The logging for Siebel Application Object Manager and CommSessionMgr are written to separate files for each user. Typically (though not necessarily), these logging mechanisms both write into the same set of files. This makes it easier to monitor or troubleshoot issues related to session communications for a particular user session. For more information about these logging parameters, see *Siebel CTI Administration Guide* .

Siebel Application Object Manager Logging Parameters

Siebel Application Object Manager parameters that log session communications activity include:

- **CommLogFile.** Specifies the name of the log file (default value is SComm.log). A separate log file is created for each agent session, in the form SComm_username.log.
- **CommLogDebug.** Specifies whether log files must contain extra detail. Setting this parameter to `FALSE` provides better performance.
- **CommMaxLogKB.** Specifies the maximum size of log files.
- **CommReleaseLogHandle.** Specifies that the log file handle must be released periodically. The default setting of `TRUE` provides better performance.

CommSessionMgr Logging Parameters

CommSessionMgr parameters that log session communications activity include:

- **LogFile.** Specifies the name of the log file (default value is SComm.log). A separate log file is created for each agent session, in the form SComm_username.log.
- **LogDebug.** Specifies whether log files must contain extra detail. Setting this parameter to `FALSE` provides better performance.
- **MaxLogKB.** Specifies the maximum size of log files.
- **ReleaseLogHandle.** Specifies that the log file handle must be released periodically. The default setting of `TRUE` provides better performance.

Improving Availability for Session Connections

This topic is part of *Guidelines for Session Communications Tuning*.

When agents log in to the Siebel application after experiencing browser failure or a dropped connection, session communications might sometimes remain unavailable.

Session communications availability can be considered a performance issue. In addition to affecting agent productivity, loss of availability of session communications wastes server resources that could support other functions.

You can improve session communications availability by using the following mechanisms:

- **Push Keep Alive driver.** Using the Push Keep Alive communications driver pushes empty messages (heartbeat messages) to agents at regular intervals. In this manner, it helps keep the communications push channel alive. This feature can help in environments where enforced timeouts sometimes cause communications session connections to be dropped.

  For example, many customers deploy some kind of network appliance to load-balance Web servers. By default, such network appliances can time out connections to browsers, causing communication interruptions for agents. The Push Keep Alive driver generates periodic traffic so connections do not time out due to inactivity.

  To use the Push Keep Alive driver, you create a driver profile, and specify a heartbeat interval (such as 180 seconds) using the `PushKeepAliveTimer` driver parameter. Then you add this profile to your communications configurations.
- **ChannelCleanupTimer parameter (communications configuration).** The `ChannelCleanupTimer` parameter for communications configurations reduces reconnection delays related to session timeouts. This parameter allows the system to identify when a connection is no longer functioning; for example, due to dropped connections or browser failure. If you are using the Push Keep Alive driver, then it is recommended to also use the `ChannelCleanupTimer` parameter.
- **CommMaxMsgQ and CommReqTimeout parameters (Siebel Application Object Manager).** In addition to setting general application timeouts, setting the Siebel Application Object Manager parameters `CommMaxMsgQ` and `CommReqTimeout` can also help you manage agent connections effectively.
- **Backup Communications Session Manager (CommSessionMgr) component.** A backup CommSessionMgr component can be specified using communications configuration parameters. The backup CommSessionMgr component runs on another Siebel Server computer and can be accessed without agent interruption in case the primary CommSessionMgr component fails and does not restart.

For more information about using these features, see *Siebel CTI Administration Guide* .

Improving Screen Pop Performance

This topic is part of *Guidelines for Session Communications Tuning*.

Screen pop response time as experienced by contact center agents is an important indicator of acceptable performance. A screen pop is the display of a view and, optionally, specific records, in response to a communications event. Such events are typically received from CTI middleware; for example, an incoming call is ringing, or the agent has answered the call.

Screen pop behavior is determined by call-handling logic that applies to a particular call based on data attached to the call. Behavior for individual agents is also affected by user settings in the Communications section of the User Preferences screen.

Screen pop performance is affected by the relative complexity of your communications configuration elements, such as event handlers and event responses, and by scripts or business services that might be invoked. Query specifications, database performance, and network capacity and latency also affect screen pop performance. For related information,

see *Improving Performance for Communications Configurations*. For more information about Siebel Web Client response time, see *Tuning Siebel Web Client for Performance*.

Reviewing Performance Impact of Activity Creation

This topic is part of *Guidelines for Session Communications Tuning*.

By default, for each communications work item, an activity record is created in the `S_EVT_ACT` table and related tables.

As you plan your deployment, you must consider how or whether such records are created, review the indexing and layout of applicable database tables, and review the performance impact of generating activity records.

Siebel Email Response Infrastructure

Siebel Email Response uses Communications Server components to enable contact center agents to read and respond to inbound email messages.

It is important to understand the infrastructure that supports Siebel Email Response communications in order to prevent or address performance issues in this area.

Siebel Email Response performance is addressed in this topic and in:

- *Performance Factors for Siebel Email Response*
- *Topology Considerations for Siebel Email Response*
- *Guidelines for Siebel Email Response Tuning*

Key Server Components

Siebel Email Response is supported in the Siebel Server environment primarily by the following components:

- **Communications Inbound Receiver (CommInboundRcvr).** Receives and queues inbound work items, and queues them for processing by Communications Inbound Processor.

  For nonreal-time work items, such as email messages for most deployments of Siebel Email Response, Communications Inbound Receiver queues work items it has received for further processing by Communications Inbound Processor.

  For real-time work items, such as phone calls for Siebel CTI or email messages for some deployments of Siebel Email Response, Communications Inbound Receiver processes work items it has received. Communications Inbound Processor is not used.
- **Communications Inbound Processor (CommInboundProcessor).** Processes inbound work items that were queued by Communications Inbound Receiver.
- **Communications Outbound Manager (CommOutboundMgr).** Sends outbound email or other types of messages.
- **Siebel File System Manager (FSMSrvr).** Writes to and reads from the Siebel File System. This component stores inbound messages prior to processing and stores attachments to inbound and outbound email messages.

Other Siebel Components or Modules

In addition to Siebel Email Response, you might be using Siebel Assignment Manager and Siebel Workflow for routing email messages to agents.

Third-Party Email Server

Siebel Email Response works in conjunction with your third-party email server. Review information presented in documentation for your email server for any requirements that affect your deployment. For information about working with third-party email servers, see *Siebel Email Administration Guide* .

Performance Factors for Siebel Email Response

This topic describes factors that drive or affect performance for Siebel Email Response deployments.

- **Inbound email messages processed per hour.** The number of inbound email messages processed per hour (or some other time period) by your communications infrastructure.
  Requirements for processing outbound messages are relatively minor and are tied to inbound message volume. However, other usage of the CommOutboundMgr component or of the email system must also be considered. For example, the Send Email command can be configured to send email through CommOutboundMgr.
- **Volume of customer data.** The total volume of customer data, including templates or categories, literature items, and so on. Template format (HTML or plain text) is a related factor.

Other factors include the size and complexity of inbound email messages and outbound replies.

Also relevant are user settings in the Outbound Communications section of the User Preferences screen, such as whether a reply contains the original message (Include Original Message in Reply setting), or whether HTML or plain text is an agent's default message format (Default Message Format setting).

> **Note:** Siebel Email Response coverage in this book focuses on inbound and outbound email processing. In a multichannel environment, session communications performance issues also apply.

Topology Considerations for Siebel Email Response

Processing inbound email messages makes more demands on server resources, particularly CPU usage levels, than processing outbound messages.

Processing of inbound messages associated with a single response group must be handled on a single computer.

If inbound message volume warrants it and if multiple server computers are available to run CommInboundRcvr, CommInboundProcessor, and other components, then consider running CommInboundRcvr and CommInboundProcessor on separate computers (or computers) from other Communications Server components.

Topology options for these components are different for real-time and nonreal-time processing. For more information about CommInboundRcvr and CommInboundProcessor, see *Siebel Email Administration Guide* .

Combining processing of messages for multiple email accounts in a single response group can make processing of inbound messages more efficient. However, if message volume is expected to grow, then limiting the number of email accounts processed by each response group will give you more flexibility to distribute processing across multiple servers, and thereby avoid processing bottlenecks.

Guidelines for Siebel Email Response Tuning

Using your hardware resources optimally, and configuring your system appropriately, can help you to achieve your performance goals. Consider your resources and requirements carefully, and test and monitor system performance on a continual basis.

Review information presented in *Siebel Email Administration Guide* , *Siebel CTI Administration Guide* , relevant third-party documentation, and other sources.

Configuring CommInboundRcvr Threads

Each CommInboundRcvr task runs multiple threads to process inbound email. To determine the number of threads, set the parameters **`MinThreads`** and **`MaxThreads`**. If extra CPU capacity exists on a given server computer, you can run more threads for each applicable CommInboundRcvr task.

Managing Email Processing Directories

By default, CommInboundRcvr temporarily writes the content of inbound email messages into subdirectories of the Siebel Server installation directory, until the messages can be processed by the applicable response group and workflow process.

You can use parameters for the Internet SMTP/POP3 Server or Internet SMTP/IMAP Server communications driver to specify alternative directory locations for incoming email, processed email, sent email, and email messages representing certain other processing statuses. You can also set certain driver parameters to specify whether to save or delete processed email messages, for example.

- You must consider the resource requirements for temporary email processing directories when you set up your system.
- Do not delete messages from incoming or queued email directories. Email messages written to processed or sent directories can subsequently be deleted or saved, according to your needs.
- Because of the frequency by which CommInboundRcvr processing writes to temporary email processing directories, the disk must be defragmented regularly.

For more information about email processing directories, see *Siebel Email Administration Guide* .

Reviewing Performance Impact of Activity Creation

For each email work item, an activity record is created in the `S_EVT_ACT` table and related tables.

Attachments to such activity records, for inbound and outbound messages, are stored in the Siebel File System.

As you plan your deployment, you must consider how such records are created and managed, review the indexing and layout of applicable database tables, and review the performance impact of generating activity records.

In addition, you must consider the resource requirements for the Siebel File System for storing activity attachments.

The FSMSrvr server component generally runs on the same Siebel Server computers where you are running CommInboundRcvr and CommOutboundMgr.

> **Note:** Because of the frequency by which Siebel Email Response processing writes to the Siebel File System, the disk must be defragmented regularly.

For more information about activity attachments stored for inbound email, see *Siebel Email Administration Guide* .

Configuring Logging for Siebel Email Response

Logging data can be analyzed as part of performance monitoring or tuning, as described in *Monitoring Siebel Application Performance with Siebel ARM*.

Higher levels of logging provide more data to help you resolve system errors or performance issues; this is appropriate for system testing. For production systems, however, logging levels must be reduced to improve performance.

For the Internet SMTP/POP3 Server or Internet SMTP/IMAP Server communications driver, an applicable parameter is `LogDebug`. For more information, see *Siebel Email Administration Guide* .

Applicable event log levels for Siebel Email Response include those for task execution, workflow step execution, workflow process execution, and workflow performance.

7 Tuning Siebel Workflow for Performance

Tuning Siebel Workflow for Performance

This chapter provides guidelines for tuning workflow processes and policies to achieve and maintain optimal performance and scalability. It contains the following topics:

- *About Siebel Workflow*
- *Monitoring Workflow Policies*
- *Tuning Workflow Policies for Performance*
- *Tuning Workflow Processes*
- *Tuning Workflow Process Manager for Performance*

For more information about Siebel Workflow, see the following documents on the *Siebel Bookshelf* :

- *Siebel Business Process Framework: Workflow Guide*
- *Configuring Siebel Business Applications*
- *Siebel System Administration Guide*

About Siebel Workflow

Siebel Workflow is an interactive software tool that automates business processes.

Workflow processes are designed and administered using the Business Process Designer, a graphical user interface provided through Siebel Tools. Designing, planning, creating, and testing individual workflow processes using the Business Process Designer are described in detail in *Siebel Business Process Framework: Workflow Guide* .

Workflow Processes and Workflow Policies are two components of Siebel Workflow that are designed and created when automating a business process. These components are defined as follows:

- **Workflow Processes.** The representation of a business process. A workflow process comprises one or more steps that indicate when a business process starts and ends and includes information about individual activities within the business process.
- **Workflow Policies.** A systematic expression of a business rule. A workflow policy contains one or more policy conditions and one or more policy actions. If all of the policy conditions for a workflow policy are true, then the policy action occurs when all of the policy conditions are met. A workflow policy is contained by one workflow policy group and is related to one workflow policy object. A workflow policy contains additional properties that govern its behavior.

Monitoring Workflow Policies

You need to monitor Workflow Policies regularly to check that all events are handled correctly and that the Siebel Server uses its resources optimally. Purging your log files periodically prevents them from becoming too large.

Workflow Policies use the `General Events` event for logging. To see informational messages, set the log level to 3. To see debugging information, set the log level to 4.

You can monitor Workflow Policies using the following views, log files, and tables:

- **Policy Frequency Analysis view.** For details, see *Using the Policy Frequency Analysis View*.
- **Workflow Agent trace files.** For details, see *Using Workflow Agent Trace Files*.
- **Workflow Policies tables.** For details, see *Monitoring Workflow Policies Tables*.

Using the Policy Frequency Analysis View

This topic is part of *Monitoring Workflow Policies*.

The Policy Frequency Analysis view provides a list of all executed policies. The Policy Frequency Analysis view is available to analyze how frequently policies are executed over time.

This view displays a log of all of the policies executed, as evidenced by a Workflow Monitor Agent process. The policy maker can monitor Workflow Agent process activity to determine whether the current policies are adequate, new policies need to be created, or policies need to be refined.

The Policy Frequency Analysis view lets you view Policy Log data in a graphical format. The log information is generated by Siebel Server components for Workflow Policies. You access the Policy Frequency Analysis view from the Siebel client by navigating to Policy Frequency Analysis view in the Administration - Business Process screen.

The Policy Frequency Analysis view contains the following fields:

- **Policy.** The name of the policy that was executed.
- **Workflow Object.** The name of the assigned workflow policy object.
- **Object Identifier.** The ID of the workflow policy object for which the policy was executed.
- **Object Values.** Identifying information for the row that executed the policy.
- **Event.** The date and time of the policy execution event.

Using Workflow Agent Trace Files

This topic is part of *Monitoring Workflow Policies*.

Workflow agent trace files include the following:

- **Workflow Monitor Agent trace file.** Workflow Monitor Agent provides detailed information about its processing in its trace file.
- **Workflow Action Agent trace file.** Workflow Action Agent provides detailed information about its processing in its trace file.

  Setting tracing on the Workflow Action Agent task is required only when the parameter `Use Action Agent` for Workflow Monitor Agent is set to `TRUE`. In this case, Workflow Action Agent must be started manually. (It also must be started manually when you use email consolidation.)

  `Use Action Agent` is `FALSE` by default: Workflow Action Agent is started automatically by Workflow Monitor Agent.
- **Email Manager and Page Manager trace files.**

- Run Email Manager and Page Manager components with `Trace Flag` set to 1 for detailed reporting on email activity.
- Query `S_APSRVR_REQ` for status information on email and page requests that were logged by Workflow Action Agent.

Monitoring Workflow Policies Tables

This topic is part of *Monitoring Workflow Policies*.

Workflow Policies use three database tables for processing and tracking requests:

- `S_ESCL_REQ`
- `S_ESCL_STATE`
- `S_ESCL_ACTN_REQ`

Monitor these tables to verify that policies are being processed correctly.

When a trigger fires against a Workflow Policy condition, a record is inserted in the escalation request table, `S_ESCL_REQ`. Records in this table identify rows in the database that could trigger a Workflow Policy to take action. After the Workflow Monitor Agent processes a request, it removes the row from this table.

The `S_ESCL_STATE` time-based table identifies all of the rows that have been executed (all conditions are true) and are waiting for the time duration element to expire.

The `S_ESCL_ACTN_REQ` table identifies all of the rows that are awaiting action execution. These rows have violated the policy; and the time duration element, if any, has expired.

If one of these tables (`S_ESCL_REQ`, `S_ESCL_STATE`, and `S_ESCL_ACTN_REQ`) becomes very large, then this could indicate that the number of policies being monitored is too large, and new Workflow Policies processes need to be created to share the load and improve performance.

If rows are being monitored, but are not being removed from a table after the time interval is met, then this could indicate that a policy was deactivated without removing the database triggers. The triggers are continuing to send data that is not being acted on by a Workflow Policies process. These tables will become very large if you do not restart Generate Triggers.

If you expire or delete any active Workflow Policies, confirm that no outstanding records remain in the `S_ESCL_REQ`, `S_ESCL_STATE`, or `S_ESCL_ACTN_REQ` tables.

Maintain the `S_ESCL_REQ`, `S_ESCL_STATE`, and `S_ESCL_ACTN_REQ` tables by adjusting parameters related to storage, access, and caching. See database documentation for additional information about properly adjusting such parameters. Also, make sure that the database administrator (DBA) is aware of these key tables.

Tuning Workflow Policies for Performance

Workflow Policies can be tuned to optimize your resources while also meeting the policy's timing requirements by grouping similar policies and assigning these policy groups to Siebel Servers that can handle the workload. Performance tuning can be handled in several interrelated ways. The following topics provide more information:

- *Creating Workflow Policy Groups to Manage Siebel Server Load*
- *Multiple Workflow Monitor Agents and Workflow Action Agents*
- *Running Workflow Agents on Multiple Siebel Servers*
- *Setting Optimal Sleep Interval for Workflow Policy Groups*
- *Setting Optimal Action Interval for Workflow Monitor Agent and Workflow Action Agent*

Creating Workflow Policy Groups to Manage Siebel Server Load

This topic is part of *Tuning Workflow Policies for Performance*.

Workflow policy groups allow you to group policies with similar polling intervals. This distributes the load to allow efficient processing. For example, if you have very critical policies that must be responded to within minutes of the policy trigger event and you have other policies that need a response within a day, then you can assign them to different workflow policy groups.

The advantage of selective grouping is that a Workflow Agent's polling resources are focused on a smaller number of policies, which helps make monitoring and action execution more effective.

Multiple Workflow Monitor Agents and Workflow Action Agents

This topic is part of *Tuning Workflow Policies for Performance*.

Each Workflow Agent combination monitors the policies within its assigned workflow policy group. If you are a high-volume call center or you have a large number of policies that need very short polling intervals, then you might want to create multiple groups with Workflow Agent processes to run in parallel. A single Workflow Agent process that is monitoring and handling a large number of events can become slow to respond and not meet the time interval commitments set by the policy.

Running multiple Workflow Monitor Agent and Workflow Action Agents in parallel:

- Focuses a component's polling resources on a smaller number of workflow policies.
- Allows faster throughput by shortening the time between when the workflow policy event is triggered and when the component notices the event.

Running Workflow Agents on Multiple Siebel Servers

This topic is part of *Tuning Workflow Policies for Performance*.

You can run Workflow Agent processes on different Siebel Servers to ease the workload on each Siebel Server. You can then adjust the polling interval for each group so that polling for noncritical policies does not prevent efficient processing of critical policies.

By distributing workflow policy processes across Siebel Servers:

- High-maintenance policies can be grouped on a Siebel Server with sufficient resources to handle the workflow CPU requirements.
- Low-maintenance policies can be run on a Siebel Server that shares resources with other Siebel processes.

Setting Optimal Sleep Interval for Workflow Policy Groups

This topic is part of *Tuning Workflow Policies for Performance*.

By creating groups with similar polling intervals, you can assign the workflow policy group to a Workflow Agent process with a polling rate that matches the workflow policy group. Different polling intervals can be assigned to each workflow policy group using the `Sleep Time` parameter. For more information about Workflow Policies server administration, see *Siebel Business Process Framework: Workflow Guide* .

After Workflow Agents process all requests, the agent processes sleep for the interval specified by this argument before processing begins again. Set the sleep intervals as large as is possible, but at an interval that still meets your business requirements.

> **Note:** Setting sleep intervals at values that are too small can put undue stress on the entire infrastructure. Make sure that the sleep interval is as large as possible within the context of the business process.

Adjust the sleep interval for each Workflow Agent process to meet the requirements of each workflow policy group.

For example, workflow policy group A contains accounts that require a response to a Severity 1 service request within 10 minutes. Workflow policy group B contains policies that require a customer follow-up call within 14 days.

Workflow policy group A is very time-critical, so you could set the sleep interval to 60 seconds so that the assigned Workflow Policies instance polls frequently. Workflow policy group B is not as time-critical, so you could set the sleep interval to 48 hours and the Workflow Policy instance can still meet its commitments.

Another example where optimal configuration of the `Sleep Time` parameter might be required is in the case of multiple users who might need to update the same record. If you have, for example, a workflow policy that monitors service requests and you have multiple users that retrieve and modify open service request records, then you need to set the `Sleep Time` parameter so that users will have enough time to update the text fields.

If the sleep interval is not set high enough, then you might encounter an error message stating: `The selected record has been modified by another user since it was retrieved. Please continue.` In this case, you will lose your changes as the new field values for this record are displayed.

> **Note:** If you find that Workflow Policies runs significantly slower during a certain time period, then investigate what other processes might be contending for CPU resources on the Siebel Server. You might discover that the Siebel Server has certain time periods with high activity that interfere with the ability of the Workflow Policies process to monitor or act. Arrange the Workflow Policies processes on the Siebel Servers so that the polling periods are compatible with the resources available.

Setting Optimal Action Interval for Workflow Monitor Agent and Workflow Action Agent

This topic is part of *Tuning Workflow Policies for Performance*.

For each Workflow Monitor Agent or Workflow Action Agent component, you can set the `Action Interval` parameter, which determines when actions for a given policy are re-executed on a given base table row. This setting limits the number of times actions are executed if a row keeps going in and out of a matching condition.

You set the `Action Interval` parameter for Workflow Monitor Agent (rather than Workflow Action Agent) if you have set the parameter `Use Action Agent` to `TRUE` for Workflow Monitor Agent. `Use Action Agent` is `FALSE` by default.

For example, if a service request severity is set to critical and triggers a policy, then you do not want to re-execute the policy action if it is changed and has been reset to critical during this interval.

Tuning Workflow Processes

In order to improve performance when running workflow processes, you can follow the guidelines explained in the following topics:

- *Minimizing Usage of Parameter Search Specification*
- *Monitoring Conditions Based on Parent and Child Business Components*
- *Configuring Siebel CRM for Workflow Performance*
- *Monitoring Memory Overhead for Workflow Processes*

> **Note:** This performance tuning information is provided as general guidelines for tuning and optimizing performance of workflow processes. Every implementation of Siebel CRM is unique, so every use of workflow processes is also unique.

Minimizing Usage of Parameter Search Specification

This topic is part of *Tuning Workflow Processes*.

Although the server component parameter `Search Specification` (alias `SearchSpec`) is a feature of Siebel Workflow, it is recommended that you minimize your use of this parameter with workflow processes that are frequently invoked.

Minimizing `SearchSpec` use, especially for frequently invoked processes, improves Workflow engine performance during runtime because the engine does not have to construct the `SearchSpec` string.

It is important, however, that you do not completely avoid using `SearchSpec`. Not using this parameter can indicate actions taking place on the current row in some cases, and on all rows in other cases. For specific guidelines, note the following:

- For Siebel operations, minimize usage of `SearchSpec`.
- For batch process requests, use `SearchSpec` on the business object to limit the number of rows processed.

Indexing Fields in SearchSpec

If you determine that `SearchSpec` does need to be used, then make sure that all of the fields being used are properly indexed. Proper indexing of the fields helps Siebel Workflow and the underlying database to efficiently build queries.

Monitoring Conditions Based on Parent and Child Business Components

This topic is part of *Tuning Workflow Processes*.

When a condition is being evaluated at a decision step or any other step using a combination of parent and child business components, it is recommended that you closely benchmark the expression or the condition. In some cases, this will require spooling the SQL. For more information, see *Analyzing Generated SQL for Performance Issues*.

> **Note:** The query plan of the SQL might show an extended and poorly performing query. In such cases, it is better to break the conditions up into multiple decision steps and evaluate the conditions separately.

Configuring Siebel CRM for Workflow Performance

This topic is part of *Tuning Workflow Processes*.

In some cases, you might need to perform a comparison between different objects.

Assume, for example, that a service request is assigned to a candidate depending on the industry of the account associated with it. In this case, it is necessary to perform a query against Account to fetch the appropriate industries, or to check an industry against all of the industries with which the account is associated.

If the workflow process in this example is going to be evaluated frequently, consider exposing Account Industry on Service Request by the appropriate configuration in order to enhance workflow performance.

Monitoring Memory Overhead for Workflow Processes

This topic is part of *Tuning Workflow Processes*.

Overhead and performance and scalability characteristics vary depending on whether you are running workflows locally in the Siebel Application Object Manager or in Workflow Process Manager (WfProcMgr), and also on where you run WfProcMgr. The performance and scalability characteristics also depend on whether you are using asynchronous mode for workflow process requests. For more information, see *Siebel Deployment Planning Guide* and *Siebel Business Process Framework: Workflow Guide* .

Running Workflows Locally in Siebel Application Object Manager

A workflow instance (that is, one run of a workflow definition) can run within a Siebel Application Object Manager. In this case, the workflow runs locally, within the current thread that the logged-in user is using. This means that if N users are connected and they all need to run a workflow definition, then the definition would run in that user thread.

In this mode, Workflow adds a fixed overhead (100 to 200 KB) to the user session memory (sometimes referred to as the model) plus memory taken up by other objects (such as business components) contained in the tasks within that workflow.

In general, this option provides the best performance, but is suitable only where scalability is not an important factor.

Running Workflows in Workflow Process Manager

The workflow itself runs within a separate component, which uses a fixed set of resources (parameters `MaxMTServers`, `MaxTasks`) to schedule the workflow. The Workflow Process Manager component (alias WfProcMgr) is a multithreaded process that runs multiple workflows and is more scalable because it uses a pool of threads and models.

Generally, the mode of the workflow used depends on what the application is trying to achieve. It is generally recommended that you try to schedule a workflow task in WfProcMgr, especially if the results of a run are not immediately needed.

You can optionally run WfProcMgr on the same Siebel Server (colocating) as the Siebel Application Object Manager where the workflow is invoked, or run it on dedicated Siebel Server computers. Compared to running workflows locally, running workflows in WfProcMgr might reduce performance, but improve scalability. Running WfProcMgr on dedicated Siebel Servers typically provides the best scalability, while colocating WfProcMgr and Siebel Application Object Manager might provide better performance.

About Asynchronous Mode for Workflow Process Requests

For all Workflow Processes deployment options described previously, workflow process requests can be handled synchronously or using asynchronous mode. Using asynchronous mode comes with the advantages and disadvantages listed in the following table.

Advantages	Disadvantages
All user threads are not loaded. More scalable as long as: - There are maximum N simultaneously connected users. - There are maximum X simultaneous running workflows. - If X is smaller than N, then a WfProcMgr with X tasks can handle a much larger pool (N) of users.	On error, you must look at the log files, because there is no automatic notification. The SRBroker component could have a timeout or retry feature. Slightly more latency. Additional cost (minimal) of one request per response.

Tuning Workflow Process Manager for Performance

This topic provides general approaches to tune and optimize performance of Workflow Process Manager.

> **Note:** Every implementation of Siebel CRM is unique, and so every use of workflow processes is also unique. It is in your best interest to test, continually monitor, and tune your workflow processes to achieve optimal throughput.

You can follow the guidelines explained in the following information:

- *Caching Business Services*
- *Caching Sessions*

Note: Consider the information provided in this topic as general background information. No attempt is made to detail the many variables that affect tuning at specific sites. This content is not a substitute for specific tuning recommendations made by Global Customer Support or other Oracle service organizations.

Caching Business Services

This topic is part of *Tuning Workflow Process Manager for Performance*.

Business services invoked through Workflow Process Manager must have the Cache property set to `TRUE`. This feature makes it possible for the Workflow engine to not reload and reparse the business service, and therefore enhances the performance of workflows that invoke business services.

Note: Predefined Siebel business services that have the Cache property set to `FALSE` must *not* be reset to `TRUE`.

Caching Sessions

This topic is part of *Tuning Workflow Process Manager for Performance*.

The parameter `OM - Model Cache Maximum` (alias `ModelCacheMax`) for Workflow Process Manager determines the size of the cache for model objects (also known as cached sessions). Cached sessions maintain database connections and session data for locale, user preferences, and access control.

Note: Session caching applies only to noninteractive Object Manager-based server components like Workflow Process Manager. It does not apply to Siebel Application Object Manager or EAI Object Manager components.

This feature maintains and reuses existing sessions rather than creating a new session each time one is requested. Using this feature can improve login performance for Workflow Process Manager.

Each model in the cache creates two database connections for the life of the model: one connection for insert, update, and delete operations; another connection for read-only operations.

The default value of `ModelCacheMax` is 10. A value of 0 disables this parameter. The maximum value is 100. In general, you set `ModelCacheMax` to a value approximately equal to the number of concurrent sessions the Workflow Process Manager component is expected to support.

Note: When component sessions use multiple user IDs, session caching provides less benefit relative to its cost. The benefit is greatest for component sessions using the same user ID.

See also *Siebel System Administration Guide* .

8 Tuning Siebel Product Configurator for Performance

Tuning Siebel Product Configurator for Performance

This chapter describes some issues that affect the performance and throughput of server-based deployments of Siebel Product Configurator, and provides guidelines for tuning this module to achieve and maintain optimal performance and scalability. It contains the following topics:

- *Siebel Product Configurator Infrastructure*
- *Performance Factors for Siebel Product Configurator*
- *Topology Considerations for Siebel Product Configurator*
- *Guidelines for Siebel Product Configurator Tuning*
- *About Siebel Product Configurator Caching*
- *Administering the Siebel Product Configurator Cache*

For more information about Siebel Product Configurator, see the following documents on the *Siebel Bookshelf* :

- *Siebel Product Administration Guide*
- *Siebel System Administration Guide*
- *Siebel Deployment Planning Guide*
- *Siebel Installation Guide*

Also see documents for related Siebel Order Management modules:

- *Siebel Pricing Administration Guide*
- *Siebel Order Management Guide*
- *Siebel eSales Administration Guide*

Siebel Product Configurator Infrastructure

Siebel Product Configurator provides product configuration and solution-computing capabilities, and can be deployed as a server-based module. Siebel Product Configurator is one of the Siebel Order Management modules. These modules work together to support various phases in conducting commerce, including online selling.

Siebel Product Configurator uses several infrastructure elements to manage configuration sessions. Siebel Product Configurator is supported in the Siebel Server environment by the following components:

- **Siebel Application Object Manager.** Siebel Product Configurator functions can be performed within the Siebel Application Object Manager, such as Call Center Object Manager (alias SCCObjMgr_enu in a U.S. English environment) for Siebel Call Center.
- **Siebel Product Configuration Object Manager.** An optional component, suitable for some Siebel Product Configurator deployments, that processes configuration requests for user sessions submitted from a Siebel Application Object Manager component.

- This component has the alias eProdCfgObjMgr_locale, such as eProdCfgObjMgr_jpn in a Japanese locale. Typically, this component is run on a separate Siebel Server computer than the one running the Siebel Application Object Manager. Multiple instances of this component can be run on a Siebel Server, where it is possible to distribute requests across the various instances. For more information, see *Topology Considerations for Siebel Product Configurator*.

Note: The three-letter extension to the alias of the Siebel Product Configuration Object Manager component (such as jpn in the example eProdCfgObjMgr_jpn) must correspond to the value for the `Locale Code` parameter (alias `LocaleCode`) associated with the invoking Application Object Manager. For more information about this requirement, see *Siebel Deployment Planning Guide* . For more information about the `Locale Code` parameter, see *Siebel Global Deployment Guide* .

For more information about the elements of the internal architecture of Siebel Product Configurator, including Instance Broker (Complex Object Instance Service business service) and Configurator Object Broker (Cfg Object Broker business service), see *Siebel Product Administration Guide* .

Performance Factors for Siebel Product Configurator

In planning Siebel Product Configurator server-based deployments, or in troubleshooting performance for existing deployments, you must consider several key factors that determine or influence performance. Subsequent topics provide information and guidelines to help you achieve and maintain optimal performance and scalability.

Performance contexts to consider include response times for:

- **Loading customizable products.** This is the time elapsed from the moment a user clicks Customize in a quote or order until the user interface for the customizable product has been loaded and displayed to the user.
- **Responding to user selections.** This is the time elapsed from the moment a selection is made by the user until Siebel Product Configurator returns a response such as an update to the customizable product or a conflict message.

The following factors, particularly customizable product size and complexity, are relevant in both of these contexts.

Some of the key performance factors for server-based deployments of Siebel Product Configurator include:

- **Number of concurrent configuration users.** The number of concurrent users who access customizable product models. This figure will be some percentage of the total number of concurrent users on the Siebel Application Object Manager.

  More specifically, you would be concerned with the total number of configuration sessions per hour, and the average length of those sessions.
- **Size and complexity of product models.** The total size and complexity of each customizable product model, particularly where multiple hierarchical levels, many constraints, and a complex user interface are defined.

  A major potential performance factor is custom scripting attached to update events on applicable business components, such as Quote, Quote Item, Quote Item Attribute, Order, Order Item, and Order Item Attribute.
- **Number of product models.** The number of customizable product models accessed by users. It is assumed that each user accesses no more than one customizable product model at one time. A given group of concurrent users can access multiple models, however, each of which must be separately cached.
- **Type of Siebel Product Configurator deployment.** Performance of a customizable product is the same irrespective of whether the deployment is running Siebel Product Configurator in the Siebel Application Object

Manager component or running the Siebel Product Configuration Object Manager component on dedicated servers.

Topology Considerations for Siebel Product Configurator

This topic describes considerations for defining the topology for Siebel Product Configurator server-based deployments. There are two major topology approaches to deploying Siebel Product Configurator:

- Running Siebel Product Configurator in the Siebel Application Object Manager component. For more information, see *Running Siebel Product Configurator in the Siebel Application Object Manager Component*.
- Running the Siebel Product Configuration Object Manager component on one or more dedicated Siebel Servers intended to run only Siebel Product Configurator. (Dedicated servers are sometimes referred to as remote servers, because they are remote to the computer on which Siebel Application Object Manager is running. In general, this chapter uses the term dedicated servers.) For more information, see *Running Siebel Product Configurator on Dedicated Servers*.

  For configuration information for this option, see *Configuring Siebel Application Object Manager for Dedicated Siebel Product Configurator Deployments*.

The optimal deployment approach for Siebel Product Configurator, and the optimal number of server computers that you require for this module, depends on factors such as those described in *Performance Factors for Siebel Product Configurator*. See also *Siebel Product Configurator Infrastructure*.

Running Siebel Product Configurator in the Siebel Application Object Manager Component

This topic is part of *Topology Considerations for Siebel Product Configurator*.

You can run Siebel Product Configurator in the Siebel Application Object Manager component, such as for Siebel Call Center.

If a small number of concurrent users require configuration sessions, or there are a small number of customizable product models, then this deployment option can yield reasonable performance and make the most effective use of your hardware resources.

With this option, you set all parameters for managing Siebel Product Configurator caching on each applicable Siebel Application Object Manager. For more information, see *About Siebel Product Configurator Caching*.

Running Siebel Product Configurator on Dedicated Servers

This topic is part of *Topology Considerations for Siebel Product Configurator*.

You can run Siebel Product Configurator on one or more dedicated Siebel Server computers using a server component other than the Siebel Application Object Manager. This component is Siebel Product Configuration Object Manager (alias eProdCfgObjMgr_jpn in a Japanese locale, for example).

Possible variations on this general topology option include:

- Running one eProdCfgObjMgr component with one Siebel Application Object Manager component
- Running multiple eProdCfgObjMgr components with one Siebel Application Object Manager component
- Running one eProdCfgObjMgr component with multiple Siebel Application Object Manager components
- Running multiple eProdCfgObjMgr components with multiple Siebel Application Object Manager components

If a large number of concurrent users require configuration sessions, or there are a large number of customizable product models, then this deployment option (using one or more dedicated servers) can yield the best performance and make the most effective use of your hardware resources.

With this option, you set some parameters for managing Siebel Product Configurator caching on each applicable Siebel Application Object Manager, and some on each applicable dedicated Siebel Product Configurator server. For more information, see *Configuring Siebel Application Object Manager for Dedicated Siebel Product Configurator Deployments* and *About Siebel Product Configurator Caching*.

As of Siebel CRM 21.8 Update, administrators can configure session pooling for Siebel Product Configurator. This feature is intended to provide better performance when users or agents work with customizable products that are routed to remote (dedicated) instances of the Siebel Product Configurator Object Manager. For more information, see *Configuring Session Pooling for Siebel Product Configurator*.

Configuring Siebel Application Object Manager for Dedicated Siebel Product Configurator Deployments

This topic is part of *Topology Considerations for Siebel Product Configurator*.

When you designate one or more dedicated server computers to run the Siebel Product Configuration Object Manager (alias eProdCfgObjMgr_jpn in a Japanese locale, for example) component, then you must configure any Siebel Application Object Manager components from which users will initiate configuration sessions to route configuration requests to these computers.

The Siebel Application Object Manager forwards each configuration session request to the dedicated Siebel Product Configurator server with the fewest concurrent users.

The following information lists server parameters for managing dedicated Siebel Product Configurator deployments. Using Server Manager, set these parameters on each Siebel Application Object Manager. *Do not set these parameters on the dedicated Siebel Product Configurator server computer.*

Parameter Name	Display Name	Data Type	Default Value	Description
eProdCfgRemote	Product Configurator-Use remote service	Boolean	`FALSE`	Set this parameter to `TRUE` if you are running the eProdCfgObjMgr component on one or more dedicated servers. Set this parameter to `FALSE` for Siebel Product Configurator deployments using Siebel Application Object Manager only.

Parameter Name	Display Name	Data Type	Default Value	Description
eProdCfgServer	Product Configurator- Remote Server Name	Text	Not applicable	**Note:** Instead of using this parameter, customers are now advised to configure dedicated Siebel Product Configurator servers by using the Administration - Product screen, Cache Administration view. When you have not enabled explicit product mapping for products to a Siebel Product Configurator server, set this parameter to the names of the dedicated computers on which you are running eProdCfgObjMgr. Otherwise, set the value of this parameter to NULL.
eProdCfgTimeOut	Product Configurator- Time out of connection	Integer	20	Sets the length of time, in seconds, that the Siebel Application Object Manager tries to connect to a dedicated Siebel Server running eProdCfgObjMgr. After the timeout has been reached, an error is returned to the user.
eProdCfgKeepAliveTime	Product Configurator - Keep Alive Time of Idle Session	Integer	900	Setting in seconds to determine the maximum interval of inactivity during a configuration session. If the interval of inactivity reaches this value, then the user session is stopped and the worker returns to the pool. If this parameter is not set, then an infinite interval is assumed. Set this parameter on the Siebel Application Object Manager only. It does not apply on the dedicated Siebel Product Configurator server. **Note:** On the dedicated Siebel Product Configurator server (eProdCfgObjMgr component), set the parameter **`ConnIdleTime`** to a value like that of **`eProdCfgKeepAliveTime`**, plus 1 second.

Guidelines for Siebel Product Configurator Tuning

Using your hardware resources optimally, and configuring your system appropriately, can help you to achieve your performance goals. Consider your resources and requirements carefully, and test and monitor system performance on a continual basis.

Review information presented in *Siebel Product Administration Guide* , *Siebel System Administration Guide* , and other sources.

Activities that you can perform to achieve performance and scalability goals include:

- Adjusting your system topology. For more information, see *Topology Considerations for Siebel Product Configurator*.
- Configuring Siebel Server server components for Siebel Product Configurator.
- Designing and deploying your customizable product models. For more information, see *Defining Customizable Product Models and Classes*.

This topic applies to deployments using Siebel Web Client. It contains the following information:

- *Tuning Siebel Product Configurator*
- *Specifying the Siebel Product Configurator File System Location*
- *Defining Customizable Product Models and Classes*

Tuning Siebel Product Configurator

This topic is part of *Guidelines for Siebel Product Configurator Tuning*.

How you configure your Siebel Server components for Siebel Product Configurator server deployments, for appropriate tuning, depends in part upon which deployment method you use, as described in *Topology Considerations for Siebel Product Configurator*.

- If you deploy Siebel Product Configurator on the Siebel Application Object Manager, then your Siebel Product Configurator tuning calculations must be made in combination with your Siebel Application Object Manager tuning calculations.
- If you deploy Siebel Product Configurator using the Siebel Product Configuration Object Manager (eProdCfgObjMgr) server component on a dedicated Siebel Server computer, then your Siebel Product Configurator tuning calculations will be only indirectly related to your Siebel Application Object Manager tuning calculations and will be determined primarily by configuration-related concurrent users and request loads.

In particular, note that, for a dedicated Siebel Product Configurator server, the **MaxTasks** parameter is generally set much lower than it is for a Siebel Application Object Manager. By default, the ratio of **MaxTasks** to **MaxMTServers** is 20:1 for eProdCfgObjMgr. In addition, depending on the request load, **MaxTasks** is generally set lower for a Siebel Application Object Manager running Siebel Product Configurator than it is for a Siebel Application Object Manager that is *not* running Siebel Product Configurator.

You can follow this general procedure to determine how to set these parameters:

- Determine what percentage of users for your Siebel application are also users of Siebel Product Configurator. For example, for every 100 users, 60 work with Quotes.
- Calculate what percentage of time these users spend using Siebel Product Configurator. For example, out of the 60 users mentioned previously, only 30 are concurrently using Siebel Product Configurator.
- Maintain the default ratio of 20:1 for **MaxTasks** divided by **MaxMTServers**.

If you deploy Siebel Product Configurator using eProdCfgObjMgr on a dedicated Siebel Server computer and the database connection (login and logout) is slow, then it is recommended that you do the following:

- Enable database connection pooling

  To enable connection pooling, set the parameters **MaxSharedDbConns** and **MinSharedDbConns** to positive integer values (at least 1) that are no higher than **MaxTasks** minus 1.

This setting pools all user connections without sharing and avoids the creation and deletion of a new database connection for each eProdCfgObjMgr session. For more information about database connection pooling, see *Configuring Database Connection Pooling for Siebel Application Object Managers*.

- Use third-party user authentication
  Using third-party user authentication, such as LDAP, rather than database authentication avoids creating an additional database connection for authentication. For more information about authentication options, see *Siebel Security Guide* .

Specifying the Siebel Product Configurator File System Location

This topic is part of *Guidelines for Siebel Product Configurator Tuning*.

The Siebel Product Configuration Object Manager, running on a dedicated Siebel Server computer, can use a Siebel File System directory to cache all configuration-related object definitions. The server parameter, **`Product Configurator - FS location`** (alias **`eProdCfgCacheFS`**), specifies the location of this directory. Specify a value for this parameter to reference a directory path that has write permission. For example, you might specify **`\\MyServer\SiebFS\SiebConfig`**. Note the following considerations:

- Although you can specify any network-accessible directory for caching object definitions, it is strongly recommended that you specify the full path to a directory that is local to the dedicated Siebel Product Configurator server, such as **`c:\siebel\SiebConfig`** on Microsoft Windows. Do not specify a top-level directory of a network directory. For example, if **`SiebFS`** is a top-level directory (at a location like **`\\MyServer\SiebFS`**), then specify a subdirectory, such as **`\\MyServer\SiebFS\SiebConfig`**, as the value for the **`eProdCfgCacheFS`** parameter.
- If you do not specify a value for **`eProdCfgCacheFS`**, then Siebel Product Configurator attempts to use the location of the Siebel File System directories, as configured for the Siebel Server, for caching object definitions.

Note: If your deployment uses the File System Manager (alias FSMSrvr) component, however, then Siebel Product Configurator cannot cache object definitions in the Siebel File System directories, and you must specify a location using the **`eProdCfgCacheFS`** parameter. For more information about the Siebel File System, see *Siebel Installation Guide* and *Siebel System Administration Guide* .

For more information about caching in the Siebel Product Configurator File System, see *About Siebel Product Configurator Caching*.

Defining Customizable Product Models and Classes

This topic is part of *Guidelines for Siebel Product Configurator Tuning*. It describes some guidelines about creating customizable products and classes in a manner that will optimize performance:

- To maintain good performance, do not make your customizable products or classes any larger or more complex than absolutely necessary.
- Complexity is a function of the number of hierarchical levels and constraints built into the customizable product models and of the structure of the class.
- For defining class relationships, use specific classes as much as possible. For example, avoid defining class relationships without specifying classes, or use a subclass rather than a parent class if it is so defined.
- Minimize the complexity of user interface elements that you associate with your customizable product models.

- Generally, using interactive or automatic pricing updates for customizable products is recommended. If performance is adversely affected, consider switching to manual pricing updates.
- When creating rules, using the Set Preference template allows you to create soft constraints that guide the Siebel Product Configurator engine in producing solutions, but which the engine can ignore if needed to avoid conflicts or performance problems.
- By default, when you add a customizable product to a quote, for example, default products and selections will be included, and Siebel Product Configurator can be invoked to create this default instance. If the customizable product default selections are large and complex, and if users are required to immediately customize the product, then turning off the Default Instance Creation feature will enhance performance with no loss of functionality.

For more information about these issues, see *Siebel Product Administration Guide* .

About Siebel Product Configurator Caching

Siebel Product Configurator supports several types of caching of customizable product information, to optimize response time for configuration-session users. Caching options include:

- Caching in memory
- Siebel Product Configurator caches versions of customizable products, product classes, and attribute definition objects in memory. When the size limit for this cache is reached, the versions of the objects that were least recently used are discarded. For more information, see *Default Caching Behavior for Siebel Product Configurator*.

  > **Note:** The memory resources for your Siebel Product Configurator server computer must be sufficient to support your caching requirements.

- Caching in the Siebel Product Configurator File System
- This Siebel Product Configurator File System directory caches versions of the customizable products, product classes, and attribute definition objects that were loaded into memory. This is default behavior. For more information, see *Specifying the Siebel Product Configurator File System Location* and *Default Caching Behavior for Siebel Product Configurator*.
- In addition to these caching options, you can also specify which server or component caches versions of customizable products, product classes, and attribute definition objects.
- The specified cache can be updated at regular intervals. Using these options can improve response times to requests for a specific customizable product. For more information, see *Cache Management for Siebel Product Configurator*.

This topic contains the following information:

- *Default Caching Behavior for Siebel Product Configurator*
- *Cache Management for Siebel Product Configurator*
- *Parameters for Configuring Siebel Product Configurator Caching*
- *Determining Rough Sizing for Caching Parameters*
- *Configuring Session Pooling for Siebel Product Configurator*

Default Caching Behavior for Siebel Product Configurator

This topic is part of *About Siebel Product Configurator Caching*.

The default caching behavior for Siebel Product Configurator is as follows:

- When a user starts a configuration session, Siebel Product Configurator looks for new cache update requests that affect objects in the cache. If there are new cache update requests that affect objects currently in the cache, then Siebel Product Configurator updates or removes these objects.
- Siebel Product Configurator checks whether the requested customizable product is cached in memory.
- If the customizable product is not already cached in memory, then Siebel Product Configurator looks in the Siebel Product Configurator File System.

  > **Note:** The location of the Siebel Product Configurator File System is specified by the value of the `Product Configurator - FS location` parameter (alias `eProdCfgCacheFS`). For more information about the Siebel Product Configurator File System, see *Specifying the Siebel Product Configurator File System Location* .

- If the customizable product is not in the Siebel Product Configurator File System, then it is loaded from the Siebel database. The product is added to the memory cache and to the Siebel Product Configurator File System.
- Thereafter, when a configuration session starts, the customizable product is loaded from the memory cache or from the Siebel Product Configurator File System.
- Before loading the customizable product from the Siebel Product Configurator File System, the system checks the Siebel database to make sure each item in the product is the current version.
- If the cached product has changed in the database, then the current version of the item is loaded from the database. This makes sure that the most recent version of a customizable product and its contents are loaded.
- When the product administrator releases a new version of a customizable product, the changes are written to the Siebel database and a cache update request is posted for the modified customizable product. The memory cache and the Siebel Product Configurator File System are not updated with the changes until the next configuration session is requested for the customizable product.

  > **Note:** It is recommended that you avoid the use of start or end dates in rules for customizable products. The arrival of a date does not cause the customizable product to be refreshed in the cache.

Cache Management for Siebel Product Configurator

This topic is part of *About Siebel Product Configurator Caching*.

Administrators use the Cache Administration view in the Administration - Product screen to manage mappings of customizable products or bundled promotions with remote Siebel Product Configurators. Cache management allows administrators to route customizable products (whether stand-alone products or products that are components of a bundled promotion) to remote Siebel Product Configurators.

When a user starts a configuration session, Siebel Product Configurator loads the requested customizable product into memory. You can specify a cache (server or component) to serve requests for frequently requested customizable products to improve response times. The specified cache loads the customizable products that are mapped to it into memory before any user requests are received.

You can also specify a time interval so that the specified cache updates the customizable products it holds at regular intervals. This reduces the possibility that a user request requires the retrieval of data from the database and the loading of a revised customizable product. To specify the time interval, you set values for the following parameters on the eProdCfgObjMgr component. For more information about these parameters, see *Parameters for Configuring Siebel Product Configurator Caching*.

- `Server Session Loop Sleep Time` (alias `ServerSessionLoopSleepTime`)
- `Product Configurator - Cache Engine Objects` (alias `eProdCfgCacheEngineObjects`)

Requests for other customizable products that are not mapped to a specific cache are served by a cache that has the setting *Explicit Product Mappings Only* disabled.

The following procedure describes how to configure product caching by mapping a product to a cache that has the setting *Explicit Product Mappings Only* enabled.

Cache Management Changes in Siebel CRM 21.8 Update

As of Siebel CRM 21.8 Update, the user interface in the Cache Administration view is enhanced to define routing of customizable products within a bundled promotion to remote instances of the Siebel Product Configurator Object Manager. The Cache Administration view and related repository objects are now nonextensible objects and cannot be customized by customers. See also *Configuring Session Pooling for Siebel Product Configurator*.

The Product applet in the Cache Administration view contains additional fields: Promotion Name, Component Description, and Promotion Rule Id. For a given product, the administrator can specify one or multiple associated bundled promotions using the Promotion Relationship pick applet.

> **Note:** To use this functionality, you must run the RepositoryUpgrade utility and update the Siebel repository with database schema changes, including those in the Cache Administration view. For more information, see *Siebel Database Upgrade Guide* and *Siebel CRM Update Guide and Release Notes* on My Oracle Support for your Siebel CRM 21.x Update release.

To configure product caching

1. Navigate to the Administration - Product screen, then the Cache Administration view.

   The Cache Administration view appears.
2. In the Cache applet, select a cache.

   > **Note:** Only one cache can be active at a time.
3. In the Cache Type field, select a value as described in the following list:
   - **Server.** Select Server as the cache type to route configuration requests to the Siebel Servers associated with the cache.
   - **Component.** Select Component as the cache type to route configuration requests to the components associated with the cache. These components can span multiple Siebel Servers depending on where components are active.

     For component cache-based routing, you must generate the lbconfig.txt file by executing the `generate lbconfig` command in Server Manager. Configure the Application Object Manager parameter `VirtualHostsFile` to point to this lbconfig.txt file. For example: `VirtualHostsFile = SIEBSRVR_ROOT\ADMIN\lbconfig.txt`

If you select Component as the cache type, then you must set the same value for the component parameter `Enable internal load balancing` (alias `EnableVirtualHosts`) on both the Siebel Application Object Manager and the eProdCfgObjMgr component. For example, if `EnableVirtualHosts` is set to True on the Siebel Application Object Manager component, then it must also be set to True on the eProdCfgObjMgr component.

4. In the Components applet, specify a Siebel Server name or a component name to associate with the cache that you selected in the preceding steps. For example, if you set Cache Type equal to Server, then you enter the name of a Siebel Server. If you set Cache Type equal to Component, then you enter the name of a component.

   Note: Component names must be unique in the Siebel Enterprise. Multiple eProdCfgObjMgr components can run on a single server. To create multiple remote eProdCfgObjMgr components, administrators can copy eProdCfgObjMgr and name the copy eProdCfgObjMgr2, for example.

5. If you want a server cache or component cache to only serve products that are mapped to that server cache or component cache, then select *Explicit Product Mappings Only*.
6. In the Product applet, select the customizable product(s) or promotion(s) that you want to associate with the component that you selected in the preceding steps.

   Note: When a promotion is selected, all of its components are added to the Products applet, with prepopulated values for the Promotion Name, Component Description, and Promotion Rule Id fields, in order to uniquely identify a specific component of the promotion. Product or system administrators can also route different components under a promotion to different cache components or servers. To do this, add a product and then select a promotion for it by specifying the Promotion Name in the Promotion Relationship applet, along with the appropriate promotion rule ID. The Promotion Relationship pick applet only displays the promotions that the product is a part of.

7. In the Cache applet, click Validate.

   The application validates that the Siebel Server or component names that you select are valid for the cache type that you specified in the preceding steps.
8. If the configuration that you created validates correctly, then click Release to enable the cache that you selected cache instances of the products that are mapped to it.

   The file ecfgserver.txt is created or updated when the cache is released.

Related Topics

Configuring Session Pooling for Siebel Product Configurator

Parameters for Configuring Siebel Product Configurator Caching

This topic is part of *About Siebel Product Configurator Caching*.

Siebel Product Configurator caching is enabled by default (`eProdCfgSnapshotFlg` is set to `TRUE`). Other parameters must be sized following guidelines such as those described in *Determining Rough Sizing for Caching Parameters*.

This topic lists the server parameters for configuring Siebel Product Configurator caching. Set these parameters on the Siebel Application Object Manager component for a Siebel Application Object Manager deployment of Siebel Product Configurator. For a dedicated Siebel Product Configurator server deployment, set these parameters on the Siebel Application Object Manager *and* on the eProdCfgObjMgr component, unless otherwise stated. For information on how to configure server parameters, see *Siebel System Administration Guide* .

Parameter Alias	Parameter Name	Data Type	Default Value	Description
eProdCfgCacheFS	Product Configurator - FS location	String	Not applicable	Specifies the location of the Siebel Product Configurator File System. If no value is specified for **eProdCfgCacheFS**, then Siebel Product Configurator looks in the Siebel File System. For more information, see *Specifying the Siebel Product Configurator File System Location* .
eProdCfgAttrSnapshotFlg	Product Configurator - Collect and Use the snapshots of the ISS_ATTR_DEF Ob	Boolean	**TRUE**	Set to **TRUE** to enable caching for attribute definitions. This caches attribute definitions in memory. It is strongly recommended that you do not change this parameter.
eProdCfgNumOfCachedAttrs	Product Configurator - Number of Attribute Definitions Cached in Memory	Integer	100	Sets the number of attribute definitions kept in memory at any given time during configuration.
eProdCfgClassSnapshotFlg	Product Configurator - Collect and Use the snapshots of ISS_CLASS_DEF Ob	Boolean	**TRUE**	Set to **TRUE** to enable caching for product class definitions. It is strongly recommended that you do not change this parameter.
eProdCfgNumOfCachedClasses	Product Configurator - Number of Class Definitions Cached in Memory	Integer	100	Sets the number of class definitions kept in memory at any given time during configuration.
eProdCfgProdSnapshotFlg	Product Configurator - Collect and Use the snapshots of ISS_PROD_DEF Ob	Boolean	**TRUE**	Set to **TRUE** to enable caching for product definitions. This caches product definitions in memory. It is strongly recommended that you do not change this parameter.
eProdCfgNumOfCachedProducts	Product Configurator - Number of Product Definitions Cached in Memory	Integer	1000	Sets the number of product definitions kept in memory at any given time during configuration.
eProdCfgSnapshotFlg	Product Configurator-Collect and use snapshots of the Cfg objects	Boolean	**TRUE**	Set to **TRUE** to turn on Siebel Product Configurator caching. It is strongly recommended that you do not change this parameter.
eProdCfgNumbOfCachedCatalogs	Product Configurator-Number of cached catalogs	Integer	10	Sets the maximum number of Model Manager catalogs that can be cached in memory. Catalog data is flushed from the cache on a least recently used basis.

Parameter Alias	Parameter Name	Data Type	Default Value	Description
eProdCfgNumbofCachedWorkers	Product Configurator- Number of workers cached in memory	Integer	50	Sets the maximum number of workers that can be cached in memory. This number applies to all Model Manager catalogs.
eProdCfgCacheEngineObjects	Product Configurator - Cache Engine Objects	Boolean	`TRUE`	Set to `TRUE` to enable content cache and precaching. **Note:** This parameter supports precaching. Set this parameter only on the dedicated Siebel Product Configurator server. Precaching does not apply on the Siebel Application Object Manager. See also *Cache Management for Siebel Product Configurator*.
ServerSessionLoopSleepTime	Server Session Loop Sleep Time	Integer	300	Specify an interval time (in seconds) to refresh cached products that are mapped to a Siebel Product Configurator server cache or component cache using the explicit product mapping setting. `Product Configurator - Cache Engine Objects` must be set to `TRUE`. **Note:** This parameter supports precaching. Set this parameter only on the dedicated Siebel Product Configurator server. Precaching does not apply on the Siebel Application Object Manager. See also *Cache Management for Siebel Product Configurator*.

Determining Rough Sizing for Caching Parameters

This topic is part of *About Siebel Product Configurator Caching*.

To help you to determine how to set the Siebel Product Configurator caching parameters, a general suggestion is to measure the incremental memory required for a customizable product.

Requirements for Model Manager and Worker caching are more relevant than those for object caching. Object caching has a small requirement, and applies to multiple users. Model Manager caching applies to multiple users (using the same customizable product). Worker caching also applies to multiple users.

You can try this on a Siebel Developer Web Client (similar to the Siebel Mobile Web Client, but using a dedicated database connection) by checking the memory used by the siebel.exe process before and after you click Customize for a customizable product included in a quote or order, and again after you have further configured the customizable product (to reach the maximum likely memory usage).

For example, X might be the before-loading memory size, Y might be the after-loading size, and Z might be the memory size after additional product configuration.

Of the incremental memory observed, consider the following breakdown:

- The size of a Model Manager for a customizable product is about 75% of the incremental memory required to instantiate the product (that is, 75% of Y minus X).
- The size of a Worker for a customizable product varies during runtime, generally increasing as user selections are made. This size can be approximated by subtracting the Model Manager size from the difference of Z less X.

Configuring Session Pooling for Siebel Product Configurator

This topic is part of *About Siebel Product Configurator Caching*.

As of Siebel CRM 21.8 Update, administrators can optionally configure session pooling for customizable products and multiple remote instances of the Siebel Product Configurator Object Manager. This feature is intended to provide better performance when agents or self-service users work with customizable products that are routed to remote Siebel Product Configurators.

Multiple customizable products within a single bundled promotion can be configured to be routed to multiple remote Siebel Product Configurators. Alternatively, all customizable products in a bundled promotion can be routed to the same remote Siebel Product Configurator. Customizable products or bundled promotions that are mapped in this way can now use a session pool.

> **Note:** In this guide, a remote Siebel Product Configurator and a dedicated Siebel Product Configurator mean the same thing: one or more instances of the Siebel Product Configurator Object Manager, wherever they are located within the Siebel CRM deployment.

How Session Pooling Works

During product configuration, a session pool is maintained between the Siebel Application Object Managers and the remote Siebel Product Configurators. The pooled sessions are reused, which reduces delays associated with frequent logins and logouts on the remote Siebel Product Configurators, compared to deployments that do not use session pooling. Call-handling agents can expect reduced response times and call times from this feature. Self-service users can also expect reduced response times.

With session pooling, a new session is created as needed to work with a customizable product on a remote Siebel Product Configurator (based on the mappings in the Cache Administration view), along with a corresponding session handle on the calling Application Object Manager. After the user completes work with the customizable product, the session is enqueued and is available to be reused for any customizable product that is mapped to the same remote Siebel Product Configurator. A session not in use remains enqueued unless it expires and becomes invalid, based on the

value of the `eProdCfgKeepAliveTime` parameter, which is described in *Running Siebel Product Configurator on Dedicated Servers*.

The session handle becomes invalid if the remote Siebel Product Configurator session becomes invalid. The session handle is also invalid if the remote Siebel Product Configurator goes offline, in which case a new session handle and session are created again later when the remote Siebel Product Configurator is up again.

For a subsequent session request, if existing sessions are in use or there are no enqueued sessions for any of the mapped remote Siebel Product Configurators, then a new session is created on one of the mapped remote Siebel Product Configurators, along with a corresponding session handle on the calling Application Object Manager.

Mappings in the ecfgserver.txt File

Although you define mappings in the Cache Administration view, during runtime the routing is achieved by referencing the ecfgserver.txt file, which is located in the `ISS_OBrkCache/CFGDefs` subdirectory in the Siebel File System. When you have set up the mappings, the ecfgserver.txt file is generated with entries using the following format: `eCfgServer=CP-RowId:Promo-RowId_Promo-RuleId,CP-RowId2:Promo-RowId2_Promo-RuleId2`, where:

- `eCfgServer` is the name of the remote Siebel Product Configurator Object Manager. For example, for component cache, this might be `eProdCfgObjMgr_enu` or `eProdCfgObjMgr2_enu`. For server cache, this might be `SiebelServer1` or `SiebelServer2`.
- `CP-RowId` (or `CP-RowId2`, and so on) is the row ID of the customizable product.
- `Promo-RowId` (or `Promo-RowId2`, and so on) is the row ID of the bundled promotion.
- `Promo-RuleId` (or `Promo-RuleId2`, and so on) is the row ID of the bundled promotion rule.

Alternatively, entries might specify only customizable products, as in the following format: `eCfgServer=CP-RowId,CP-RowId2` Each entry in the ecfgserver.txt file is terminated by a semicolon.

Event Logging for Session Pooling

Event logging for session pooling details the creation of a new RemoteCfgSession. The session pool queue size is logged when the following log event is set:

`eProdCfgLog=4`

System Preference for Session Pooling

Before you can configure session pooling for Siebel Product Configurator, you must enable the feature by setting the eProdCfg Session Pooling system preference to Y. The default setting is N, which disables session pooling for remote Siebel Product Configurators. Changing the value of this system preference requires that you restart the Siebel Enterprise, as described in *Siebel System Administration Guide* .

Related Topics

Cache Management for Siebel Product Configurator

Running Siebel Product Configurator on Dedicated Servers

Related Books

Siebel Database Upgrade Guide

Siebel CRM Update Guide and Release Notes on My Oracle Support

Siebel System Administration Guide

Administering the Siebel Product Configurator Cache

Siebel administrators or product administrators can refresh or update the Siebel Product Configurator cache in several ways. This topic describes procedures to refresh or update the Siebel Product Configurator cache with changes for customizable products, product classes, and attribute definitions.

This topic contains the following information:

- *Refreshing the Entire Siebel Product Configurator Cache*
- *Refreshing the Siebel Product Configurator Cache with Product Changes*
- *Updating the Siebel Product Configurator Cache with Product Class Changes*
- *Refreshing the Siebel Product Configurator Cache with Product Class Changes*
- *Updating the Siebel Product Configurator Cache with Product Class Changes*

See also *About Siebel Product Configurator Caching*.

Refreshing the Entire Siebel Product Configurator Cache

This topic is part of *Administering the Siebel Product Configurator Cache*.

You can refresh the Siebel Product Configurator cache with changes made to customizable products, product classes, and attribute definitions in one operation. Consider performing the task described here as part of standard maintenance for your Siebel Product Configurator deployment or, for example, if you intend to migrate data from a development to a production environment.

Related topics describe how you can update the Siebel Product Configurator cache with changes you made to customizable products, product classes, and attribute definitions in separate operations.

To refresh the entire Siebel Product Configurator cache

1. Navigate to the Administration - Product screen, then the Cache Administration view.
2. Select the record for the cache that you want to refresh.
3. Click the Menu button in the Cache list, then choose Refresh Product Cache.

Refreshing the Siebel Product Configurator Cache with Product Changes

This topic is part of *Administering the Siebel Product Configurator Cache*.

While editing a product record, a product administrator can select Refresh Product Cache to refresh the Siebel Product Configurator cache with changes made to the selected product. The next time a user requests the customizable product, the user receives a freshly instantiated version reflecting the product change and the cache is refreshed with this version. For example, you could change the product description and then refresh the cache.

To refresh the cache with product changes

1. Navigate to the Administration - Product screen.

2. Select the record for a customizable product that has been changed or that is to be refreshed.
3. Click the Menu button in the Products list, then choose Refresh Product Cache.

Updating the Siebel Product Configurator Cache with Product Class Changes

This topic is part of *Administering the Siebel Product Configurator Cache*.

While editing a product class record, a product administrator can select Update Cache to remove the version access keys of a cached product class from the Siebel Product Configurator cache for all versions of the selected product class. This forces the application to consult the database the next time it requires a version access key. A product administrator must select Update Cache if he or she modifies a nonversioned cached property such as, for example, a product name. If the product administrator does not select Update Cache, then a Siebel Application Object Manager that has a version already cached uses the old version.

To update the cache with class changes

1. Navigate to the Administration - Product screen, then the Product Classes view.

   The Product Classes list appears.
2. Select a product class and modify it or its attribute definitions as needed.
3. From the menu in the Product Classes list, choose Update Cache.

Refreshing the Siebel Product Configurator Cache with Product Class Changes

This topic is part of *Administering the Siebel Product Configurator Cache*.

While editing a product class record, a product administrator can select Refresh Cache to refresh the customizable products in the Siebel Product Configurator cache with changes made to the product class record. The next time a user requests the customizable product, the user receives a freshly instantiated version reflecting the product change and the cache is refreshed with this version. This new instance reflects the changes you made to the product class.

To refresh the cache with class changes

1. Navigate to the Administration - Product screen, then the Product Classes view.

   The Product Classes list appears.
2. Select a product class and modify it or its attribute definitions as needed.
3. From the menu in the Product Classes list, choose Refresh Cache.

Updating the Siebel Product Configurator Cache with Attribute Definition Changes

This topic is part of *Administering the Siebel Product Configurator Cache*.

While editing an attribute definition record, a product administrator can select Update Cache to remove the version access keys of an attribute definition record from the Siebel Product Configurator cache for all versions of the selected attribute definition. This forces the application to consult the database the next time it requires a version access key. A product administrator must select Update Cache if he or she modifies a nonversioned cached property such as, for example, an attribute definition name. If the product administrator does not select Update Cache, then a Siebel Application Object Manager that has a version already cached uses the old version.

To refresh the cache with attribute definition changes

1. Navigate to the Administration - Product screen, then the Attribute Definitions view.

   The Attribute Definitions list appears.
2. Select an attribute definition and modify it as needed.
3. From the menu in the Attribute Definitions list, choose Update Cache.

9 Tuning Siebel EAI for Performance

Tuning Siebel EAI for Performance

This chapter discusses tuning for Siebel Enterprise Application Integration (Siebel EAI) that might be required for optimal performance. It contains the following topics:

- *About Siebel Enterprise Application Integration*
- *Guidelines for Siebel EAI Tuning*

For more information about Siebel EAI, see the following documents on the *Siebel Bookshelf* :

- *Overview: Siebel Enterprise Application Integration*
- *Integration Platform Technologies: Siebel Enterprise Application Integration*
- *Transports and Interfaces: Siebel Enterprise Application Integration*
- *Business Processes and Rules: Siebel Enterprise Application Integration*
- *XML Reference: Siebel Enterprise Application Integration*
- *Siebel REST API Guide*

About Siebel Enterprise Application Integration

Siebel EAI provides components for integrating Siebel CRM with external applications and technologies within your company. Siebel EAI works with technologies, standards, or applications that include XML, HTTP, Java, REST, and various third-party middleware products and application integration solutions.

Siebel EAI provides bidirectional real-time and batch solutions for integrating Siebel CRM with other applications. Siebel EAI is designed as a set of interfaces that interact with each other and with other Siebel components.

Guidelines for Siebel EAI Tuning

This topic describes guidelines for maintaining acceptable performance using Siebel EAI.

General guidelines are followed by recommendations specific to Siebel EAI features such as IBM WebSphere MQ (formerly MQSeries) Transport adapter, HTTP Inbound Transport adapter, EAI Siebel Adapter, virtual business components, and Workflow Process Manager used with Siebel EAI.

Follow these general guidelines to improve overall performance for data integration and throughput of Siebel CRM:

- Minimize round trips between systems. For example, if an integration needs to request three pieces of data, do not send a request for one piece of data, wait for the response, and then send the next request. If you need multiple pieces of data, gather the data in a single request.

- Keep processing in a single session wherever possible, to avoid having to make calls between server components.
- Within a session, minimize the nesting of calls between components such as workflow, scripting, and the EAI Siebel Adapter. For example, use a workflow process to sequence the calling of business services and keep scripting code in self-contained steps. Workflow subprocesses can be used to package together commonly called sequences of services.
- Use alternatives to scripting, where possible. If you use scripting, then use it minimally and economically and apply documented guidelines. For more information, see *Guidelines for Siebel Scripting*.
- Configure business components, business services, caching, and other application functionality that supports integration processing to obtain optimal performance. For more information, see other topics in this chapter and see *Tuning Customer Configurations for Performance*.
- Perform capacity planning for all servers that support integration processing. For sizing reviews, consult Oracle Advanced Customer Services. Contact your Oracle sales representative to request assistance from Oracle Advanced Customer Services.
- Represent the incoming external data in the same code page and encoding that the Siebel application uses internally (UCS-2). This eliminates the need to use the Transcode business service in your workflow process, thus improving performance.

The following topics discuss specific technologies and what you can do to improve performance in each area:

- *Tuning the Anonymous Pool*
- *Improving IBM WebSphere MQ Transport Performance*
- *Improving HTTP Inbound Transport Performance*
- *EAI Siebel Adapter Performance*
- *Virtual Business Component Performance*
- *Improving Workflow Process Manager Performance*
- *Other Guidelines for Siebel EAI*

Tuning the Anonymous Pool

When you configure the Siebel Application Interface profile using Siebel Management Console, as described in *Siebel Installation Guide* , you can configure the size of the anonymous pool used for inbound Siebel EAI requests. The Anonymous Pool Size field appears when you select both Configure EAI HTTP Inbound Transport and Configure Anonymous Pool during configuration.

The anonymous pool is a set of EAI tasks used for WS-Security SOAP Web services. It provides a set of EAI threads and tasks ready for use and reuse, over and over. Tasks are prestarted and logged in as the anonymous user. As new requests come in, they impersonate and run on the context of the user making the request, based on the SOAP header, then switch back to the anonymous user and stay idle, ready for reuse by a new request.

Without load balancing, there is a 1:1 mapping to EAI servers (Siebel Servers), so the upper limit of EAI threads is theoretically the same as the setting of MaxTasks. However, when you size the anonymous pool, you must also account for requests that use the EAI framework but are not otherwise EAI requests, including REST, JCA, and session management SOAP calls. Therefore, the upper limit of EAI threads is actually MaxTasks minus these other non-anonymous EAI use cases.

With load balancing, one anonymous pool on the Siebel Application Interface might map to more than one EAI server. In this case, the upper limit is the combination of the MaxTasks values for all of the available servers, again discounting non-anonymous EAI use cases. The larger the anonymous pool, the larger the memory requirements on the Siebel Application Interface. And, the larger the anonymous pool, the more EAI tasks are started that wait to be used, and

the larger the corresponding impact on memory consumption for the EAI server. With multiple instances of Siebel Application Interface, the effective size of the combined anonymous pool is the sum of the pool size on each instance. See also the other topics in this guide that discuss MaxTasks.

You can further adjust load balancing behavior through configuration of third-party load balancers and Web servers, as well as configuration of Siebel Application Interface profiles.

No single log indicates the current effective anonymous pool size. However, you can derive this information from the Siebel Application Interface statistics page, by looking at the EAI Object Manager correlated with the anonymous user name, which is appended to the session string. For an example of a statistics page, see the chapter about configuring Siebel Application Interface logging and monitoring in *Siebel System Monitoring and Diagnostics Guide* . For example, in the following string from a statistics page, 900017 is the task ID on the EAI server:

```
siebel.TCPIP.NONE.NONE://SIEBSRVR_HOST:2321/siebel/EAIObjMgr_enu/!1.2a40.900017.5d1 60ee4anonuser
```

Improving IBM WebSphere MQ Transport Performance

This topic is part of *Guidelines for Siebel EAI Tuning*.

The performance of an IBM WebSphere MQ queue is highly dependent on the disk performance of the queue manager computer and the layout of the queue's files on the disk. Test your queue with stand-alone utilities so that you have an upper boundary for the performance that can be expected in a live application.

To achieve higher throughput, consider the following options:

- **Run multiple MQ Receiver tasks.** Run multiple MQ Receiver tasks in parallel on the same computer or across several computers. The optimal number of MQ Receiver tasks depends on the transaction type.

  > **Note:** This guide refers to *MQ Receiver*, where the actual Siebel Server component that you are using might be MQSeries Server Receiver (alias MqSeriesSrvRcvr) or MQSeries AMI Receiver (alias MqSeriesAMIRcvr).

  The default number of MQ Receiver tasks is 1. You can set this to 10 or more, depending on the nature of your transactions and on available server capacity.

  Adding MQ Receivers is generally most helpful for handling CPU-bound transactions, where the dequeuing rate is low and MQ contention is not experienced.

  Sometimes contention is still experienced after adding MQ Receiver tasks, such as when multiple MQ Receivers connect to the same MQ queue manager or queue. See the next item for more information.
- **Run multiple MQ queue managers.** If you experience diminishing returns from adding MQ Receiver tasks, then you might benefit from running additional MQ queue managers. Doing so can help to reduce contention of MQ resources stored in physical folders on disk.
- **Turn off persistent queuing if it is unneeded.** Performance issues for non CPU-bound transactions or for persistent queuing are often related to MQ contention, which is not helped by adding receivers. If you do not require persistent queuing, then turn it off.

  Persistent queuing is significantly slower than normal queuing for WebSphere MQ. If you do not use this feature, however, then messages will be lost if the queue manager goes down.
- **Set Maximum Number of Channels parameter.** Set the `Maximum Number of Channels` parameter in the WebSphere MQ queue manager to be greater than or equal to the maximum number of simultaneous clients that you have running.

In addition, you can take specific actions to improve WebSphere MQ Transport performance for outbound and inbound transports, as detailed in this topic.

Inbound Messages

For inbound WebSphere MQ messages, run multiple MQ Receivers in parallel to increase throughput.

Outbound Messages (Send, SendReceive)

Caching of WebSphere MQ Transport business services can improve outbound performance by eliminating the need to connect to the queue for each message. Caching is disabled by default because it is not usable in every situation. Follow these tips to enable caching:

- Cache in client sessions only. Do not use caching if your transport will be called within the Workflow Process Manager (WfProcMgr) component. The threading model of this component is not compatible with the WebSphere MQ APIs.
- To enable caching for a business service, set the Cache property to `TRUE` in Siebel Tools, then update and publish the Siebel runtime repository.
- If you need to call the WebSphere MQ Transport in Workflow Process Manager and in a client session, then make a separate copy of the service (one cached and one uncached) for each situation.
- Caching occurs on a per-queue basis and only one connection is kept open at a time. If a single session is going to talk to multiple queues, then consider making a copy of the transport for each outbound queue.

Note: See your IBM WebSphere MQ documentation for performance and sizing guidelines.

Performance Events

You can get detailed performance tracing of the WebSphere MQ Transport by setting the `EAITransportPerf` event to level 5.

You can set this event level for multiple Siebel Server components that play a role in Siebel EAI functionality, including Workflow Process Manager (WfProcMgr), EAI Object Manager (EAIObjMgr), MQ Receiver, or other components. For example, you can use srvrmgr to set the event level for MQ Receiver:

```
change evtloglvl EAITransportPerf=5 for comp MqSeriesSrvRcvr
```

Improving HTTP Inbound Transport Performance

This topic is part of *Guidelines for Siebel EAI Tuning*.

The HTTP Inbound Transport supports both persistent sessions and intermittent sessions:

- With persistent sessions, the session stays live until a logout call is made.
- With intermittent sessions (where SessionType in the SOAP header = None), logging in and logging out occur automatically for each request.

Use persistent sessions whenever possible, because the time required to log in to the application is usually significantly longer than the time required to process an average request.

You can also use the `SessPerSisnConn` component parameter to control the number of sessions sharing the same physical Siebel Internet Session Network Application Programming Interface (SISNAPI) connection between the Siebel Application Interface and the EAI Object Manager.

Setting this parameter to 1 provides a dedicated physical connection for each Siebel session. The default value is 20, to allow up to 20 sessions to share the same SISNAPI connection. For the EAI Object Manager, it is recommended that you set `SessPerSisnConn` to 1. If setting `SessPerSisnConn` to 1 results in an excessive number of sessions, then consider increasing the value of `SessPerSisnConn` or provide additional hardware resources.

You can change this parameter using srvrmgr at the Enterprise or Server level. For example, to set the parameter at the Enterprise level for the EAI Object Manager, you enter the following command:

```
change param SessPerSisnConn=1 for compdef eaiobjmgr_enu
```

> **CAUTION:** If you set `SessPerSisnConn` to a value of 1, then you must set the parameter `ConnIdleTime` to a value other than -1 (its default value). Otherwise, this combination of settings might cause the Siebel Application Interface to stop running.

For more information about configuring `SessPerSisnConn`, see *Configuring SISNAPI Connection Pooling for Siebel Application Object Managers*.

Related Books

Integration Platform Technologies: Siebel Enterprise Application Integration

Transports and Interfaces: Siebel Enterprise Application Integration

EAI Siebel Adapter Performance

This topic is part of *Guidelines for Siebel EAI Tuning*.

Use the techniques described here to improve the EAI Siebel Adapter performance and throughput.

Reviewing Scripting

Avoid scripting events on business components used by the EAI Siebel Adapter. Perform any scripting task either before or after the EAI Siebel Adapter call, rather than within it. For general scripting guidelines, see also *Guidelines for Siebel Scripting*.

Disabling Logging

Disable logging for performance-critical processes that are functioning correctly to gain about 10% faster performance. You can disable logging for the EAI Object Manager (or other applicable server components, such as MQ Receiver) by setting the `BypassHandler` parameter to `TRUE`.

Minimizing Integration Object Size

The size of an integration object and its underlying business components can have an impact on the performance of the EAI Siebel Adapter. To minimize this impact, you can:

- Consider copying business objects and business components and modifying them to remove any elements (such as scripts, joins, multi-value fields, user properties, and so on) that you do not require in the Siebel EAI

context. Base your integration objects on these relatively streamlined object definitions. Verify that user keys on your integration objects make effective use of indexes when queries are performed.

- Inactivate unneeded integration components and integration component fields in your integration objects. Activate only the components and fields needed for message processing, according to your business needs.
- Inactivate unneeded fields for each underlying business component. For fields that are unneeded, if `Force Active` is set to `TRUE`, then set it to `FALSE`. Setting `Force Active` to `FALSE` prevents the EAI Siebel Adapter from processing these fields. If you do not inactivate these fields, then the adapter processes them even when they are not actually included in the integration object.

For more information, see *Limiting the Number of Active Fields*.

Analyzing SQL Produced by EAI Siebel Adapter

Requests to the EAI Siebel Adapter eventually generate SQL to be executed against the Siebel database. By setting the event `SQL` to level 4 in the component running in the EAI Siebel Adapter, you can get a trace of the SQL statements being executed, along with timings for each statement, in milliseconds.

You can get timings for each EAI Siebel Adapter operation by setting the event `EAISiebAdptPerf` to `4` or `5`. Do this to correlate the EAI Siebel Adapter calls with their associated SQL.

After you have this information, look through the logs to find any SQL statements taking significantly longer than average. To improve the performance of such statements, look at the business component (perhaps eliminating unnecessary joins and fields) or at the physical database schema (perhaps adding indexes).

> **Note:** The overall timing across operations (equivalent to the `TotalTimeForProcess` event) cannot be determined by adding the individual logged values associated with the `EAISiebAdptPerf` event, because the EAI Siebel Adapter requires some additional overhead. Overhead is greater when `EAISiebAdptPerf` is set to a high value. Set this event to a lower value for a production system for optimal performance.

Running EAI Siebel Adapter in Parallel

A common technique to improve throughput is to run multiple instances of the EAI Siebel Adapter in parallel.

For the MQ Receiver, you do this by running multiple receiver tasks. For more information, see *Improving IBM WebSphere MQ Transport Performance*.

For the EAI Object Manager, you do this by setting the `MaxTasks`, `MaxMTServers`, and `MinMTServers` parameters, in order to run more threads (tasks) on more multithreaded processes for the EAI Object Manager component. Also start multiple simultaneous HTTP sessions. There is little interaction between each instance of the EAI Siebel Adapter.

If the Siebel database server is large enough, then almost linear scalability of the EAI Siebel Adapter is possible until either the limits of the CPU or the memory limits of the Siebel Server are reached.

> **CAUTION:** If two sessions attempt to simultaneously update or insert the same record, then one will succeed and one will produce an error. Therefore, when running the EAI Siebel Adapters in parallel, you must prevent the simultaneous update of the same record in multiple sessions. You can prevent this by either partitioning your data or retrying the EAI Siebel Adapter operation where the error occurs.

Caching Business Objects

The EAI Siebel Adapter caches business objects by default. The default cache size is five objects. Using caching, subsequent runs on the adapter are significantly faster because the business objects do not need to be re-created for each run.

Use the `BusObjCacheSize` parameter on the EAI Siebel Adapter to change the size of the cache, if required. However, the five-object cache size is enough for most purposes. Making this number too large creates an unnecessarily large memory footprint.

Virtual Business Component Performance

This topic is part of *Guidelines for Siebel EAI Tuning*. Because users must wait for the virtual business component (VBC) response to display the GUI component for the integration on their screens, this type of integration is especially sensitive to latency. To improve virtual business component performance when your integration has multiple requests, put the requests for a given system in a single batch.

Improving Workflow Process Manager Performance

This topic is part of *Guidelines for Siebel EAI Tuning*. It discusses performance issues for the Workflow Process Manager component. For more information about Siebel Workflow performance, see *Tuning Siebel Workflow for Performance* Also see *Siebel Business Process Framework: Workflow Guide* .

Workflow Process Manager is a task-based server component. A new thread is created for each request. However, sessions for Object Manager components (such as EAI Object Manager or Siebel Application Object Managers) that can invoke workflow processes are cached and reused for subsequent requests. When sizing a system, consider the maximum number of workflow tasks that you expect to have active at a given time. This determines the maximum number of Object Manager sessions that Siebel applications create. In general, creating smaller workflow processes is recommended. If you cannot avoid creating a large workflow process, then divide the workflow process into subprocesses.

CPU and Memory Consumption

The exact CPU and memory consumption of each task depends on the actions performed in your workflow processes. To estimate CPU and memory consumption in your production environment, run a single task, measure its resource consumption, and make an estimation based on your maximum concurrent sessions. Take session caching into account when making these measurements.

If you need a large number of sessions, then you might want to run Workflow Process Manager on multiple Siebel Server computers. You can then use Siebel Server load balancing to load-balance requests across the Siebel Servers. If you plan to run a significant number of tasks per server (such as 100 or more), then you might also want to run multiple multithreaded processes.

If you are going to run several different types of workflows, then run each type in a separate process. This makes it easier to monitor the overall CPU and memory usage of each process type.

The number of multithreaded processes and the number of tasks per process are controlled through the parameters `MaxMTServers` (`Maximum MT Servers`), `MinMTServers` (`Minimum MT Servers`), and `MaxTasks` (`Maximum Tasks`).

Note: These parameters are per Siebel Server. For example, `MaxMTServers` refers to how many multithreaded processes to run on each Siebel Server computer. For details, see *Siebel System Administration Guide* .

Performance Events

You can get performance tracing of workflows by setting the event `WfPerf` for the component in which your workflow is running. Setting the event to level 4 gives timing for the execution of the overall process. Setting the event to level 5 provides timing for each step as well.

You can set this event level for any Siebel Server component that invokes a workflow process as part of Siebel EAI functionality. For example, to set this event level for the MQ Receiver using srvrmgr, enter the following:

```
change evtloglvl WfPerf=5 for comp MqSeriesSrvRcvr
```

These events can be useful not just for measuring workflow performance but also for measuring the performance of business services executed within these workflows.

Other Guidelines for Siebel EAI

This topic is part of *Guidelines for Siebel EAI Tuning*. Review the following issues for applicability to your deployment, for optimizing Siebel EAI performance:

- **Check disks on the computer.** Do a preliminary test on the queue manager that you are using to see how many sends and corresponding receivers it can support per second (use multiple drivers). Queue vendors such as IBM WebSphere MQ provide test programs that you can use to drive these and determine how much the queue itself can scale. The speed of the disks on the computer is important.
- **Optimize messages.** In the messages, reference only the columns that you require.
- **Create smaller business components.** Messages might use only a small portion of the actual business components.

  Create copies of the business components that you are using. In the copies, keep active all fields used by the optimized integration object or otherwise used for correct processing of messages (like the visibility fields or status fields). Deactivate all other fields. Also deactivate the join definitions and multi-value links (MVLs) that are not needed for processing of the messages.

  The original business components are often large and complex and contain elements that you will not need for your integration purposes. Use the smaller business components and business objects and links created when creating the optimized integration object.

  Business components can have fields with `Force Active` set to `TRUE`. Check this property for fields in the business components, using Siebel Tools. If the fields are not needed, then set `Force Active` to `FALSE`.
- **Set user property All Mode Sort to FALSE.** Set the user property All Mode Sort to `FALSE` for optimized business components (if not already set). Do this only for the smaller business components created for use with Siebel EAI, because this user property changes the order in which rows are retrieved, which might not be appropriate or normal clients. For more information about All Mode Sort, see *Siebel Developer's Reference* .
- **Optimize database queries.** Review queries generated by the receiver process and verify that they are optimized.
- **Turn off logging.** Turn off server-side logging that you do not require.

10 Tuning Siebel EIM for Performance

Tuning Siebel EIM for Performance

This chapter describes recommended guidelines for improving the performance of Siebel EIM. It contains the following topics:

- *About Siebel EIM*
- *Siebel EIM Architecture Planning Requirements*
- *Siebel EIM Usage Planning*
- *General Guidelines for Optimizing Siebel EIM*
- *Recommended Sequence for Implementing Siebel EIM Processes*
- *Troubleshooting Siebel EIM Performance*
- *Database Guidelines for Optimizing Siebel EIM*
- *Data Management Guidelines for Optimizing Siebel EIM*
- *Run Parameter Guidelines for Optimizing Siebel EIM*
- *Monitoring the Siebel Server During a Siebel EIM Task*

About Siebel EIM

Siebel Enterprise Integration Manager (Siebel EIM) is a server component in the Siebel EAI component group that transfers data between the Siebel database and other corporate data sources. This exchange of information is accomplished through intermediary tables called EIM tables. (In earlier releases, EIM tables were known as interface tables.) The EIM tables act as a staging area between the Siebel application database and other data sources.

Siebel EIM is your primary method of loading mass quantities of data into the Siebel database. Use Siebel EIM to perform bulk operations to import, update, merge, or delete data. For more information about Siebel EIM, see *Siebel Enterprise Integration Manager Administration Guide* .

In the Siebel database, there are application tables (known as base tables), which Siebel CRM uses. For data to come from other corporate data sources (external databases) into Siebel application tables, the data must go through EIM tables. So, the data exchanges between the Siebel database and external databases occurs in two phases:

1. Load data into EIM tables.
2. Run Siebel EIM to import the data from the EIM tables into the Siebel base tables.

> **Note:** While the first phase of this data-exchange process involves the intermediary tables that are called EIM tables, only the second phase involves the functionality of Siebel EIM.

When data is entered through the Siebel user interface, the application references properties that are set at the business component object type. However, when data is entered into Siebel base tables through Siebel EIM, EIM references properties that are set at the table object type.

Note: You must use Siebel EIM to perform bulk imports, exports, merges, and deletes, because it is not supported to use native SQL to load data directly into Siebel base tables (the tables targeted to receive the data). Additionally, be aware that Siebel EIM translates empty strings into NULL.

Siebel EIM Architecture Planning Requirements

You must consider the size and complexity of the implementation before executing any single item with the Siebel application. Aspects that have a direct impact on how the production application will perform might not be your highest priority when you begin your Siebel implementation. However, the decisions made during the initial phases of an implementation have a far reaching impact, not only on performance and scalability but also on the overall maintenance of the Siebel application.

It is strongly recommended to have a Siebel certified principal consultant or architecture specialist from Oracle Advanced Customer Services involved in designing the most effective logical and physical architecture for your organization. This includes capacity planning and system sizing, physical database layout, and other key architecture items. Contact your Oracle sales representative to request assistance from Oracle Advanced Customer Services.

For more information, see the following:

- *Database Sizing Guidelines*
- *Database Layout Guidelines (Logical and Physical)*

Database Sizing Guidelines

This topic is part of *Siebel EIM Architecture Planning Requirements*.

One of the most important factors to determine about the database is its overall size. During the planning phase, you need to allocate space for system storage, rollback segments and containers, temporary storage space, log files, and other system files required by the relational database management system (RDBMS), as well as space for the Siebel application data and indexes. If you allocate too little space for the system, then performance will be affected and, in extreme cases, the system itself can be halted.

The space needed by the database depends on the total number and types of supported users. It is recommended that you consult your vendor RDBMS technical documentation for more information about these requirements.

The space required for Siebel data and indexes depends on the functionality being implemented and the amount and nature of data supporting this functionality.

The process for making accurate database size calculations is a complex one involving many variables. Use the following guidelines:

- Determine the total number, and types, of users of Siebel CRM (for example, 500 sales representatives and 75 sales managers).
- Determine the functionality that you will implement and the entities required to support them. Typically, the largest entities are as follows:
    - Accounts
    - Activities
    - Contacts

    - Forecasts
    - Opportunities
    - Service Requests

- Estimate the average number of entities per user (for example, 100 accounts per sales representative) and calculate an estimated total number of records per entity for the total user base.
- Using standard sizing procedures for the specific database, and *Siebel Data Model Reference* on My Oracle Support (Article ID 2285310.1), calculate the average record size per entity and multiply by the total number of records. Typically, these entities span multiple physical tables, all of which must be included in the row size calculation. This determines the estimated data sizes for the largest entities.
- You must add additional space for the storage of other Siebel application data. A rough guideline for this additional amount would be one-half the storage required for these key entities.
- Indexes typically require approximately the same amount of space as data.
- Be sure to allow for a margin of error in the total size calculation.
- Be sure to factor growth rates into the total size calculation.

Database Layout Guidelines (Logical and Physical)

This topic is part of *Siebel EIM Architecture Planning Requirements*.

The overall performance of Siebel CRM largely depends on the input/output (I/O) performance of the database server. To achieve optimal I/O performance, it is critical that the tables and indexes in the database be arranged across available disk devices in a manner that evenly distributes the I/O load.

The mechanism for distributing database objects varies by RDBMS, depending on the manner in which storage space is allocated. Most databases have the ability to assign a given object to be created on a specific disk. These objects, and guidelines for some of them, are provided in the following list.

A redundant array of independent disks, or RAID, can provide large amounts of I/O throughput and capacity, while appearing to the operating system and RDBMS as a single large disk (or multiple disks, as desired, for manageability). The use of RAID can greatly simplify the database layout process by providing an abstraction layer above the physical disks while ensuring high performance.

Regardless of the implemented RDBMS and the chosen disk arrangement, be sure that you properly distribute the following types of database objects:

- Database log or archive files.
- Temporary workspace used by the database.
- Tables and Indexes: In most implementations, the tables and corresponding indexes in the following list tend to be some of the more heavily used and must be separated across devices. In general, the indexes listed in the following table must be placed on different physical devices from the tables on which they are created.

Table Name	Table Name
S_ACCNT_POSTN	S_PARTY_REL
S_OPTY	S_PARTY

Table Name	Table Name
S_ADDR_ORG	S_SRV_REQ
S_OPTY_POSTN	S_EVT_ACT
S_CONTACT	S_OPTY
S_POSTN_CON	S_ORG_EXT
S_DOCK_TXN_LOG	

Note: If you plan on making extensive use of Siebel EIM, then put the key EIM tables (based on the unique business requirements) and their corresponding indexes on different devices from the Siebel base tables and indexes, because all of them are accessed simultaneously during Siebel EIM operations.

Siebel EIM Usage Planning

This topic provides several general guidelines for effective and efficient implementations of Siebel EIM, regardless of the size of the overall Siebel implementation. You must take a strategic perspective when implementing Siebel EIM to make sure that your deployment is successful.

For more information, see the following:

- *Defining the Siebel EIM Team*
- *Mapping Data into Siebel CRM*
- *Testing Siebel EIM Processes*

Defining the Siebel EIM Team

This topic is part of *Siebel EIM Usage Planning*.

Based on customer experience, it is recommended that a team of individuals be assigned to manage and maintain the Siebel EIM processes required for your organization. Consider using individuals with the following skill sets:

- For small to medium-sized Siebel CRM implementations:
    - A database administrator with a detailed understanding of not only the RDBMS used by your organization, but also the Siebel Data Model. This individual would be responsible for identifying the actual data to be loaded into the EIM tables and making sure that the physical layout of the database provides optimal performance. This person would also be responsible for the task of mapping the data into the Siebel base tables. For more information about performing this task, see *Siebel Enterprise Integration Manager Administration Guide* .
    - A system administrator with a strong background in the systems used by your organization. This individual would be responsible for developing scripts unique to your organization to automate the

loading of data into the EIM tables, and to execute Siebel EIM in order to process the data into the Siebel base tables.

> **Note:** Your organization might have one individual with both these skill sets and so you might rather dedicate only a single individual to these tasks. If this is the case, then consider having a backup person, so that when this primary individual is unavailable, the backup person is capable of performing what needs to be done to keep the Siebel implementation operational.

- For larger to very large-sized Siebel implementations:
  - A database administrator with a detailed understanding of not only the RDBMS used by your organization, but also the Siebel Data Model. This individual would be responsible for identifying the actual data to be loaded into the EIM tables and for making sure that the physical layout of the database provides optimal performance. This team member would also be responsible for the crucial task of mapping the data into the Siebel base tables. For more information about performing this task, see *Siebel Enterprise Integration Manager Administration Guide* .
  - A system administrator with a strong background in the systems (both the database server and application server) used by your organization. This individual would be responsible for developing scripts unique to your organization to automate the loading of data into the EIM tables, and to execute Siebel EIM in order to process the data into the Siebel base tables.
  - A business analyst with a strong understanding of the Siebel Data Model and its intended usage in the Siebel implementation. This individual would act as a liaison between the business and technical members of the Siebel EIM team.

Mapping Data into Siebel CRM

This topic is part of *Siebel EIM Usage Planning*.

Siebel EIM uses EIM table mappings to map columns from EIM tables to Siebel base tables. Predefined Siebel EIM mappings are fixed and cannot be remapped.

> **Note:** Siebel EIM uses only EIM table mappings to determine table relationships. Siebel EIM does not use configuration logic in the Siebel repository to determine table relationships.

Using Siebel Tools, you can view:

- Mappings of EIM tables to Siebel base tables
- Mappings of EIM table columns to Siebel base table columns
- Mappings of Siebel base tables to EIM tables
- Mappings of Siebel base table columns to EIM table columns

Some base tables might not be mapped to a corresponding EIM table. In such cases, use Siebel Visual Basic (VB) to load data into these base tables and inform Global Customer Support regarding the missing mapping. For information about using Siebel VB, see *Siebel VB Language Reference* .

If you have licensed Database Extensibility and created extensions, you can use the Column Mapping screen to specify mappings to the new fields. Database Extensibility and Siebel EIM support mappings between columns in extension tables and in EIM tables only if these columns share the same base table. To map EIM table extensions to base table extensions, you must specify which column the extended field will point to in the base table. For more information about Database Extensibility, see *Configuring Siebel Business Applications* .

CAUTION: Manually mapping new extension columns to columns in EIM tables presents risks of errors during Siebel EIM execution. Whether or not you have licensed Database Extensibility, it is strongly recommended for you to request an EIM Data Mapping and Design review or other assistance from Oracle Advanced Customer Services to help you perform the necessary tasks. This review can be used to make sure that the EIM mappings are correct and will accomplish intended goals. Contact your Oracle sales representative to request assistance from Oracle Advanced Customer Services.

To map data into a Siebel application

1. Determine which Siebel base table columns need to be populated for the Siebel implementation, along with the external data that will be loaded into these base tables.
2. Determine which EIM table and columns will be used to import from the source to the destination.
3. Analyze this external data to determine which attributes need to be stored and the relationship this data has to other entities.

Testing Siebel EIM Processes

This topic is part of *Siebel EIM Usage Planning*.

Fully and completely testing Siebel EIM processes must not be overlooked. Testing is more than simply mapping the data and then running a Siebel EIM process using the default Siebel EIM configuration file. Complete testing requires you to run a large number of identical Siebel EIM jobs with similar data. Testing in this way allows you to not only find areas that you might have overlooked, but also provides some insight into optimal sizing of the Siebel EIM batches and exposure to scenarios that can occur in a production environment.

Before using Siebel EIM, a database administrator must populate the EIM tables with data to be processed by Siebel EIM. Then, you can invoke Siebel EIM to process this data, with Siebel EIM making multiple passes through the tables to complete the specified process.

Siebel EIM reads a special configuration file that specifies the Siebel EIM process to perform (import, merge, delete, or export) and the appropriate parameters. The Siebel EIM configuration file (the default file is default.ifb) is an ASCII text file of extension type IFB that resides in the `admin` subdirectory under the Siebel Server directory. Before running a Siebel EIM process, you must edit the contents of the Siebel EIM configuration file to define the processes that Siebel EIM will perform.

The Siebel EIM log file can contain information at different levels of detail, depending on the values of three flags: the Error flag, the SQL flag, and the Trace flag. For more information about these flags, see *Siebel Enterprise Integration Manager Administration Guide* . Some of the recommended settings are described in the following list:

- As a starting point, it is recommended to set the Error flag = 1, the SQL flag = 1, and the Trace flag = 1. These settings will show errors and unused foreign keys. Setting the Trace flag = 1 will provide a summary (after each batch) of the elapsed time after Siebel EIM updates primary child relationships in the Siebel database tables as necessary and runs optional miscellaneous SQL statements.
- Set the Error flag = 1, the SQL flag = 8, and the Trace flag = 3. These settings will produce a log file with SQL statements that include how long each statement took, which is useful for optimizing SQL performance.
- Set the Error flag = 0, the SQL flag = 0, and the Trace flag = 1. These settings will produce a log file showing how long each Siebel EIM step took, which is useful when figuring out the optimal batch size, as well as useful for monitoring any performance deterioration in a particular step.

General Guidelines for Optimizing Siebel EIM

The following guidelines are recommended for improving Siebel EIM performance:

- Verify that all indexes exist for the tables involved. Keep in mind, however, that for large loads you must drop most of the indexes from the target tables to increase the speed of the process, rebuilding those indexes afterward when the process is finished.
- In some cases, custom indexes might be necessary on EIM tables or base tables, to expedite processing and reduce elapsed time.
- Limit tables and columns to be processed using `ONLY BASE TABLES/COLUMNS` configuration parameters to minimize Siebel EIM processing.
- Set the system preference that enables transaction logging during the initial database load. This setting (in the Administration - Siebel Remote screen, and Remote System Preferences view) is a check box labeled Enable Transaction Logging. This setting reduces transaction activity to the Siebel docking tables, which are used for synchronizing mobile clients
- Consider disabling transaction logging during the Siebel EIM run. Turning off transaction logging improves performance; however, this benefit must be balanced with the need for mobile users to reextract afterward.
- Altering batch sizes to find the optimal batch size for a given business component typically helps resolve performance issues. The batch size is dependent upon the quantity of data and which type of Siebel EIM process that you are running.

  Note: Although the limit of rows that you can process is directly related to the capabilities of your database server, executing batches greater than 100,000 rows is strongly discouraged.

- For Siebel EIM delete processes that use the `DELETE EXACT` parameter, use a batch size of 20,000 rows or less.
- Try using batch ranges (BATCH = *X*-*Y*). This allows you to run with smaller batch sizes and avoid the startup overhead on each batch. The maximum number of batches that you can run in a Siebel EIM process is 1,000.
- Perform regular table maintenance on EIM tables. Frequent insert or delete operations on EIM tables can cause fragmentation. Consult your database administrator to detect and correct fragmentation in the EIM tables.
- Delete batches from EIM tables on completion. Leaving old batches in the EIM table wastes space and could adversely affect performance.
- Run independent Siebel EIM jobs in parallel.
- Set the `USING SYNONYMS` parameter to `FALSE` in the IFB file to indicate that account synonyms do not need to be checked.
- If no other strategy appears to be successful, then use the `SQLPROFILE` parameter to identify slow-running steps and queries. For more information, see *Using the SQLPROFILE Parameter*.

Recommended Sequence for Implementing Siebel EIM Processes

The following sequence is recommended for implementing Siebel EIM processes:

1. Customize and test the IFB file to meet the business requirements.

2. Tune the IFB file parameters.
3. Separate the Siebel EIM processes.
4. Set the database parameters, making sure that the basic requirements are met, including the hardware, the settings, and no or minimal fragmentation.

Before you start optimizing Siebel EIM processes, make sure there are no network problems or server performance problems that can affect the results. Oracle Advanced Customer Services recommends using at least 100 MB network segments and network-interface cards (NICs) to connect the Siebel Server and the Siebel database, and also recommends using a network switch or similar technology, rather than a hub, to maximize throughput. Contact your Oracle sales representative to request assistance from Oracle Advanced Customer Services.

This topic contains the following information:

- *Optimizing the IFB File for Siebel EIM*
- *Checking IFB File Optimization for Siebel EIM*
- *Separating Siebel EIM Processes by Operation*

Optimizing the IFB File for Siebel EIM

This topic is part of *Recommended Sequence for Implementing Siebel EIM Processes*.

When you have finished coding and testing the IFB file to meet your business requirements, the next step is to optimize the IFB file. The selected parameters in each section of the IFB file determine the focus of each Siebel EIM task. The following recommendations are provided for each section of the IFB file:

- **ONLY BASE TABLES** or **IGNORE BASE TABLES.** These parameters specify and restrict the selected base tables for the Siebel EIM process. A single EIM table (sometimes referred to as an interface table) is mapped to multiple user or base tables. For example, the table `EIM_ACCOUNT` is mapped to `S_PARTY`, `S_ORG_EXT`, and `S_ADDR_ORG`, as well as other tables. The default configuration is to process all base tables for each EIM table.

  > **Note:** Oracle Advanced Customer Services strongly recommends that you always include these parameters in every section of the IFB file, and list only those tables and columns that are relevant for a particular Siebel EIM task. Contact your Oracle sales representative to request assistance from Oracle Advanced Customer Services.

- **ONLY BASE COLUMNS** or **IGNORE BASE COLUMNS.** These parameters specify and restrict the selected base columns for the Siebel EIM process. The default is to process all base columns for each base table. It is likely that you are not using every column in a base table, and these parameters will make sure that Siebel EIM is only processing the desired columns in the table.

  You will see an additional performance increase if you exclude those columns that are defined as foreign keys (FKs) and are not used by the Siebel configuration, because Siebel EIM does not need to perform the interim processing (using SQL statements) to resolve the values for these FKs. Set the Siebel EIM task parameter `Error Flags` to a value of 1 to see which FKs are failing to be resolved by Siebel EIM (you might have missed excluding that FK with this parameter).

  > **Note:** Do not use the `IGNORE BASE COLUMNS` parameter for merge processes or export processes. Use this parameter only for import processes and delete processes.

Checking IFB File Optimization for Siebel EIM

This topic is part of *Recommended Sequence for Implementing Siebel EIM Processes*.

One method to find out whether the IFB file is optimized is to check the status of the records being processed in the EIM tables. This indicates if there are tables or columns that are being processed unnecessarily. The following query can be used to check the status of records in an EIM table:

```
select count(*), IF_ROW_STAT from EIM Table
where IF_ROW_BATCH_NUM = ?
group by IF_ROW_STAT;
```

If many rows have a status of PARTIALLY IMPORTED, then it is likely that further tuning can be done by excluding base tables and columns that are not necessary. For example, two tests were run to import 5000 accounts from `EIM_ACCOUNT` table. The first test included all of the base tables while the second test only focused on the four necessary tables by including the following line in the IFB file:

```
ONLY BASE TABLES = S_ORG_EXT, S_ADDR_ORG, S_ACCNT_POSTN, S_ORG_TYPE
```

The first test took 89 minutes to import (excluding the Updating Primaries step), while the second test took only 2 minutes to import (excluding the Updating Primaries step).

Separating Siebel EIM Processes by Operation

This topic is part of *Recommended Sequence for Implementing Siebel EIM Processes*.

Wherever possible, divide the Siebel EIM batches into insert-only transactions and update-only transactions. For example, assume that you are loading 50,000 records into an EIM table as part of a weekly process. 10,000 records represent new data and 40,000 records represent updates to existing data.

By default, Siebel EIM can determine which records are to be added and which records are to be updated in the base tables, however, Siebel EIM will need to perform additional processing (through SQL statements) to make these determinations. If you were able to divide the 50,000 records into different batch numbers based on the type of transaction, then you could avoid this additional processing.

In addition, the columns being processed as part of the update activity might be fewer than those for the insert activity (resulting in an additional performance increase). To illustrate this, the IFBs in the example can be coded with the following sections:

- IFB for mixed transactions:

```
[Weekly Accounts]
TYPE = IMPORT
BATCH = 1-10
TABLE = EIM_ACCOUNT
ONLY BASE TABLES = S_ORG_EXT
IGNORE BASE COLUMNS = S_ORG_EXT.?
```

- IFB for separate insert or update transactions:

```
[Weekly Accounts - New]
TYPE = IMPORT
BATCH = 1-2
TABLE = EIM_ACCOUNT
```

```
ONLY BASE TABLES = S_ORG_EXT
IGNORE BASE COLUMNS = S_ORG_EXT.?
INSERT ROWS = TRUE
UPDATE ROWS = FALSE
[Weekly Accounts - Existing]
TYPE = IMPORT
BATCH = 3-10
TABLE = EIM_ACCOUNT
ONLY BASE TABLES = S_ORG_EXT
ONLY BASE COLUMNS = S_ORG_EXT.NAME, S_ORG_EXT.LOC, S_ORG_EXT.?
INSERT ROWS = FALSE
UPDATE ROWS = TRUE
```

Troubleshooting Siebel EIM Performance

Before troubleshooting performance issues that are specific to Siebel EIM, verify that there are no performance bottlenecks on the Siebel Server computer or on the network. This topic contains the following information:

- *Optimizing SQL for Siebel EIM*
- *Using the USE INDEX HINTS and USE ESSENTIAL INDEX HINTS Parameters*
- *Using USE INDEX HINTS and USE ESSENTIAL INDEX HINTS: Example*
- *Using USE INDEX HINTS and USE ESSENTIAL INDEX HINTS: Criteria for Passing Indexes to the Database*
- *Using the SQLPROFILE Parameter*
- *Additional Indexes on Siebel EIM Tables*
- *Creating Proper Statistics on Siebel EIM Tables*
- *Dropping Indexes in Initial Runs of Siebel EIM*
- *Controlling the Size of Batches for Siebel EIM*
- *Controlling the Number of Records in Siebel EIM Tables*
- *Using the USING SYNONYMS Parameter with Siebel EIM*
- *Using the NUM_IFTABLE_LOAD_CUTOFF Extended Parameter with Siebel EIM*
- *Disabling Transaction Logging for Siebel EIM*
- *Disabling Database Triggers for Siebel EIM*
- *Running Siebel EIM Tasks in Parallel*

Optimizing SQL for Siebel EIM

This topic is part of *Troubleshooting Siebel EIM Performance*.

During this process, you need to be able to run several similar batches. If you do not have enough data with which to experiment, then you might need to back up and restore the database between runs, so that you can continue processing the same batch.

First, run a Siebel EIM job with the following flag settings: Error flag = 1, SQL flag = 8, and Trace flag = 3. This will produce a log file that contains SQL statements and shows how long each statement took. Identify SQL statements that are taking too long (on a run of 5000 rows in a batch, look for statements that took longer than one minute). These

are the statements that you want to concentrate on, and it is recommended that you consult an experienced database administrator at this point. The process of optimizing the SQL for Siebel EIM involves the following:

- Use the respective database vendor's utility or a third-party utility to analyze the long-running SQL statements.
- Based on the review of the data access paths, review the SQL statements for proper index usage. There might be cases where an index is not used at all or the most efficient index is not being chosen. This can require a thorough analysis.
- Based on this analysis, use a systematic approach to tuning these long-running statements. Perform one change at a time and measure the results of each change by comparing them to the initial benchmarks. For example, you might find that dropping a particular index to improve the performance of one long-running statement might negatively impact the performance of other SQL statements.

  Base the decision of whether to drop the index on the impact to the overall process, as opposed to the individual long-running SQL statement. For this reason, it is important that one change be implemented at a time in order to measure the impact of the change.
- After repetitively going through and optimizing each long-running SQL statement, the focus can be shifted to other tuning measures, such as increasing the number of records processed in the EIM table at a time and the running of parallel Siebel EIM tasks.

Using the USE INDEX HINTS and USE ESSENTIAL INDEX HINTS Parameters

This topic is part of *Troubleshooting Siebel EIM Performance*.

Perform testing with the IFB file parameters `USE INDEX HINTS` and `USE ESSENTIAL INDEX HINTS`, trying both settings (`TRUE` and `FALSE`). The default value for `USE INDEX HINTS` is `FALSE`. The default value for `USE ESSENTIAL INDEX HINTS` is `TRUE`.

> **Note:** If your configuration file has more than one process section, then you must specify `USE INDEX HINTS` within each one.

If these parameters are set to `FALSE`, then Siebel EIM does not generate hints during processing. By setting the value to `FALSE`, you can realize performance gains if the `TRUE` setting means that hints are being generated that direct the database optimizer to use less than optimal indexes. Siebel EIM processing must be tested with both the `TRUE` and `FALSE` settings to determine which one provides better performance for each of the respective Siebel EIM jobs.

> **Note:** The `USE INDEX HINTS` parameter is applicable only for Oracle Database. The `USE ESSENTIAL INDEX HINTS` parameter is applicable only for Oracle Database and Microsoft SQL Server.

These two parameters work for different queries, so you have to enable both to get all of the index hints on Oracle Database.

Further information is provided as follows:

- *Using USE INDEX HINTS and USE ESSENTIAL INDEX HINTS: Example*
- *Using USE INDEX HINTS and USE ESSENTIAL INDEX HINTS: Criteria for Passing Indexes to the Database*

Using USE INDEX HINTS and USE ESSENTIAL INDEX HINTS: Example

This topic is part of *Troubleshooting Siebel EIM Performance*.

See also:

- *Using the USE INDEX HINTS and USE ESSENTIAL INDEX HINTS Parameters*
- *Using USE INDEX HINTS and USE ESSENTIAL INDEX HINTS: Criteria for Passing Indexes to the Database*

This following table illustrates the results achieved for an SQL statement with index hints and without index hints. This example was performed on Microsoft SQL Server.

SQL User Name	CPU	Reads	Writes	Duration	Connection ID	SPID
SADMIN	549625	**38844200**	141321	**626235**	516980	9

```
UPDATE dbo.S_ASSET5_FN_IF
SET T_APPLDCVRG__RID =
(SELECT MIN(BT.ROW_ID)
FROM dbo.S_APPLD_CVRG BT (INDEX = S_APPLD_CVRG_U2)
WHERE (BT.COVERAGE_CD = IT.CVRG_COVERAGE_CD AND
BT.TYPE = IT.CVRG_TYPE AND
BT.ASSET_ID = IT.T_APPLDCVRG_ASSETI AND
(BT.ASSET_CON_ID = IT.T_APPLDCVRG_ASSETC OR
(BT.ASSET_CON_ID IS NULL AND IT.T_APPLDCVRG_ASSETC IS NULL)) AND
(BT.INSITEM_ID = IT.T_APPLDCVRG_INSITE OR
(BT.INSITEM_ID IS NULL AND IT.T_APPLDCVRG_INSITE IS NULL))))
FROM dbo.S_ASSET5_FN_IF IT
WHERE (CVRG_COVERAGE_CD IS NOT NULL AND
CVRG_TYPE IS NOT NULL AND
T_APPLDCVRG_ASSETI IS NOT NULL AND
IF_ROW_BATCH_NUM = 10710001 AND
IF_ROW_STAT_NUM = 0 AND
T_APPLDCVRG__STA = 0)
SET STATISTICS PROFILE ON
GO
SET STATISTICS IO ON
GO
select
(SELECT MIN(BT.ROW_ID)
FROM dbo.S_APPLD_CVRG BT (INDEX = S_APPLD_CVRG_U2)
WHERE (BT.COVERAGE_CD = IT.CVRG_COVERAGE_CD AND
BT.TYPE = IT.CVRG_TYPE AND
BT.ASSET_ID = IT.T_APPLDCVRG_ASSETI AND
(BT.ASSET_CON_ID = IT.T_APPLDCVRG_ASSETC OR
(BT.ASSET_CON_ID IS NULL AND IT.T_APPLDCVRG_ASSETC IS NULL)) AND
(BT.INSITEM_ID = IT.T_APPLDCVRG_INSITE OR
(BT.INSITEM_ID IS NULL AND IT.T_APPLDCVRG_INSITE IS NULL))))
FROM dbo.S_ASSET5_FN_IF IT
WHERE (CVRG_COVERAGE_CD IS NOT NULL AND
CVRG_TYPE IS NOT NULL AND
T_APPLDCVRG_ASSETI IS NOT NULL AND
IF_ROW_BATCH_NUM = 10710001 AND
IF_ROW_STAT_NUM = 0 AND
```

```
T_APPLDCVRG__STA = 0)
```

With Hints

```
Table 'S_APPLD_CVRG'. Scan count 1, logical reads 394774, physical reads 0,
read-ahead reads 280810.
Table 'S_ASSET5_FN_IF'. Scan count 1, logical reads 366, physical reads 0,
read-ahead reads 0.
```

Without Hints

```
Table 'S_APPLD_CVRG'. Scan count 1268, logical reads 10203, physical reads 697,
read-ahead reads 0.
Table 'S_ASSET5_FN_IF'. Scan count 1, logical reads 366, physical reads 0,
read-ahead reads 0.
```

Using USE INDEX HINTS and USE ESSENTIAL INDEX HINTS: Criteria for Passing Indexes to the Database

This topic is part of *Troubleshooting Siebel EIM Performance*. It explains how Siebel EIM determines which indexes to include on the hint clause passed to the database when using the `USE INDEX HINTS` and `USE ESSENTIAL INDEX HINTS` parameters.

See also:

- *Using the USE INDEX HINTS and USE ESSENTIAL INDEX HINTS Parameters*
- *Using USE INDEX HINTS and USE ESSENTIAL INDEX HINTS: Example*

When determining which indexes to pass on to the database as index hints, Siebel EIM takes the following steps:

1. Before generating a query, Siebel EIM makes a list of columns for which it has determined that an index is needed.
2. Siebel EIM then checks all of the indexes in the repository to find the index with the most matching columns.

Siebel EIM uses the following selection criteria in choosing indexes:

- Unique indexes have priority over nonunique indexes.
- Required columns have priority over nonrequired columns.

If a new index is created and it is declared in the repository, then there is a chance that Siebel EIM will choose it and pass it to the database on a hint.

Using the SQLPROFILE Parameter

This topic is part of *Troubleshooting Siebel EIM Performance*.

The inclusion of the `SQLPROFILE` parameter greatly simplifies the task of identifying the most time-intensive SQL statements. By inserting the following statement in the header section of the IFB file, the most time-intensive SQL statements will be placed in the file:

```
SQLPROFILE = c:\temp\eimsql.sql
```

Below is an example of the file eimsql.sql.

Start of the file: list of most time-intensive queries:

```
EIM: Integration Manager v8.2.2.3 [23021] LANG_INDEPENDENT SQL profile dump (pid
9096).
********************************************************************************
Top 100 SQL statements (of 24564) by total time:
Batch Step Pass Total Rows Per Row What
-------- -------- -------- -------- -------- -------- ------------------------
1000 4 106 124.07 0 virtual NULL key
1003 4 706 101.68 0 virtual NULL key
1002 4 506 93.15 0 virtual NULL key
1000 2 101 89.36 10000 0.01 default/fixed column
```

...list of queries continues

Statistics by step and by pass:

```
********************************************************************************
Statements per step by total time:
Step Stmts Total Min Max Avg %
-------- -------- -------- -------- -------- -------- --------
9 3405 14727.44 0.00 75.80 4.33 58.10
4 12011 2854.16 0.01 124.07 0.24 11.26
2 454 2165.35 0.50 89.36 4.77 8.54
```

...list of statistics continues...

SQL Statements

```
********************************************************************************
batch 1000, step 4, pass 106: "virtual NULL key":
(total time 2:04m (124s), no rows affected)
UPDATE SIEBEL.EIM_CONTACT
SET T_CONTACT_BU_ID = ?
WHERE (CON_BI IS NULL AND
T_CONTACT_BU_ID IS NULL AND
IF_ROW_BATCH_NUM = ? AND
IF_ROW_STAT_NUM = 0 AND
T_CONTACT__STA = 0)
```

...list of SQL statements continues...

Additional Indexes on Siebel EIM Tables

This topic is part of *Troubleshooting Siebel EIM Performance*.

An examination of the data access path will assist you in determining whether additional indexes are necessary to improve the performance of the long-running SQL. In particular, look for table scans and large index range scans. In the following example, after evaluating the inner loop of the nested select, it was recommended to add an index on all T2 columns.

Inner loop:

```
(SELECT MIN(ROW_ID)
FROM siebel.EIM_ACCOUNT T2
WHERE (T2.T_ADDR_ORG__EXS = 'Y' AND
T2.T_ADDR_ORG__RID = T1.T_ADDR_ORG__RID AND
T2.IF_ROW_BATCH_NUM = 105 AND
T2.IF_ROW_STAT_NUM = 0 AND
T2.T_ADDR_ORG__STA = 0))
```

The index was created to consist of T2 columns used in the WHERE clause with ROW_ID at the end of the index. This influenced the database optimizer to choose this index for index-only access. Since the query wants the minimum (ROW_ID), the very first qualifying page in the index will also contain the lowest value.

> **CAUTION:** All EIM indexes *must* start with IF_ROW_BATCH_NUM, or else serious performance and contention issues are unavoidable.

Adding Indexes to Improve Performance of S_ORG_EXT Table

The S_ORG_EXT table has indexes on many columns, but not all columns. If you have a large number of records (such as several million accounts) in S_ORG_EXT, then you might get a performance improvement in deleting and merging by adding an index to one or more of the following columns:

```
PR_BL_OU_ID
PR_PAY_OU_ID
PR_PRTNR_TYPE_ID
PR_SHIP_OU_ID
```

You can make subqueries to base tables that access only indexes. Performance is enhanced because all related records are physically collocated, and because index leaf pages contain many more records per page than wider base table pages. Before implementing any additional indexes, first discuss this with qualified support personnel.

Creating Proper Statistics on Siebel EIM Tables

This topic is part of *Troubleshooting Siebel EIM Performance*.

On IBM DB2, you can use the IFB file parameter UPDATE STATISTICS to control whether Siebel EIM dynamically updates the statistics of EIM tables. The default setting is TRUE. This parameter can be used to create a set of statistics on the EIM tables that you can save and then reapply to subsequent runs. After you have determined this optimal set of statistics, you can turn off the UPDATE STATISTICS parameter in the IFB file (UPDATE STATISTICS = FALSE), thereby saving time during the Siebel EIM runs.

> **Note:** It is recommended to manage statistics manually. Also note that using UPDATE STATISTICS in IFB executes RUNSTATS in SHRLEVEL REFERENCE mode, which causes exclusive locks and can prevent parallel EIM execution.

To determine the optimal set of statistics, you need to run several test batches and RUNSTATS commands with different options to see what produces the best results.

Before and after each test, execute the db2look utility in mimic mode to save the statistics from the database system catalogs. For example, if you are testing Siebel EIM runs using EIM_CONTACT1 in database SIEBELDB, then the following command generates UPDATE STATISTICS commands in the file EIM_CONTACT1_mim.sql:

```
db2look -m -a -d SIEBELDB -t EIM_CONTACT1 -o EIM_CONTACT1_mim.sql
```

The file EIM_CONTACT1_mim.sql contains SQL UPDATE statements to update database system catalog tables with the saved statistics. You can experiment with running test Siebel EIM batches after inserting the RUNSTATS commands provided in *IBM DB2 Options*. After you find the set of statistics that works best, you can apply that particular mim.sql file to the database. Between runs, save statistics with db2look.

Note: The `db2look` utility runs on IBM DB2 for Linux, Unix, and Windows (most of the IBM DB2 references in this guide are to this product). On IBM DB2 for z/OS, you can use the Optimization Service Center (OSC) utility instead. For more information about using Siebel EIM with IBM DB2 for z/OS, see *IBM DB2 for z/OS and Siebel EIM*.

IBM DB2 Options

The syntax for IBM DB2 commands provides more options, as follows:

- `shrlevel change`
- `allow write access`
- `allow read access`

The clauses `allow read access` and `shrlevel change` provide the greatest concurrency.

Dropping Indexes in Initial Runs of Siebel EIM

This topic is part of *Troubleshooting Siebel EIM Performance*.

Typically, the Siebel EIM initial load is a very database-intensive process. Each row that is inserted into the base tables requires modifications on the data page and the index pages of all of the affected indexes. However, most of these indexes are never used during a Siebel EIM run. Index maintenance is very time-consuming for most database managers and must be avoided as much as possible.

Therefore, the goal is to identify any indexes that are unnecessary for Siebel EIM and that can be dropped for the durations of the Siebel EIM run. You can create these indexes later in batch mode by using parallel execution strategies available for the respective database platform. Using this approach can save a significant amount of time.

Note: Under normal operations, using parallel execution strategies is *not* recommended.

- **Target Table Indexing Strategy.** For a target base table (such as `S_ORG_EXT`), you only need to use the Primary Index (Px, for example P1) and the Unique Indexes (Ux, for example U1), and then drop the remaining indexes for the duration of the Siebel EIM import. Past experience has determined that the Fx and Mx indexes can be dropped after an extensive SQL analysis of sample Siebel EIM runs.
- **Nontarget Table Indexing Strategy.** For child tables (such as `S_ADDR_ORG`), you only need to use the Primary Index (Px), the Unique Indexes (Ux), and the Foreign Key Indexes (needed for setting primary foreign keys in the parent table). Past experience has determined that the Fx and Mx indexes can be dropped after an extensive SQL analysis of sample Siebel EIM runs.

Note: Always perform testing when dropping indexes (or adding indexes) to make sure that the expected results are achieved.

Controlling the Size of Batches for Siebel EIM

This topic is part of *Troubleshooting Siebel EIM Performance*.

After tuning the long-running SQL statements, further tests can be run to determine the optimal batch size for each entity to be processed. The correct batch size varies and is influenced by the amount of buffer cache available. Optimal

batch ranges have been observed to range anywhere between 500 and 15,000 rows. Run several tests with different batch sizes to determine the size that provides the best rate of Siebel EIM transactions per second. Using the setting Trace Flag = 1 while running Siebel EIM helps in this task because you are then able to see how long each step takes and how many rows were processed by the Siebel EIM process.

Note: Also monitor this throughput rate when determining degradation in parallel runs of Siebel EIM.

Recommended Number of Rows for a Single Batch

For an initial load, you can use 30,000 rows for a large batch. For ongoing loads, you can use 20,000 rows for a large batch. Do not exceed 100,000 rows in a large batch.

Furthermore, for Microsoft SQL Server and Oracle Database environments, limit the number of records in the EIM tables to those that are being processed. For example, if you have determined that the optimal batch size for your implementation is 19,000 rows per batch and you are going to be running eight parallel Siebel EIM processes, then you must have 152,000 rows in the EIM table. Under no circumstances can you have more than 250,000 rows in any single EIM table because this reduces performance. The restrictions mentioned in the preceding example do not apply to IBM DB2 environments. As long as an index is being used to access the EIM tables, the numbers of rows in the EIM tables does not matter in DB2 environments.

CAUTION: For all supported RDBMS platforms, if indexes are added to EIM tables with any column other than `IF_ROW_BATCH_NUM` in the first position, then parallel Siebel EIM operations performed on that table will likely fail with index contention issues.

Note: The number of rows that you can load in a single batch can vary depending on your physical computer setup and on which table is being loaded. To reduce demands on resources and improve performance, generally try to vary batch sizes to determine the optimal size for each entity to be processed. In some cases, a smaller batch size can improve performance. But for simpler tables such as `S_ASSET`, you might find that loads perform better at higher batch sizes than for more complex tables such as `S_CONTACT`.

Controlling the Number of Records in Siebel EIM Tables

This topic is part of *Troubleshooting Siebel EIM Performance*.

Determine the number of records that can reside at one time in an EIM table while still maintaining an acceptable throughput rate during Siebel EIM processing. One observed effect of increasing the number of records in an EIM table is reduced performance of Siebel EIM jobs. This is often caused by object fragmentation or full table scans and large index range scans.

Note: In an IBM DB2 environment, EIM table size is not an important factor that impacts performance, because it is relatively easy to correct table scans and nonmatching index scans. So, a large number of records in an EIM table is not likely to reduce performance in a DB2 environment.

After addressing any object fragmentation and after the long-running SQL statements have been tuned, it is likely that you can increase the number of records that can reside in the EIM tables during Siebel EIM processing. When loading millions of records, this can result in a significant time savings because it reduces the number of times that the EIM table needs to be staged with a new data set.

When performing large data loads (millions of records) it is recommended that you perform initial load tests with fewer records in the EIM table. For example, while identifying and tuning the long-running SQL, you might start with approximately 50,000 records. After tuning efforts are complete, run additional tests, while gradually increasing the number of records. For example, you can incrementally increase the number of records to 100,000, then 200,000, and so on, until you have determined the optimal number of records to load.

Using the USING SYNONYMS Parameter with Siebel EIM

This topic is part of *Troubleshooting Siebel EIM Performance*.

The `USING SYNONYMS` parameter controls the queries of account synonyms during import processing. This parameter is also related to the `S_ORG_SYN` table. When set to `FALSE`, this parameter saves processing time because queries that look up synonyms are not used. The default setting is `TRUE`. Set this parameter to `FALSE` only when account synonyms are not needed.

Using the NUM_IFTABLE_LOAD_CUTOFF Extended Parameter with Siebel EIM

This topic is part of *Troubleshooting Siebel EIM Performance*.

Setting the `NUM_IFTABLE_LOAD_CUTOFF` extended parameter to a positive value will reduce the amount of time taken by Siebel EIM to load repository information. This is because when you set this parameter to a positive value, only information for the required EIM tables is loaded. For more information about this parameter, see *Siebel Enterprise Integration Manager Administration Guide* .

> **Note:** While this parameter is especially important for merge processes, it can also be used for any of the other types of processes.

Here is an example of using this parameter while running on Microsoft Windows from the server command line mode:

```
run task for comp eim server siebserver with config=account2.ifb,
ExtendedParams="NUM_IFTABLE_LOAD_CUTOFF=1", traceflags=1
```

Disabling Transaction Logging for Siebel EIM

This topic is part of *Troubleshooting Siebel EIM Performance*.

Typically, you set the system preference that disables transaction logging only for the initial data load. This setting (in the Administration - Siebel Remote screen, and Remote System Preferences view) is a check box labeled Enable Transaction Logging. This setting specifies whether or not the Siebel application logs transactions for the purpose of routing data to Siebel Mobile Web Clients.

By default, transaction logging is enabled. If there are no Siebel Mobile Web Clients, then you can disable this system preference. If you have Siebel Mobile Web Clients, then you must enable this system preference in order to route transactions to the Siebel Mobile Web Clients. However, during initial data loads, you can disable this system preference

to reduce transaction activity to the Siebel docking tables. After the initial loads are complete, enable transaction logging again.

> **Note:** For incremental data loads, transaction logging must be enabled if there are mobile clients. If this setting is changed for incremental data loads, then you will need to perform a reextract of all of the mobile clients.

Disabling Database Triggers for Siebel EIM

This topic is part of *Troubleshooting Siebel EIM Performance*.

Disabling database triggers, by removing them through the Administration - Server screens, can also help improve the throughput rate. This can be done by running the Generate Triggers server task with both the `REMOVE` and `EXEC` parameters set to `TRUE`. Be aware that components such as Workflow Policies and Assignment Manager will not function for the new or updated data. Also, remember to reapply the triggers after completing the Siebel EIM load.

Running Siebel EIM Tasks in Parallel

This topic is part of *Troubleshooting Siebel EIM Performance*.

Running Siebel EIM tasks in parallel is the last strategy that you might want to adopt in order to increase the Siebel EIM throughput rate. In other words, do not try this until all long-running SQL statements have been tuned, the optimal batch size has been determined, the optimal number of records to be processed at a time in the EIM table has been determined, and the database has been appropriately tuned. Before running tasks in parallel, check the value of the `Maximum Tasks` parameter. This parameter specifies the maximum number of running tasks that can be run at a time for a service. For more information about this parameter, see *Siebel System Administration Guide* .

> **Note:** `UPDATE STATISTICS` must be set to `FALSE` in the IFB file when running parallel Siebel EIM tasks on IBM DB2 for z/OS. Failure to do so can cause Siebel EIM tasks and executing `RUNSTATS` to take a longer time to complete. Also, when running parallel Siebel EIM tasks, deadlock and timeout will occur if `UPDATE STATISTICS` is set to `TRUE` in the IFB file.

Database Guidelines for Optimizing Siebel EIM

This topic describes Siebel EIM tuning tips for the database platforms supported by Siebel CRM. It contains the following information:

- *Microsoft SQL Server and Siebel EIM*
- *Oracle Database and Siebel EIM*
- *IBM DB2 and Siebel EIM*
- *IBM DB2 for z/OS and Siebel EIM*
- *IBM DB2 for z/OS Loading Process for Siebel EIM*
- *General Recommendations for the IBM DB2 for z/OS Loading Process*

Microsoft SQL Server and Siebel EIM

This topic is part of *Database Guidelines for Optimizing Siebel EIM*.

The information that follows describes Siebel EIM tuning tips for Microsoft SQL Server.

Fixing Table Fragmentation

Table and index fragmentation occurs on tables that have many insert, update, and delete activities. Because the table is being modified, pages begin to fill, causing page splits on clustered indexes. As pages split, the new pages might use disk space that is not contiguous, hurting performance because contiguous pages are a form of sequential input/output (I/O), which is faster than nonsequential I/O.

Before running Siebel EIM, it is important to defragment the tables by executing the `DBCC DBREINDEX` command on the table's clustered index. This applies especially to those indexes that will be used during Siebel EIM processing, which packs each data page with the fill factor amount of data (configured using the `FILLFACTOR` option) and reorders the information on contiguous data pages. You can also drop and recreate the index (without using the `SORTED_DATA` option). However, using the `DBCC DBREINDEX` command is recommended because it is faster than dropping and recreating the index, as shown in the following example:

```
DBCC SHOWCONTIG scanning '**S_GROUPIF' table...
Table: '**S_GROUPIF' (731969784); index ID: 1, database ID: 7
TABLE level scan performed.
Pages Scanned.................................: 739
Extents Scanned...............................: 93
Extent Switches...............................: 92
Avg. Pages per Extent.........................: 7.9
Scan Density [Best Count:Actual Count].......: 100.00% [93:93]
Logical Scan Fragmentation ...................: 0.00%
Extent Scan Fragmentation ....................: 1.08%
Avg. Bytes Free per Page......................: 74.8
Avg. Page Density (full)......................: 99.08%
DBCC execution completed. If DBCC printed error messages, contact the system
administrator.
```

To determine whether you need to rebuild the index because of excessive index page splits, look at the Scan Density value displayed by `DBCC SHOWCONTIG`. The Scan Density value must be at or near 100%. If it is significantly below 100%, then rebuild the index.

Purging an EIM Table

When purging data from the EIM table, use the `TRUNCATE TABLE` statement. This is a fast, nonlogged method of deleting all rows in a table. `DELETE` physically removes one row at a time and records each deleted row in the transaction log. `TRUNCATE TABLE` only logs the deallocation of whole data pages and immediately frees all of the space occupied by that table's data and indexes. The distribution pages for all indexes are also freed.

Parallel Data Load for EIM tables Using bcp

Microsoft SQL Server allows data to be bulk copied into a single EIM table from multiple clients in parallel, using the `bcp` utility or `BULK INSERT` statement. Use the `bcp` utility or `BULK INSERT` statement when the following conditions are true:

- SQL Server is running on a computer with more than one processor.
- The data to be bulk copied into the EIM table can be partitioned into separate data files.

These recommendations can improve the performance of data load operations. Perform the following tasks, in the order in which they are presented, to bulk copy data into SQL Server in parallel:

1. Set the database option truncate log on checkpoint to TRUE using sp_dboption.
2. Set the database option select into/bulkcopy to TRUE using sp_dboption.

   In a logged bulk copy all row insertions are logged, which can generate many log records in a large bulk copy operation. These log records can be used to both roll forward and roll back the logged bulk copy operation.

   In a nonlogged bulk copy, only the allocations of new pages to hold the bulk copied rows are logged. This significantly reduces the amount of logging that is needed and speeds the bulk copy operation. Once you do a nonlogged operation, immediately back up so that transaction logging can be restarted.
3. Make sure that the table does not have any indexes, or, if the table has an index, make sure that it is empty when the bulk copy starts.
4. Make sure that you are not replicating the target table.
5. Make sure that the TABLOCK hint is specified using bcp_control with eOption set to BCPHINTS.

   > **Note:** Using ordered data and the ORDER hint will not affect performance because the clustered index is not present in the EIM table during the data load.

6. After data has been bulk copied into a single EIM table from multiple clients, any clustered index on the table must be recreated using DBCC DBREINDEX.

TempDB

This is the database that Microsoft SQL Server uses for temporary space needed during execution of various queries. Set the initial size of the TEMPDB to a minimum of 100 MB, and configure it for auto-growth, which allows SQL Server to expand the temporary database as needed to accommodate user activity.

Configuration Parameters

Additional parameters have a direct impact on SQL Server performance and must be set according to the following guidelines:

- **SPIN COUNTER.** This parameter specifies the maximum number of attempts that Microsoft SQL Server will make to obtain a given resource. The default settings are adequate in most configurations.
- **MAX ASYNC I/O.** This parameter configures the number of asynchronous inputs/outputs (I/Os) that can be issued. The default is 32, which allows a maximum of 32 outstanding reads and 32 outstanding writes per file. Servers with nonspecialized disk subsystems do not benefit from increasing this value. Servers with high-performance disk subsystems, such as intelligent disk controllers with RAM caching and RAID disk sets, can gain some performance benefit by increasing this value because they have the ability to accept multiple asynchronous I/O requests.
- **MAX DEGREE OF PARALLELISM.** This option is used to configure Microsoft SQL Server's use of parallel query plan generation. Set this option to 1 to disable parallel query plan generation. This setting is mandatory to avoid generating an unpredictable query plan.
- **LOCKS.** This option is used to specify the number of locks that Microsoft SQL Server allocates for use throughout the server. Locks are used to manage access to database resources such as tables and rows. Set this option to 0 to allow Microsoft SQL Server to dynamically manage lock allocation based on system requirements.
- **AUTO CREATE STATISTICS.** This option allows SQL Server to create new statistics for database columns as needed to improve query optimization. Make sure that this option is enabled.
- **AUTO UPDATE STATISTICS.** This allows Microsoft SQL Server to automatically manage database statistics and update them as necessary to achieve proper query optimization. Make sure that this option is enabled.

Oracle Database and Siebel EIM

This topic is part of *Database Guidelines for Optimizing Siebel EIM*.

The subtopics that follow provide Siebel EIM tuning tips for Oracle Database.

Avoiding Excessive Table Fragmentation

Before running Siebel EIM, consult an experienced DBA in order to evaluate the amount of space necessary to store the data to be inserted in the EIM tables and the Siebel base tables. Also, for example with Oracle Database, you can make sure that the extent sizes of those tables and indexes are defined accordingly.

Avoiding excessive extensions and keeping a small number of extents for tables and indexes is important because extent allocation and deallocation activities (such as truncate or drop commands) can be demanding on CPU resources.

To check whether segment extension is occurring in Oracle Database

- Use the SQL statement that follows to identify objects with greater than 10 extents.

  Note: Ten extents is not a target number for segment extensions.

  ```
  SELECT segment_name,segment_type,tablespace_name,extents
  FROM dba_segments
  WHERE owner = (Siebel table_owner)
  and extents > 10;
  ```

  To reduce fragmentation, the objects can be rebuilt with appropriate storage parameters. Always be careful when rebuilding objects because of issues such as defaults or triggers on the objects.

Purging a Siebel EIM Table

When purging data from an EIM table, use the TRUNCATE command as opposed to the DELETE command. The TRUNCATE command releases the data blocks and resets the high water mark while the DELETE command does not, which causes additional blocks to be read during processing. Also, be sure to drop and recreate the indexes on the EIM table to release the empty blocks.

Disabling Archive Logging

It is recommended that Archive Logging be disabled during initial data loads. You can enable this feature to provide for point-in-time recovery after completing the data loads.

FREELIST Parameter

Multiple Siebel EIM processes can be executed against an EIM table provided that they all use different batches or batch ranges. The concern is that you might experience contention for locks on common objects. To run multiple jobs in parallel against the same EIM table, make sure that the FREELIST parameter is set appropriately for the tables and indexes used in the Siebel EIM processing.

Note: If you are using Auto Segment Space Mgmt (ASSM) as part of defining tablespaces, then the PCTUSED and FREELIST parameters (and FREELIST groups) are ignored.

Applicable database objects include EIM tables and indexes, as well as base tables and indexes. The value of the FREELIST parameter specifies the number of block IDs that will be stored in memory which are available for record insertion. Generally, you set the value to at least half of the intended number of parallel jobs to be run against the same EIM table (for example, a FREELIST setting of 10 permits up to 20 parallel jobs against the same EIM table).

This parameter is set at the time of object creation and the default for this parameter is 1. To check the value of this parameter for a particular object, the following query can be used:

```
SELECT SEGMENT_NAME, SEGMENT_TYPE, FREELISTS
FROM DBA_SEGMENTS
WHERE SEGMENT_NAME='OBJECT NAME TO BE CHECKED';
```

To change this parameter, the object must be rebuilt. Again, be careful when rebuilding objects because of issues such as defaults or triggers on the objects.

To rebuild an object

1. Export the data from the table with the grants.
2. Drop the table.
3. Recreate the table with the desired FREELIST parameter.
4. Import the data back into the table.
5. Rebuild the indexes with the desired FREELIST parameter.

Caching Tables

Another method to improve performance is to put small tables that are frequently accessed in cache. The value of BUFFER_POOL_KEEP determines the portion of the buffer cache that will not be flushed by the LRU algorithm. This allows you to put certain tables in memory, which improves performance when accessing those tables. This also makes sure that, after accessing a table for the first time, it will always be kept in the memory. Otherwise, it is possible that the table will get pushed out of memory and will require disk access the next time it is used.

Be aware that the amount of memory allocated to the keep area is subtracted from the overall buffer cache memory (defined by DB_BLOCK_BUFFERS). A good candidate for this type of operation is the S_LST_OF_VAL table. The syntax for keeping a table in the cache is as follows:

```
ALTER TABLE S_LST_OF_VAL CACHE;
```

Updating Tables

When there are 255 or more NVL functions in an update statement, Oracle Database updates the wrong data due to hash keys overflow. This issue is specific to Oracle Database. To avoid this problem, use fewer than 255 NVL functions in the update statement.

IBM DB2 and Siebel EIM

This topic is part of *Database Guidelines for Optimizing Siebel EIM*. It describes Siebel EIM tuning tips for an IBM DB2 database.

Review the following list of tuning tips for Siebel EIM:

- Use the IBM DB2 load replace option when loading EIM tables.

  Note: You can also use the IBM DB2 load option to purge EIM tables. To do this, run the load option with an empty (null) input file in LOAD REPLACE mode. This purges the specified EIM table(s) instantly.

- Use separate tablespaces for EIM tables and the base tables.
- For large Siebel EIM loads or where many Siebel EIM tasks execute in parallel, place individual EIM tables in separate tablespaces.
- Use large page sizes for EIM tables and for the larger base tables. Previous experience has determined that a page size of 16 KB or 32 KB provides good performance. The larger page sizes allow more data to be fitted on a single page and also reduces the number of levels in the index B-tree structures.
- Similarly, use large extent sizes for both EIM tables and the large base tables.
- Make sure that the tablespace containers are equitably distributed across the logical and physical disks and across the input/output (I/O) controllers of the database server.
- Use separate bufferpools for EIM tables and the target base tables. Since initial Siebel EIM loads are quite large and there are usually no online users, it is recommended to allocate a significant amount of memory to the EIM table and base table bufferpools.
- After you load new data, reorganize the tables if the data on a disk is out of cluster. If the results of executing the RUNSTATS command indicate that clustering has deteriorated (clustering index is less than 80% clustered) and that a reorganization of tables is required, then check the system catalog to see if tables need to be reorganized. See also 477378.1 (Article ID) on My Oracle Support. This article contains sample SQL that you can use to determine which tables are out of cluster and need reorganization.

  > **Note:** Allocate time to conversion schedules to allow for the reorganization of tables and the gathering of statistics prior to allowing end users access to a system containing new data.

- Use IBM DB2 snapshot monitors to make sure that performance is optimal and to detect and resolve any performance bottlenecks.
- You can turn off logretain during the initial load. However, you must turn it back on before moving into a production environment.

  > **Note:** When logretain is enabled, you must make a full cold backup of the database.

- For the EIM tables and the base tables involved, alter the tables to set them to `VOLATILE`. This makes sure that indexes are preferred over table scans.
- Executing Siebel EIM processes in parallel will cause deadlock and timeout on IBM DB2 databases if multiple Siebel EIM processes attempt to update the same catalog table simultaneously. To avoid this, set the `UPDATE STATISTICS` parameter to `FALSE` in the Siebel EIM configuration file (IFB file).
- Executing `UPDATE STATISTICS` in each Siebel EIM process consumes significant database server resources. It is recommended that the database administrator updates statistics outside of the Siebel EIM process using the `RUNSTATS` command.

Consider the settings for IBM DB2 registry values:

Registry Value	Setting
DB2_CORRELATED_PREDICATES =	YES
DB2_HASH_JOIN =	NO
DB2_PARALLEL_IO =	"*"
DB2_STRIPPED_CONTAINERS =	When using RAID devices for tablespace containers

Registry Value	Setting

Consider the settings for the IBM DB2 database manager configuration parameters:

Registry Value	Setting
INTRA_PARALLEL =	NO (can be used during large index creation)
MAX_QUERYDEGREE =	1 (can be increased during large index creation)
SHEAPTHRES =	100,000 (depends upon available memory, SORTHEAP setting, and other factors)

Consider the settings for the IBM DB2 database parameters:

Registry Value	Setting
CATALOGCACHE_SZ =	6400
DFT_QUERYOPT =	3
LOCKLIST =	5000
LOCKTIMEOUT =	120 (between 30 and 120)
LOGBUFSZ =	512
LOGFILESZ =	8000 or higher
LOGPRIMARY =	20 or higher
MAXLOCKS =	30
MINCOMMIT =	1
NUM_IOCLEANERS =	Number of CPUs in the database server
NUM_IOSERVERS =	Number of disks containing DB2 containers
SORTHEAP =	10240 (This setting is only for initial Siebel EIM loads. During production, set it to between 64 and 256.) The value you specify for SORTHEAP impacts the result of changing the value for SHEAPTHRES. For example, if SORTHEAP = 10000, then you can execute no more than nine Siebel EIM batches if you set SHEAPTHRES = 100000.

Registry Value	Setting
	If executing concurrent Siebel EIM batches, then make sure to allocate sufficient physical memory so that memory swapping or memory paging do not occur.
STAT_HEAP_SZ =	8000

IBM DB2 for z/OS and Siebel EIM

This topic is part of *Database Guidelines for Optimizing Siebel EIM*.

For IBM DB2 for z/OS configuration settings, you can find a listing (from the JCL) of the Database Manager Configuration Parameters (DSNZPARM) in *Implementing Siebel Business Applications on DB2 for z/OS* .

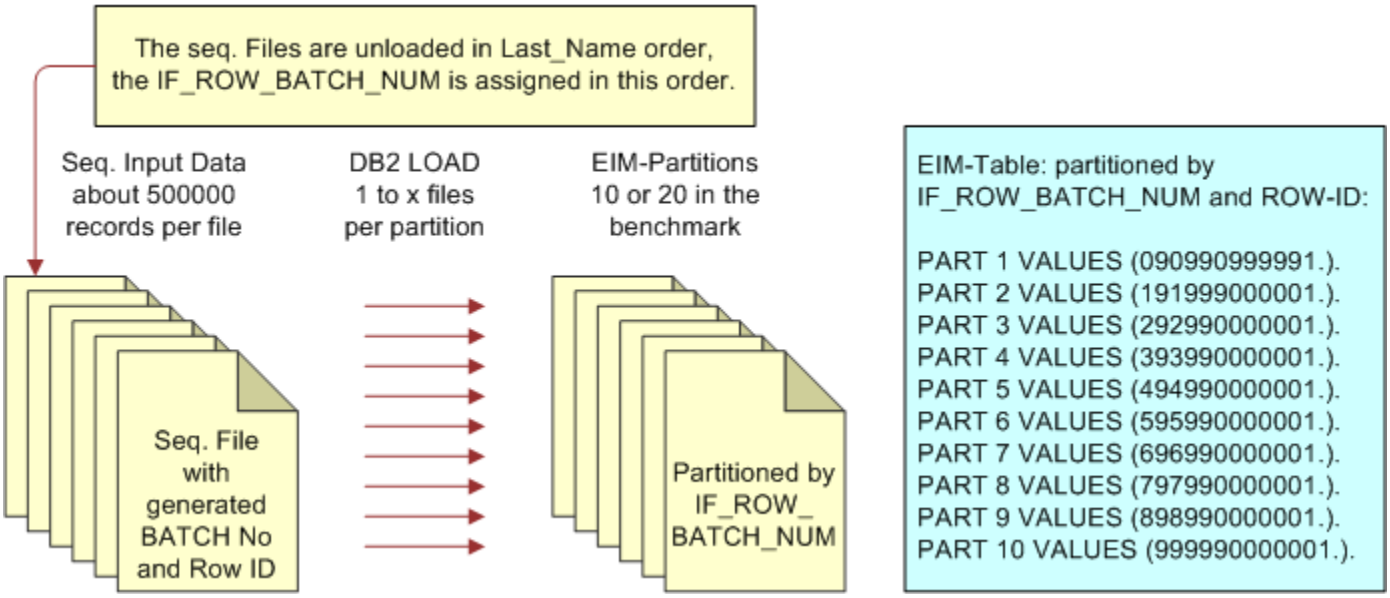
More IBM DB2 for z/OS information is provided in the following topics:

- *IBM DB2 for z/OS Loading Process for Siebel EIM*
- *General Recommendations for the IBM DB2 for z/OS Loading Process*

IBM DB2 for z/OS Loading Process for Siebel EIM

This topic is part of *Database Guidelines for Optimizing Siebel EIM*.

The following diagram illustrates the load process for IBM DB2 for z/OS. For more information, see *Siebel Enterprise Integration Manager Administration Guide* .



General Recommendations for the IBM DB2 for z/OS Loading Process

This topic is part of *Database Guidelines for Optimizing Siebel EIM*.

The following general recommendations apply when performing the IBM DB2 for z/OS loading process for Siebel EIM:

- Use the ONLY BASE TABLES and IGNORE BASE TABLES parameters or ONLY BASE COLUMNS and IGNORE BASE COLUMNS parameters in the IFB files to reduce the amount of processing performed by Siebel EIM. By using the IGNORE BASE COLUMNS option, you allow foreign keys to be excluded, which reduces both processing requirements and error log entries for keys which cannot be resolved. Remember that the key words ONLY and IGNORE are mutually exclusive. For example, the following settings exclude the options IGNORE BASE TABLES and ONLY BASE COLUMNS:

  ```
  ONLY BASE TABLES = S_CONTACT
  IGNORE BASE COLUMNS = S_CONTACT.PR_MKT_SEG_ID
  ```

- Import parents and children separately. Wherever possible, load data such as accounts, addresses, and teams at the same time, using the same EIM table.
- Use batch sizes that allow all of the EIM table data in the batch to be stored in the database cache (approximately 5,000 records for IBM DB2 for z/OS). Siebel EIM can be configured through the use of an extended parameter to use a range of batches. Remember to put the variable name into the IFB file.
- Multiple Siebel EIM processes can be executed against an EIM table, provided they all use different batches or batch ranges. However, the main limit to Siebel EIM performance is not the application server but the database. Contention for locks on common objects can occur if multiple Siebel EIM streams are executed simultaneously for the same base table. Multiple Siebel EIM job streams can run concurrently for different base tables, for example, S_ORG_EXT and S_ASSET.
- Run Siebel EIM during periods of minimum user activity, outside of business hours, if possible. This reduces the load for connected users and makes sure that the maximum processing capacity is available for the Siebel EIM processes.
- Set the system preference that disables transaction logging during the initial database load. This setting (in the Administration - Siebel Remote screen, and Remote System Preferences view) is a check box labeled Enable Transaction Logging. Unchecking this preference reduces transaction activity to the Siebel docking tables, which are used for synchronizing mobile clients.
- Disable the database triggers by removing them through the Server Administration screens. Doing this can also help to improve the throughput rate. Remember to reapply the triggers after the Siebel EIM load has completed, because the lack of triggers will mean that components, such as Workflow Policies and Assignment Manager, will not function for the new or updated data.
- Make sure that the required columns ROW_ID, IF_ROW_STAT, and IF_ROW_BATCH_NUM are correctly populated in the EIM table to be processed. The most efficient time to do this is when populating the EIM table from the data source or staging area, after cleansing the data.
- Unless there are specific processing requirements, make sure that the EIM table is empty before loading data into it for Siebel EIM processing. Always make sure that suitable batch numbers are being used to avoid conflicts within the EIM table. If you are using an automated routine, then truncating the EIM table between loads from the data source helps to preserve performance.
- When running Siebel CRM on an IBM DB2 for z/OS database, Siebel EIM can sometimes stop responding when updating the S_LST_OF_VAL base table. This is due to a data issue. The BU_ID column in the S_LST_OF_VAL base

table might have only one or very few distinct values. That makes the DB2 optimizer perform a table scan through all rows in the S_LST_OF_VAL table when most or all rows have the same BU_ID column value.

- To avoid this problem and speed up the query, modify the statistics data by running the following SQL statements:

```
update sysibm.sysindexes set firstkeycard=1000 where name='S_LST_OF_VAL_M2';
update sysibm.syscolumns set colcard = 1000 where tbname='S_LST_OF_VAL' and
name='BU_ID';
```

> **Note:** Depending on the data with which you are working, you might need to run other SQL statements beforehand.

Data Management Guidelines for Optimizing Siebel EIM

The following recommendations apply when performing the Siebel EIM loading process:

- The Siebel EIM mapping chart shows that many of the EIM table columns derive their values not from legacy database fields but from unvarying literal strings. Avoid filling up the EIM tables with this type of information, because it slows down the movement of real legacy data from the EIM tables to the base tables.
- Siebel EIM offers an alternative method for populating base table columns with unvarying literal strings, namely by using the DEFAULT COLUMN statement. This approach allows you to specify default literals that must be imported into the base tables without having to retrieve them from the EIM tables.

  For example, the Siebel EIM mapping chart shows Default Organization as the constant value for CON_BU in EIM_CONTACT, which in turn will move into BU_ID in S_CONTACT.

  The same result can be achieved with the setting DEFAULT COLUMN = CON_BU, Default Value in the IFB file. There are many other opportunities for moving literal strings from the EIM tables to the IFB file.

Run Parameter Guidelines for Optimizing Siebel EIM

The following recommendations are for setting run parameters when performing the Siebel EIM loading process:

- Set UPDATE STATISTICS to FALSE and manually manage the statistics. The provided version of RUNSTATS that UPDATE STATISTICS calls uses SHRLEVEL REFERENCE, which will cause locking contention and other issues.
- Do not set TRIM SPACES to FALSE. Using the TRIM SPACES parameter causes trailing spaces to be stored in the Siebel base table. This can lead to inefficient use of disk space since Siebel CRM uses VarChar on virtually all text columns longer than a single character. Setting TRIM SPACES to FALSE can also waste valuable bufferpool space for the tablespace data.
- Use either the IGNORE BASE TABLES parameter or the ONLY BASE TABLES parameter to limit the number of tables being inserted into or updated. The ONLY BASE TABLES parameter is preferable because the list is usually shorter and it is self-documenting. Using these parameters improves performance because it limits the number of tables that Siebel EIM attempts to load and they also save space for tables that will not be used by the user interface.
- Use either the IGNORE BASE COLUMNS parameter or the ONLY BASE COLUMNS parameter to limit the number of columns being inserted into or updated. The ONLY BASE COLUMNS parameter is preferable because the list is

usually shorter and it is self-documenting. Using these parameters improves performance because they limit the number of foreign keys that Siebel EIM attempts to resolve.

- Set the `USING SYNONYMS` parameter to `FALSE` in the IFB file. This logical operator indicates to Siebel EIM that account synonyms do not require processing during import, which reduces the amount of processing. Do not set the `USING SYNONYMS` parameter to `FALSE` if you plan to use multiple addresses for accounts. Otherwise, Siebel EIM will not attach addresses to the appropriate accounts.
- Suppress inserts when the base table is already fully loaded and the table is the primary table for an EIM table used to load and update other tables. The command format is `INSERT ROWS = table_name, FALSE`.
- Suppress updates when the base table is already fully loaded and does not require updates such as foreign key additions, but the table is the primary table for an EIM table used to load and update other tables. The command format is `UPDATE ROWS = table_name, FALSE`.

Monitoring the Siebel Server During a Siebel EIM Task

When you are monitoring the Siebel Server, the assumption is that you have allocated sufficient processor and memory resources for running the Siebel EIM task on the Siebel Servers and Siebel database servers.

If you are using Microsoft Windows Server as the operating system for the Siebel Server, then you can use the Microsoft Windows Performance Monitor to verify the amount of processor and memory resources being used by the hardware.

If you are using a supported UNIX or Linux operating system for the Siebel Server, then you can use `vmstat` and `iostat` to verify the amount of processor and memory resources being used by the hardware.

11 Tuning Siebel Remote for Performance

Tuning Siebel Remote for Performance

This chapter discusses tuning for Siebel Remote that can enhance performance. It contains the following topics:

- *About Siebel Remote*
- *Tuning Siebel Remote Server Components*
- *Tuning the Siebel Mobile Web Client in a Siebel Remote Deployment*

For more information about Siebel Remote, see *Siebel Remote and Replication Manager Administration Guide* on the *Siebel Bookshelf* . My Oracle Support also contains support documents that address performance issues for Siebel Remote.

About Siebel Remote

Siebel Remote allows Mobile Web Clients (typically operating remotely, in disconnected mode on a laptop) to connect to a Siebel Server and exchange updated data and files, a process known as synchronization. Siebel Remote supports mobile computing by allowing field personnel to share current information with members of virtual teams of other mobile and connected users across the organization.

Siebel Remote uses server components in two component groups to manage the exchange of data and files. You must enable both of these component groups in order to use Siebel Remote. Additional component groups might also be required for your deployment.

The Siebel Remote component group (alias Remote) includes the following components:

- Generate New Database (alias GenNewDb)
- Replication Agent (alias RepAgent). This component is used by Siebel Replication Manager, and has no role in supporting Mobile Web Clients
- Synchronization Manager (alias SynchMgr)
- Transaction Merger (alias TxnMerge)

The Disconnected Mobile Synchronization component group (alias MobileSync) includes the following components that are used by Siebel Remote or Siebel Replication Manager:

- Database Extract (alias DbXtract)
- Parallel Database Extract (alias PDbXtract)
- Transaction Processor (alias TxnProc)
- Transaction Router (alias TxnRoute)

Note: The MobileSync component group also includes some components that are used only by Siebel Mobile disconnected applications.

For information about how you can configure some of these components to optimize the performance of your Siebel Remote deployment, see *Tuning Siebel Remote Server Components*.

Related Books

Siebel Remote and Replication Manager Administration Guide

Siebel Installation Guide

Siebel System Administration Guide

Siebel Mobile Guide: Disconnected

Tuning Siebel Remote Server Components

This topic describes how you can improve the performance of certain components on the Siebel Remote Server. It contains the following information:

- *Increasing Throughput for the Database Extract and Parallel Database Extract Components*
- *Tuning the Transaction Router Component*

Increasing Throughput for the Database Extract and Parallel Database Extract Components

This topic is part of *Tuning Siebel Remote Server Components*. It provides tips to help you improve the throughput for the Database Extract (alias DbXtract) and Parallel Database Extract (alias PDbXtract) components.

For more information about component groups that you must enable to use Siebel Remote, see *About Siebel Remote*.

Use Parallel Database Extract for extracting a regional database or extracting remote users who have very large visibility footprints. Some of the characteristics of Database Extract and Parallel Database Extract follow:

- Parallel Database Extract can support only one task at a time. You cannot run multiple instances of Parallel Database Extract simultaneously, as you can do with Database Extract.
- Parallel Database Extract takes advantage of servers that have multiple CPUs and RAID disc array by using multiple threads.
- Database Extract by default uses a data file type of Bulk Copy format (BCP) to export data. This can lead to a faster time in extracting and initializing than if you use the data file type DAT.

The following list describes tips to improve the throughput for the Database Extract and Parallel Database Extract components:

- Run multiple concurrent instances of Database Extract

  You can increase throughput by running multiple concurrent instances of the Database Extract component. Each instance of the Database Extract component requires a temporary table. This table is called `S_DOCK_INITM_N` where N equals the value of the parameter `TS Table Number` (alias `TSTableNum`). `TS Table Number` specifies the number of the temporary table that serves the Database Extract component.

  For example, if `TS Table Number` equals 1, then temporary table 1 (`S_DOCK_INITM_1`) serves the instance of the Database Extract component that is currently executing.

By default, 48 temporary tables are available for use. If you require additional tables, then you create them using Siebel Tools.

The recommended number of temporary tables to use depends on the database platform in use. For example:

- Oracle Database

  The number of temporary tables that you use depends on the size of the shared pool that this database server can access. If the size of the shared pool is less than 300 MB, then it is recommended that you use one temporary table and execute one instance of the Database Extract component. If the size of the shared pool is greater than 600 MB, then using one temporary table for each instance of the Database Extract component can increase throughput.
- Microsoft SQL Server and IBM DB2

  Use one temporary table for each instance of the Database Extract component that you execute. For example, if you execute eleven instances of the Database Extract component, then use eleven temporary tables.

- Stop Transaction Router component

  Stop the Transaction Router component if you are running multiple instances of the Database Extract or Parallel Database Extract component in order to avoid table locking.

- Extract lists of users simultaneously

  Separate users to be extracted into groups of 50 to 100 users per list and extract these lists of users simultaneously using the Database Extract and Parallel Database Extract components. These components extract the Enterprise visibility data for all users in the list once.

- Set the `Truncate TS table` parameter to `TRUE`

  Setting this parameter (alias `TruncateTSTable`) to `TRUE` can improve performance, because deletions from `S_DOCK_INITM_N` tables are usually logged and can take a long time.

- Defragment tables

  Defragment tables to which you intend to extract data and defragment `S_DOCK_INITM_N` tables.

For more information about the components and parameters described here, see *Siebel Remote and Replication Manager Administration Guide* . For more information about performance issues for the Database Extract component, see 477174.1 (Article ID) on My Oracle Support. This document was previously published as Siebel Troubleshooting Steps 15.

Tuning the Transaction Router Component

This topic is part of *Tuning Siebel Remote Server Components*. It describes how to resolve or avoid performance issues for the Transaction Router component (alias TxnRoute).

For more information about component groups that you must enable to use Siebel Remote, see *About Siebel Remote*.

Performance issues for Transaction Router can arise from the following sources:

- Visibility-related transactions
- Docking rules and data distribution

- Slow-running queries
- Increasing Transaction Router throughput
- Setting the `Id Db Size` parameter

For more information about Transaction Router performance issues, consult the following documents:

- 476759.1 (Article ID) on My Oracle Support. This document, previously published as Siebel Troubleshooting Steps 8, describes how to diagnose and resolve Transaction Router performance issues.
- 477816.1 (Article ID) on My Oracle Support. This document, previously published as Siebel Troubleshooting Steps 38, describes how to monitor and manage the transaction backlog for a Siebel Remote implementation.

Visibility-Related Transactions

If you diagnose the root cause of the Transaction Router performance issue to be visibility-related transactions, then consider the following two possible solutions:

- Reextract all mobile users and regional nodes. For more information, see *Siebel Remote and Replication Manager Administration Guide* .
- Allow the Transaction Router component tasks to continue processing until they clear the backlog.
- Once the Transaction Router has processed all visibility-related transactions, the backlog will be processed more quickly. Starting additional Transaction Router tasks can also improve performance, but do not start more tasks than the Siebel Server or database engine can support.

Docking Rules and Data Distribution

If you diagnose the root cause of the Transaction Router performance issue to be docking rule related transactions, then it is advisable to request assistance from Global Customer Support. Create a service request (SR) on My Oracle Support. Alternatively, you can phone Global Customer Support directly to create a service request or get a status update on your current SR. Support phone numbers are listed on My Oracle Support.

When you log the SR, provide the following pieces of information:

- RDBMS trace of the Transaction Router task
- Transaction Router log files
- .dx files that the Transaction Router is processing from the `SIEBEL_ROOT\siebsrvr\Docking\txnproc` directory
- The results from executing the visrule script

  For more information about the visrule script, see 476759.1 (Article ID) on My Oracle Support. This document was previously published as Siebel Troubleshooting Steps 8.

Slow-Running Queries

If you diagnose the root cause of the Transaction Router performance issue to be slow-running queries, then consult your database administrator to determine the following:

- All indexes are present and valid on the tables involved in the poor-performing queries. To determine if all indexes are valid and present, see *Siebel Data Model Reference* on My Oracle Support (Article ID 2285310.1).
- Check whether the tables and indexes involved in the poor-performing queries require defragmentation.

Increasing Transaction Router Throughput

The following factors can impact throughput for the Transaction Router component:

- Large batch sizes for Siebel Enterprise Integration Manager (Siebel EIM)

It is recommended that, where possible, you reduce the size of the batch that Siebel EIM processes when it imports data. In addition, it is recommended that you log transactions to the Siebel File System rather than the Master Transaction Log (`s_dock_txn_log`). For more information, see *Siebel Enterprise Integration Manager Administration Guide* .

- Large batch size for Siebel Assignment Manager

  Where possible, reduce the batch file size for Siebel Assignment Manager, because large batch files can impact the performance of the Transaction Router component. For information, see *Siebel Assignment Manager Administration Guide* .

In both instances, you need to decide if an increase in throughput for the Transaction Router component is more important than a decrease in throughput for the Siebel EIM and Siebel Assignment Manager components, before you make changes.

Setting the Id Db Size Parameter

Increasing the size of the visdata.dbf file has been shown to help performance in certain situations. This file is a cached file used by the Transaction Router and Transaction Processor components. The size of the visdata.dbf file is determined by the value of the parameter `Id Db Size`, which is available for both of these components.

The value for this parameter must be the same in both components. If the value of the `Id Db Size` parameter is increased in one component, such as to `204800` (200 MB), then it must be changed in the other component.

Tuning the Siebel Mobile Web Client in a Siebel Remote Deployment

This topic discusses how you can optimize the performance of a Siebel Mobile Web Client in a Siebel Remote deployment. It contains the following information:

- *Optimizing Application Configuration File Parameters*
- *Guidelines for Optimizing Data Synchronization Between Siebel Mobile Web Client and Siebel Remote*
- *Choosing an Appropriate Routing Model*

For more information about performance tuning for Siebel Mobile Web Clients, see *Tuning Siebel Web Client for Performance*.

Optimizing Application Configuration File Parameters

This topic is part of *Tuning the Siebel Mobile Web Client in a Siebel Remote Deployment*. It discusses how you can modify values for the parameters specified in your Siebel application configuration file to optimize the performance of a Siebel Mobile Web Client.

DockTxnsPerCommit

The value of this parameter specifies the number of transactions that Siebel Remote applies to the local database before performing a commit. If you have an environment where a large number of transactions are constantly being created, then adjusting the value of `DockTxnsPerCommit` upwards might decrease the amount of time taken for initialization and

synchronization tasks for large quantities of data. In such a scenario, test a variety of values (for example, 1000, 2000, 3000) and determine which value is best for your environment.

The `DockTxnsPerCommit` parameter appears in the `[Local]` section of your application configuration file. The default value is 500.

Guidelines for Optimizing Data Synchronization Between Siebel Mobile Web Client and Siebel Remote

This topic is part of *Tuning the Siebel Mobile Web Client in a Siebel Remote Deployment*. It lists some points that can help you to optimize data synchronization between your Siebel Mobile Web Client and Siebel Remote Server. Note the following:

- Synchronize frequently

  Synchronizing frequently reduces the number of transactions to transmit and commit for each synchronization session. The longer that a user waits between synchronization sessions, the more data there is to send.
- Enable TrickleSync on the Siebel Mobile Web Client

  Each time that a Siebel Mobile Web Client connects to your Siebel Enterprise network, TrickleSync performs database synchronization. For more information, see *Siebel Remote and Replication Manager Administration Guide* .
- Use time-based filters to prevent sending data from server to client that is older than a specific date.

Choosing an Appropriate Routing Model

This topic is part of *Tuning the Siebel Mobile Web Client in a Siebel Remote Deployment*.

One way to increase performance is to reduce the volume of data transmitted to the remote users. This can be best achieved by choosing the appropriate routing model.

Routing model choices provide a useful level of granularity in setting the size of a local database. In general, using appropriate routing models improves overall performance of a remote environment and helps decrease Database Extract times and increase Transaction Router throughput.

If none of the supplied routing models are appropriate, then contact Oracle Advanced Customer Services, who can help you to develop a routing model that is appropriate to your environment. Contact your Oracle sales representative to request assistance from Oracles Application Expert Services.

12 Tuning Customer Configurations for Performance

Tuning Customer Configurations for Performance

This chapter discusses how you can avoid common performance-related problems in Siebel CRM that stem from customer configuration done using Siebel Tools (or Siebel Web Tools) or Siebel scripting languages. It contains the following topics:

- *General Performance Guidelines for Customer Configurations*
- *Analyzing Generated SQL for Performance Issues*
- *Guidelines for Siebel Scripting*
- *Guidelines for Data Objects Layer*
- *Guidelines for Business Objects Layer*
- *Guidelines for User Interface Objects Layer*

Application development information is also available in the following books on the *Siebel Bookshelf* and in *Siebel Tools Online Help* :

- *Using Siebel Tools*
- *Configuring Siebel Business Applications*
- *Configuring Siebel Open UI*
- *Siebel Developer's Reference*
- *Siebel Object Types Reference*
- *Siebel Object Interfaces Reference*
- *Siebel eScript Language Reference*
- *Siebel VB Language Reference*

General Performance Guidelines for Customer Configurations

This topic provides some general guidelines for customer configuration using Siebel Tools.

Using your hardware resources optimally, and configuring your system appropriately, can help you to achieve your performance goals. Consider your resources and requirements carefully, and test and monitor system performance on a continual basis.

Siebel CRM architecture has been designed and tuned for optimal performance, making use of features such as database indexes, data caching, RDBMS cursors, efficient SQL generation, native database APIs, and so on. However, custom configurations can have various potential performance pitfalls, the impact of which might be amplified in

environments with large databases and wide data distribution across servers. Follow the guidelines presented here and in other documentation to avoid such problems.

The following are some miscellaneous configuration guidelines for maintaining optimal performance:

- **Avoid using sort specifications on nonindexed columns or joined columns.** For more information, see *Managing Database Indexes in Sorting and Searching* and other relevant topics.
- **Avoid overly complex user interface configuration.** In general, do not include a large number of applets per view (generally include no more than four applets), or a large number of fields per applet.
- **Use the performance of standard Siebel views as a benchmark.** Custom configurations are customized versions of the standard Siebel CRM configuration. The Siebel data model is designed to have good performance for the expected standard queries for the standard views. If a custom view is slow, then verify the performance of the most appropriate standard view for that custom view. For example, if your custom version of the All Accounts view is slow, then verify the performance of the standard All Accounts view. Use this as a benchmark for the custom view and something to compare the performance to.
- **Limit the number of business components in a view.** An excessive number of different business components used in applets in a view can slow down the display of data upon entry into that view. This is because each of the applets must be populated with data.
- **Limit the number of virtual business components in a view.** Avoid using more than two virtual business components in a single view.
- **Limit the number of fields in business components or applets.** There is no set limit on the number of fields in a business component, or the number of list columns in a list applet. However, a business component with too many active fields will have degraded performance. Also, in some database systems it is possible to generate a query that is too large to be processed. See also *Limiting the Number of Active Fields*.

  In particular, reduce the number of fields that are displayed in the master applet on related views. The information is static and might not be necessary. Additional space will be available on the view for supporting data without users needing to scroll. (This will also provide a usability benefit.)

  End users can reduce or increase the number of fields that are displayed in a list applet, by using the Columns Displayed menu option. However, it is best to provide an optimal default number of visible fields for each applet. It is also best to provide the minimum required total number of fields, including those that are hidden by default.
- **Limit the number of required fields.** Required fields are always retrieved from database queries. Consequently, limiting the number of required fields (that is, fields for which the Required user property is `TRUE`) in your business components can improve performance. See also *Limiting the Number of Active Fields*.
- **Limit the number of records returned.** To limit the number of records returned for a business component, you can add a search specification to the business component or to applicable applets or links, or you can define a default predefined query on the view.
- **Limit the number of joins, extension tables, and primary ID fields in a business component.** Joins degrade performance by causing an extra row retrieval operation in the joined table for each row retrieval in the main table. Extension tables and primary ID fields also use joins, although implied rather than explicitly defined, adding a row retrieval operation for each.

  The more joins, extension tables, and primary ID fields defined in a business component, the higher the number of row retrievals required in tables other than the main table, with a corresponding performance degradation.
- **Limit the use of Link Specification property in fields.** `TRUE` settings in the Link Specification property in fields can also slow performance. If `TRUE`, then the field's value is passed as a default value to a field in the detail business component through a link.

This is necessary if the master business component has a link relationship (in the current business object) with one or more detail business components, and these detail business components utilize the `Parent:` expression in the Pre Default Value, Post Default Value, or Calculated Value properties in any fields. The master business component must pass the field value to any detail records displayed.

As with the Force Active property, fields with the Link Specification property set to `TRUE` will be retrieved every time the business component is queried.

- **Use inner joins rather than outer joins.** Inner joins can be used for joined tables, with a resulting savings in overhead, provided that you are guaranteed that all foreign key references are valid.

  For example, when the join is from a detail business component to its master, you are guaranteed of the existence of the master. You can configure the join as an inner join by setting the Outer Join Flag property of the Join object definition to `FALSE`. This improves the performance of queries that use the join. In general, avoid using double outer joins.
- **Configure Cascade Delete appropriately for many-to-many links.** The Cascade Delete property in a Link object definition must be correctly configured for use in a many-to-many link, or the first insertion or deletion in the association applet will be abnormally slow. A link object definition used in a many-to-many relationship is one that contains a non-NULL value for the Inter Table property. The Cascade Delete property in such a link must be set to None.
- **Remove unneeded sort buttons.** Remove sort buttons from list columns in list applets where this capability is not required.
- **Reduce the need to scroll in a view.** Whenever possible, design views that do not require scrolling. (This will also provide a usability benefit.)
- **Provide tuned PDQs.** Provide tuned PDQs (predefined queries) that address most user requirements. Doing so reduces the likelihood of users creating undesirably complex queries. You might also provide guidance to end users on constructing appropriate queries.
- **Cache business services.** Cache business services that must be accessible at all times in a user session. To do this, set the Cache property to `TRUE` for each applicable Business Service object definition. Caching of business services has an impact on memory, because the services are cached per session. Make sure that only frequently accessed business services in a session are marked as cacheable.
- **Avoid calculated fields that do Counts and Sums.** Reduce, where possible, the use of calculated fields that do Counts and Sums. If such fields are active, they will cause performance degradation.

Analyzing Generated SQL for Performance Issues

Performance troubleshooting is an iterative process. You need to consider performance implications during design and development. Note any changes to potentially troublesome areas, such as MVGs, business component sort and search specifications, joins, extension tables, or indexes. You then test the application to determine bottlenecks, using realistic data volumes and distribution in your test environment. Focus your testing efforts on the slowest, most important, and most highly configured views.

If a performance problem is detected in testing or production, then your next step is to analyze the SQL statements being generated by Siebel CRM. This is one of the most useful diagnostic tools available to you for performance analysis.

This topic contains the following information:

- *About Specifying SQL Logging and SQL Tagging for Siebel Application Object Manager Components*

- *Troubleshooting Poor Performing SQL at the Database Level Using Workload Tagging*
- *Specifying SQL Spooling in Siebel Developer Web Client*
- *Troubleshooting Performance Using SQL Trace Files*
- *Troubleshooting Performance Using SQL Query Plans*
- *Example of Obtaining a Query Plan*
- *SQL Queries Against Database Data*

About Specifying SQL Logging and SQL Tagging for Siebel Application Object Manager Components

This topic is part of *Analyzing Generated SQL for Performance Issues*.

SQL logging and SQL tagging are diagnostic tools that log and tag SQL statements generated and executed by Siebel CRM. When used in conjunction with SQL logging, SQL tagging can help administrators trace long-running or slow-performing SQL queries back to the user or action that triggered them. You can also use these tools to review generated SQL in a development or test environment after making configuration changes in Siebel Tools.

Note: SQL tagging must be used in development or test environments only. It is strongly recommended not to use this diagnostic feature in the production environment, because it generates a large number of log entries, which could impact performance negatively and lead to running out of disk space on the Siebel Server.

- SQL logging controls the level of SQL logging detail in a log file for an object manager-based component. To set SQL logging for a component, set the `Object Manager SQL Log` (alias `ObjMgrSqlLog`) event to an appropriate value. For example, specify the event value for a Siebel Application Object Manager component such as Call Center Object Manager (ENU).
- SQL tagging controls whether or not `SELECT` statements are tagged for an object manager-based component. SQL tagging also controls whether or not to log additional information, such as the name of a business component. SQL tagging is primarily used during the development phase. To set SQL tagging for a component, set the `OM SQL Tagging` (alias `ObjMgrSqlTag`) event to an appropriate value. By default, SQL tagging is disabled.

CAUTION: SQL tagging changes the format and content of the SQL SELECT statement by adding a bind, which inhibits the use of Oracle stored outlines, if they exist.

With SQL tagging, data that can be logged includes the server component name, server name, task ID, user ID, flow ID (with Siebel ARM ID), business object name, business component name, and view name. Which elements are logged depends on the level set for the event.

For instructions on how to use SQL tagging and logging to log and tag generated SQL statements at the server component level, see *Siebel System Monitoring and Diagnostics Guide* .

Troubleshooting Poor Performing SQL at the Database Level Using Workload Tagging

This topic is part of *Analyzing Generated SQL for Performance Issues*.

Siebel workload tagging is a diagnostic tool that tags SQL statements generated and executed by Siebel CRM running on Oracle Database. This tool helps database administrators trace and troubleshoot poor performing SQL or excessive workload at the database level.

Note: Siebel workload tagging is available only for Siebel applications running on Oracle Database.

When workload tagging is enabled, specified Siebel Application Object Manager-generated SQL statements (**SELECT**, **INSERT**, **UPDATE**, **DELETE**) are tagged with workload tagging attributes. The values for these attributes include the following:

- CLIENT_IDENTIFIER: component_name,servername,taskID,userID,FlowID:SarmID
- ACTION: View Name
- MODULE: Business Component Name

Note: The length of the previous attribute values is limited to 64 bytes for CLIENT_IDENTIFIER, 32 bytes for ACTION, and 48 bytes for MODULE. Truncation occurs when the length exceeds these limitations.

When poor performing SQL statements are suspected, the database administrator can use this tagging information to trace details about the generated SQL and the user that initiated it. Database administrators can then use Oracle Enterprise Manager to find the component task and log detail for more in-depth analysis of the performance issue. Alternatively, you can use SQL*Plus or similar tools to query the Oracle Database V$SESSION view to look up the CLIENT_IDENTIFIER, ACTION, and MODULE attributes by using SQL_ID of the SQL statement.

This topic contains the following information:

- *About Enabling and Disabling Workload Tagging*
- *Requirements for Enabling and Disabling Workload Tagging*
- *Enabling and Disabling Workload Tagging Using the Siebel Application*
- *Enabling and Disabling Workload Tagging Using the Siebel Server Manager*
- *Using Workload Tagging to Troubleshoot Poor Performing SQL at the Database Level*

About Enabling and Disabling Workload Tagging

By default, workload tagging is disabled. You choose which SQL statements you want enabled for workload tagging by setting the log level of the OCI SQL Tagging event type for the applicable Siebel Application Object Manager server component. The log levels are listed in the following table.

Log Level	Description
0	Workload tagging is disabled.
1	Workload tagging is disabled. This is the default setting.
2	Workload tagging is enabled only for **SELECT** statements.
3	Workload tagging is enabled only for **INSERT**, **UPDATE**, and **DELETE** statements.
4	Workload tagging is enabled for **SELECT**, **INSERT**, **UPDATE**, and **DELETE** statements.

Note: The OCI SQL Tagging event type is available to all Object Manager-based components for supported languages, such as Call Center Object Manager (ENU), Sales Object Manager (DEU), eService Object Manager (FRA), and so on.

You can use either of the following methods to enable and disable workload tagging:

- *Enabling and Disabling Workload Tagging Using the Siebel Application*
- *Enabling and Disabling Workload Tagging Using the Siebel Server Manager*

Requirements for Enabling and Disabling Workload Tagging

Before enabling (or disabling) workload tagging, make sure that the following requirements are met:

- Siebel Server is up and running.
- Oracle Enterprise Manager is configured for the database server. For information about Oracle Enterprise Manager, see the documentation on Oracle Technology Network.

Enabling and Disabling Workload Tagging Using the Siebel Application

Use the following procedure to set the log level of the OCI SQL Tagging event type for the applicable Siebel Application Object Manager server component using the Siebel application GUI.

To enable and disable workload tagging using the Siebel application

1. Navigate to the Administration - Server Configuration screen, then the Components view.
2. Select the appropriate Siebel Application Object Manager for which you want to enable workload tagging. For example, if the application that you are using is Siebel Call Center for U.S. English, then select Call Center Object Manager (ENU).
3. Click the Events subview, and then select the OCI SQL Tagging event type.
4. Set the Log Level as described in *About Enabling and Disabling Workload Tagging*.

After workload tagging is enabled, database administrators can use Oracle Enterprise Manager to diagnose and troubleshoot the problematic generated SQL.

Enabling and Disabling Workload Tagging Using the Siebel Server Manager

Use the Siebel Server Manager (srvrmgr program) to set the OCI SQL Tagging event type for the applicable Siebel Application Object Manager server component by way of the command-line interface. For more information about Siebel Server Manager, see *Siebel System Administration Guide* .

To enable and disable workload tagging using the Siebel Server Manager

1. Make sure that the Siebel Application Object Manager server component for your Siebel application is running.
2. Start the srvrmgr program.
   For information about starting the srvrmgr program, see *Siebel System Administration Guide* .
3. To enable workload tagging, run the following command:
   - `change evtloglvl OCISqlTag = loglevel for comp appobjmgr_lang`
     where:
   - loglevel determines the types of SQL statements that are tagged (see *About Enabling and Disabling Workload Tagging*).
   - appobjmgr is the Siebel Application Object Manager server component for your application, and lang is the three-letter identifier for the language specific to your environment.

For example, the following command enables workload tagging only for `SELECT` statements for Siebel Call Center for U.S. English:

```
change evtloglvl OCISqlTag = 2 for comp sccobjmgr_enu
```

4. To disable workload tagging, run the following command:

```
change evtloglvl OCISqlTag = 0 for comp appobjmgr_lang
```

In this command, appobjmgr is the Siebel Application Object Manager server component for your application, and lang is the three-letter identifier for the language specific to your environment.

After workload tagging is enabled, database administrators can then use Oracle Enterprise Manager to diagnose and troubleshoot problematic generated SQL. For information about how to use Oracle Enterprise Manager for workload tagging, see *Using Workload Tagging to Troubleshoot Poor Performing SQL at the Database Level*.

Using Workload Tagging to Troubleshoot Poor Performing SQL at the Database Level

Use the following procedure to diagnose and troubleshoot poor performing SQL at the database level using Oracle Enterprise Manager. For more information about Oracle Enterprise Manager, see the documentation on Oracle Technology Network.

To use workload tagging to troubleshoot poor performing SQL at the database level

1. Make sure that workload tagging is enabled.
2. Log in to Oracle Enterprise Manager.
3. Navigate to the Database Control Top Activity page.
4. From the Top Sessions section, select the Top Sessions view, and then drill down on the Session ID that you want to troubleshoot.
5. Click the General tab.
6. In the Client section, note the values in the Current Client ID field.

   The format for this field is:

   component_name,servername,taskID,userID,FlowID:SarmID

   where:

   - component_name is the server component alias, for example SCCObjMgr_enu.
   - servername is the name of the Siebel Server from which the SQL originated.
   - taskID is the task ID of the user who generated the query.
   - userID is the login name of the user who generated the query.
   - FlowID is the flow ID of the task.
   - SarmID is the SARM ID of the task.

   For example:

   ```
   SCCObjMgr_enu,server1,10485770,SADMIN,000008814d7f1fe8:163
   ```

Specifying SQL Spooling in Siebel Developer Web Client

This topic is part of *Analyzing Generated SQL for Performance Issues*.

Optionally, after making configuration changes in Siebel Tools, you can spool the SQL that is generated by the Siebel application during runtime. You do this to troubleshoot configuration-related performance issues.

To spool the generated SQL into a trace file, start the Siebel application in the Siebel Developer Web Client (connecting to the Siebel database) using the command-line option `/s sql_trace_file` . For more information about installing and running the Siebel Developer Web Client, see *Siebel Installation Guide* .

The SQL trace file contains all of the unique SQL statements generated during the current session, and identifies the amount of time spent processing each one. The trace file can be opened in a text editor for examination after the session has ended. The SQL trace file, which is simply a text file holding the spooled SQL from the session, is overwritten during every new session.

You can specify the `/s sql_trace_file` option by modifying properties for the Start menu item or desktop shortcut from which the Siebel application is invoked. The following example shows a command line for spooling generated SQL from Siebel Call Center using the Siebel Developer Web Client:

```
"D:\Siebel\Client\bin\siebel.exe /c D:\Siebel\Client\bin\enu\uagent.cfg /s siebel_
sql.txt"
```

If you do not specify a path, then the SQL trace file is created in the Siebel client root `bin` directory, such as `D:\Siebel \Client\bin`.

You can programmatically start and stop SQL spooling though the Siebel Object Interfaces by using the TraceOn and TraceOff methods on the Application object. For more information about these methods, see *Siebel Object Interfaces Reference* .

Troubleshooting Performance Using SQL Trace Files

This topic is part of *Analyzing Generated SQL for Performance Issues*.

As described, you can generate SQL trace files related to your configuration changes, such as for a particular view you have configured. Analyze the contents of the SQL trace file to identify any possible performance issues.

As you look through the SQL trace file, be aware of factors such as:

- The number and complexity of SQL statements.
- Execution times for SQL statements. This is the SQL execution time plus the time it takes to return rows. It does not include time for client-side processing.
- Selection criteria in the `WHERE` clauses, indicating search specifications.
- Sorting criteria in the `ORDER BY` clauses, indicating sort specifications. (In general, it is better for a query to first filter data using `WHERE` clauses, in order to reduce the volume of data to be then sorted. Applying sorting criteria that match users' needs reduces the likelihood of users performing their own sort operations, which would require additional system resources.)
- The use of joins.

Note: If the same SQL statement is executed repeatedly, the Siebel application displays the entire statement for the first query. For each subsequent iteration of the same query, only the bind variables are displayed. You can recognize a query that is repeated by the specific set of bind variables it uses.

SQL statements are displayed for all queries, including housekeeping queries. These are queries that are necessary for system operation, such as looking up the user's login to obtain responsibilities, and determining today's alarms in the calendar. You will also see queries to the `S_LST_OF_VAL` table to populate picklists. Queries that populate views are also present in the SQL trace file, and are easily distinguishable based on the tables they access.

Troubleshooting Performance Using SQL Query Plans

This topic is part of *Analyzing Generated SQL for Performance Issues*.

If you identify a problematic query in the SQL trace file, then you can obtain more information about it by using the database query tool provided with the RDBMS, such as SQL*Plus for Oracle Database.

Copy and paste the SQL statement from the trace file into the database query tool, execute the query against the Siebel database, then generate a query plan. A query plan is a detailed reporting of various statistics about the query you executed. For an example of generating a query plan against the local database, see *Example of Obtaining a Query Plan*.

Use query plans to check:

- The use of indexes
- The use of temporary tables
- The use of sequential table scans

Finally, compare your results with a standard application (that is, not custom-configured) in order to identify any potentially slow queries. You can resolve many performance issues either by modifying search specifications or sort specifications, or by creating new indexes on the base table.

CAUTION: Only specially trained Oracle personnel can modify existing Siebel indexes. This restriction is enforced so that performance in other modules (such as Siebel EIM) is not adversely affected by any index modifications that you make to improve query performance through the user interface. For more information, see *Managing Database Indexes in Sorting and Searching*.

Consider any potential performance implications before modifying search specification and sort specification properties for a business component. By spooling out the SQL into trace files, you can analyze which indexes are likely to be used when your application queries the business component through each applet.

Run your query plans against datasets that are comparable to the production dataset. For example, you will not obtain useful results analyzing the performance of a query against a 30-record test dataset when the production database has 200,000 records.

You might find it useful to prioritize the views to examine, as follows:

- **First priority.** Views that are known to have the biggest performance bottlenecks.
- **Second priority.** Views that are accessed most frequently.
- **Third priority.** Views that are the most highly configured (as compared to the standard Siebel application).

Comparison with the standard Siebel application provides you with a benchmark for evaluation. It is often very useful to obtain a trace file from the standard Siebel application, following a preselected route through the views. Then you

obtain a separate trace file from the custom-configured application, following the same route as closely as possible. The two trace files are compared, noting differences in the bullet items listed previously.

> **Note:** When you review a query plan, keep track of the business object to which each query applies, You can tell where each new business object is being opened by searching for the `S_APP_QUERY` statement. The business object that was accessed is represented using the bind variable statements beneath the query.

Bind variables are the values that determine which records are brought back. The RDBMS substitutes the value of a bind variable into an SQL statement when the same SQL statement is being reused, generally in place of each occurrence of a question mark or series of question marks. For example, a business object bind variable is used in an `S_APP_QUERY` statement because the purpose of this statement is to open the business object.

Watch for the following indications of potential problems:

- Unnecessary fields are being accessed, especially ones not exposed in the user interface and not needed for calculated fields, nor used for passing values to detail records.
- Unnecessary joins are occurring, particularly to tables that are not being accessed.
- Unnecessary multiple joins are being made to the same table. This can indicate duplicate join or Multi Value Link (MVL) object definitions, or joins using the same foreign key.
- Multiple short queries similar to the following:

```
...FROM
 SIEBEL.S_ADDR_PER T1
```

When a short query appears many times, this generally indicates that an MVG without a primary join is being accessed by a list applet. The system is running a secondary query for each master record to obtain its detail records. The secondary queries are the short queries appearing in the log file. This is usually your best diagnostic indicator of the need for a primary join.

When a short query appears only once, it indicates the same situation, but accessed in a form applet. In either case, the cure is a primary join, as explained in *Using Primary ID Fields to Improve Performance*.

Example of Obtaining a Query Plan

This topic is part of *Analyzing Generated SQL for Performance Issues*.

The following procedure shows an example of obtaining a query plan when running against a local database using the Siebel Mobile Web Client. It is assumed that customers will obtain query plans using the RDBMS for their enterprise database, such as Oracle Database, IBM DB2, or Microsoft SQL Server.

To obtain a query plan for an SQL statement in your trace file

1. Log in to the local database, as described in *Siebel Remote and Replication Manager Administration Guide* .
2. In order to analyze an SQL statement from the SQL trace file, copy the SQL statement and paste it into the database query tool you are using.
3. Replace bind variable references with the corresponding bind variable values.
4. Execute the query.

   The query runs against the local database. The Statistics pane provides analysis information.

SQL Queries Against Database Data

This topic is part of *Analyzing Generated SQL for Performance Issues*.

The database that underlies Siebel CRM can be queried to obtain information on a read-only basis.

> **CAUTION:** Update queries must never be directly performed on the Siebel database. All data manipulation and restructuring must be performed through Siebel Tools or through the Siebel application.

Guidelines for Siebel Scripting

This topic provides guidelines for Siebel scripting using Siebel eScript or Siebel VB, or for using declarative alternatives in place of scripts. It contains the following information:

- *Using Declarative Alternatives to Siebel Scripting*
- *Siebel Scripting Guidelines for Optimal Performance*

Using Declarative Alternatives to Siebel Scripting

This topic is part of *Guidelines for Siebel Scripting*.

Often, customers use scripts for data validation, responses to data changes, or other purposes that might best be addressed through declarative means: by defining properties or specifying business service method invocation using Siebel Tools.

Scripting is often unnecessary and must be minimized or avoided because it can introduce performance problems, add risk and complexity, require greater maintenance, and duplicate functionality already available in Siebel CRM.

For example, the Validation field property, which allows for common VB expressions and comparison operators, can be used to perform field validation or string manipulation of data entered through the user interface or through Siebel Object Interfaces.

Expressions for the Validation property can include methods such as LoginId(), LoginName(), LookupValue(), ParentFieldValue(), PositionId(), PositionName(), Today(), and so on.

The Force Case field property can also be useful in a data-validation context, such as to ensure that personal names entered have initial capital letters.

For more information on supported expressions and operators, see *Siebel Developer's Reference* .

Setting the Auto Primary property on MVL object definitions can also help you achieve results that you might otherwise use scripting for. For example, if your business requirement is to assign the first record in an MVG as the primary record (for example, primary address or primary owner), then set Auto Primary to the value Default. For more information about using Primary ID fields, see *Using Primary ID Fields to Improve Performance* and see *Configuring Siebel Business Applications* .

Scripting can be used in combination with declarative methods, such as to present customized error messages that guide users to enter data appropriately for each field subject to validation rules.

Functionality such as custom responses to data changes, which can often be handled through scripting, might best be addressed through declarative means. Such mechanisms, many of which can be used in combination, include:

- User properties on applets, business components, fields, controls, list columns, and other object definitions (for example: Required, Pre-Default, Post Default, Search Spec, Type Field, or Type Value)
- Siebel Workflow
- State model
- Siebel Personalization
- Run-time events
- Named methods
- Business services
- Visibility configuration

For more scripting guidelines, see *Configuring Siebel Business Applications* .

Siebel Scripting Guidelines for Optimal Performance

This topic is part of *Guidelines for Siebel Scripting*. It provides guidelines for appropriate use of Siebel scripting using Siebel eScript or Siebel VB. For more information about these and other guidelines, see:

- *Siebel eScript Language Reference*
- *Siebel VB Language Reference*
- *Siebel Object Interfaces Reference*
- *Configuring Siebel Business Applications*
- *Using Siebel Tools*
- *Configuring Siebel Open UI*

The following are some guidelines for appropriate use of Siebel scripting:

- **Use declarative alternatives.** Try all other possibilities before using scripting to accomplish a functional requirement. See also *Using Declarative Alternatives to Siebel Scripting*.
- **Use browser scripts for simple client-side functions such as field validation.** Browser scripts are best used to perform simple procedural logic on the client side, such as performing field validation, or displaying blocking messages or alerts to users. Some such uses, particularly field validation, can reduce server round trips. Using more complex browser scripts, however, might reduce performance.

  For example, using Set or Get Profile attribute calls, or invoking multiple business service methods, can require more server round trips and lead to performance problems. Adding extra functionality to scripts that display messages can have a similar effect.

  > **Note:** Setting the Immediate Post Changes field property has a similar effect on server round trips. Use this property only for constrained picklists and calculated fields that must be updated dynamically.

- **Do not return large result sets from server business services to browser scripts.** Browser scripts that invoke server scripts must return simple values or a single record, and must not return large result sets.
- **Minimize scripting on field-level or control-level events.** Field-level or control-level events are fired more often than most other types of events. Consequently, invoking scripts from such events can dramatically

impact scalability. Avoid scripting frequent events, or simplify scripts on these events. Examples of such events include BusComp_PreGetFieldValue(), WebApplet_PreCanInvokeMethod(), and WebApplet_ShowControl().

- **Use simple scripts on applet-level and business component-level events.** Scripts written on events for applets or business components (for example, for Change Record events) must be very simple, because such events are fired often. Complex or I/O-intensive operations in such events will adversely affect performance.
- **Caching data in Siebel eScript scripts.** Executing the same SQL statements from various locations in a Siebel eScript script can generate an excessive number of script API calls and a redundant number of business component queries. In order to reduce the performance impact (assuming that data does not change between invocations), you can cache a limited set of data within your scripts. (In some cases, you might not want to cache data at the script level, such as if the data that needs to be cached is too complex or too large.)
- **Declare your variables.** Declaring your variables and specifying their data type, as appropriate, can use less memory and improve performance.
- **Destroy any created objects when you no longer need them (Siebel eScript).** Theoretically, the Siebel eScript interpreter takes care of object cleanup. However, complex code involving many layers of object instantiation can in some cases cause the interpreter not to release objects in a timely manner. Destroying or releasing objects helps to minimize the impact on resources such as server memory.

  Explicit destruction of Siebel objects must occur in the procedure in which they are created. To destroy an object in Siebel eScript, set it to NULL, or set the variable that contains it to another value. Destroy objects in reverse order of creation; that is, destroy child objects before you destroy parent objects.
- **Verify that your script is defined on the appropriate method.** A script that is not defined on the right method might have a performance impact. For example, if special code needs to be run at the record level when an insert or update is done, then it is better to invoke a script from BusComp_WriteRecord() rather than BusComp_SetFieldValue(). The reason for this is that SetFieldValue events are fired much more often than WriteRecord events. Limit your use of specialized invocation methods.
- **Verify that your script is implemented in the right view.** A script that is not implemented in the right view might cause significant performance impact. Verify that this script is implemented in the right place in the configuration, based on data manipulations, navigation requirements, and business requirements in general.
- **Avoid redundant repository object settings.** Do not perform unnecessary object validation. Each method invocation that you perform has a performance cost. Details on this issue regarding field activation, for example, are provided below.
- **Use the ActivateField() method sparingly (Siebel eScript).** Do not activate a field if you will not use it. Use the ActivateField() method sparingly. Using this method increases the number of columns retrieved by a query, and can lead to multiple subqueries involving joins. These operations can use a significant amount of memory, and can degrade application performance.

  Do not perform any unnecessary field activation (for fields that are already active). Each method invocation that you perform has a performance cost.

  - Do not activate system fields, because they are already activated by default. Such fields include Created, Created By, Updated, and so on.
  - Do not activate any other fields that are already active. Check the Force Active field property in Siebel Tools to see whether you need to activate it.
- **Use the ExecuteQuery() method sparingly (Siebel eScript).** Removing calls to execute a business component, using the method ExecuteQuery(), can yield significant performance benefit. It is better practice to use shared variables to share values of specific business component records across scripts than to separately invoke ExecuteQuery() in each script.

- **Use the SetSearchSpec() method rather than NextRecord() method (Siebel eScript).** You can improve performance by using the SetSearchSpec() method to get a specific record, rather than using the NextRecord() method to go through a list of retrieved methods until a specific record is found.
- **Use ForwardOnly cursor mode (Siebel eScript).** Use the ForwardOnly cursor mode for ExecuteQuery() unless ForwardBackward is required. Using ForwardBackward uses a significant amount of memory, which can degrade application performance.
- **Use appropriate error handling.** Appropriate error handling can help maintain optimal performance. Although error handling is important, it also has a performance cost. For additional guidelines for using error handling in scripts, see 477766.1 (Article ID) on My Oracle Support. This document was previously published as Siebel Technical Note 514.
- **Avoid nested query loops.** Nested query loops can involve a large number of subqueries and can significantly impact performance. Use this technique very sparingly. Implement a nested query loop in the correct order in order to minimize the number of iterations. Be aware that a nested query loop can be invoked implicitly, depending on how your script is written.
- **Use the this object reference (Siebel eScript).** The special object reference this is eScript shorthand for "this (the current) object." Use it in place of references to active business objects and components.

  For example, in a business component event handler, use this in place of `ActiveBusComp()`, usage of which can have a significant performance impact. Refer to the following example:

```
function BusComp_PreQuery()
{
this.ActivateField("Account");
this.ActivateField("Account Location");
this.ClearToQuery();
this.SetSortSpec( "Account(DESCENDING)," +
t" Account Location(DESCENDING)");
this.ExecuteQuery();
return (ContinueOperation);
}
```

- **Use the Switch construct (Siebel eScript).** The Switch construct directs the program to choose among any number of alternatives that you require, based on the value of a single variable. Using this construct offers better performance than using a series of nested If statements, and is easier to maintain.
- **Use the Select Case construct (Siebel VB).** The Select Case construct directs the program to choose among any number of alternatives that you require, based on the value of a single variable. Using this construct offers better performance than using a series of nested If statements, and provides other benefits.
- **Test your custom scripts.** Make sure that your scripts are fully tested and optimized, and are no more complex than required to meet your business needs.
- **Set environment variables to improve performance.** When you use scripting, you might see some performance benefits from setting the following environment variables on the Siebel Server. After configuring these variables, you must restart the Siebel Server.

  (Windows) Log in and add the following environment variables:

```
Name = USE_NEW_RM, Value = 1
Name = USE_NEW_MM, Value = 1
```

  (UNIX) Log in and add the following environment variables:

```
setenv USE_NEW_RM = 1
setenv USE_NEW_MM = 1
```

Guidelines for Data Objects Layer

This topic describes guidelines for configuring selected elements in the data objects layer for optimal performance. It contains the following information:

- *Multilingual LOVs Query and Cache Performance*
- *Managing Database Indexes in Sorting and Searching*
- *Reusing Standard Columns*
- *Limiting Extension Columns*
- *Case Insensitivity and Performance*

Multilingual LOVs Query and Cache Performance

This topic is part of *Guidelines for Data Objects Layer*.

Multilingual List of Values (MLOV) fields are implemented below the business component level. Fields that point to MLOVs with enabled target columns return display values that match the current language setting for the session.

For display, the underlying language-independent code is converted to its corresponding display value using a Siebel application lookup. For searching and sorting, however, a database join to the list of values table (`S_LST_OF_VAL`) is performed. Make sure that any configuration directly involving the `S_LST_OF_VAL` table is compatible with your Siebel application MLOV functionality.

When a view with MLOVs is displayed for the first time, a separate query on the `S_LST_OF_VAL` table is made for each field that has an MLOV. The query obtains all of the display values for that MLOV and writes the values to the LOV cache in memory. When the view is subsequently displayed during the same session, the values are obtained from the cache rather than by issuing another query.

> **Note:** Displaying multiple records in a list applet that contains one or more MLOV fields will cause memory consumption to increase, and can produce poor performance. The problem manifests particularly when multiple fetches are performed against a given logical result set; that is, you scroll through records. It can also manifest when client-side export is performed to automate this behavior, or any time the NextRecord method is invoked repeatedly on the business component. It is generally recommended to use MLOV fields sparingly in list applets, or to disable client-side export from list applets containing MLOVs.

For more information about configuring MLOVs, see *Configuring Siebel Business Applications* and *Siebel Global Deployment Guide* .

Managing Database Indexes in Sorting and Searching

This topic is part of *Guidelines for Data Objects Layer*.

A *database index* is a data structure in the RDBMS that is associated with a table. It provides references to all records in the table for quick lookup and filtering, and is sorted in a particular order for sorting in that order quickly. The Siebel database uses an index to efficiently retrieve and sort the result set of a query.

Indexes provided in the Siebel Data Model are tuned for optimal performance of standard Siebel applications. When you add new business components with custom sorting or filtering requirements, you need to make sure that a database index is present that supports the requirement and delivers the result set efficiently. You might need to add new indexes.

You add indexes using the Index and Index Column object types. The index is added in the database as a result of its being created in Siebel Tools and database extensions being applied.

> **Note:** The addition of custom indexes does not always improve performance and can reduce performance in some cases. The incremental value of an index depends in large part on the heterogeneity and distribution of the data.

When data is heterogeneous, all or most of the values are unique (such as with row ID values, which are unique). The less heterogeneous the data (that is, the more homogeneity or repeated instances of values), the less benefit the index offers relative to its costs.

For Boolean fields, indexes generally offer little value. Some performance benefit might be found when querying for the least commonly represented values. Little or no benefit is found when querying on more commonly represented values or values that are evenly distributed. Similar guidelines apply for other homogeneous data, such as fields that are constrained to a list of values.

Indexing generally improves performance of SELECT operations. However, it can significantly reduce performance for batch UPDATE and INSERT operations, such as are performed by Siebel EIM.

Discuss any custom index requirements with Oracle Advanced Customer Services. Contact your Oracle sales representative to request assistance from Oracle Advanced Customer Services.

Sort Specification

The Sort Specification property for a business component, picklist, or predefined query orders the records retrieved in a query, and serves as the basis for the ORDER BY clause in the resulting SQL issued. An index needs to be present that supports the order specified in the sort specification. Otherwise, the RDBMS engine physically sorts the entire result set in a temporary table.

The index needs to include the base columns for all of the fields, and to use them in the same order. There can be more columns specified in the index than are used in the sort specification, but the reverse is not true.

For example, the sort specification Last Name, First Name in the Contact business component is supported by at least one index on the S_CONTACT base table. One of these indexes is called S_CONTACT_U1, and it contains the LAST_NAME, FST_NAME, MID_NAME, PR_DEPT_OU_ID, OWNER_PER_ID, and CONFLICT_ID columns, in that order. If you want a sort specification that orders contacts in first-name order, then you would need to create a custom index.

Do not sort on joined columns, because indexes cannot be used.

Search Specification

The Search Specification property for a business component, applet, link, or picklist selectively retrieves rows from the underlying table that meet the criterion specified in the property. The search specification is the basis for the WHERE clause in the resulting SQL issued. An index needs to be present that supports the criterion. Otherwise, the RDBMS might scan through all rows in the table rather than only those to be returned by the query.

The index needs to contain all of the columns referenced by fields in the search specification.

In Sales Rep views such as My Accounts or where organization access control is implemented, if the user queries or sorts columns that are denormalized to the intersection table (for example, NAME and LOC in S_ORG_EXT), then performance

is likely to be good. The Siebel application uses the intersection to determine visibility to records in the base table, and indexes can be used on the intersection table to improve performance.

For related information, see *Reusing Standard Columns*.

> **Note:** If a query or sort includes columns that are not denormalized to the intersection table, then performance is likely to degrade, because indexes are not used.

Reusing Standard Columns

This topic is part of *Guidelines for Data Objects Layer*.

The architecture and data model of your application has been tuned for best performance. This optimization is achieved by using proper indexes, data caching, and efficient SQL generation, and also by denormalizing columns on certain tables. These denormalized columns are indexed so that the application can improve the performance of complex SQL statements by using these columns for search or sort operations instead of the columns of the original tables.

> **Note:** Do not remap existing fields, especially those based on User Key columns, to other columns in the same table.

> **CAUTION:** Do not use custom denormalized columns without the assistance of Oracle Advanced Customer Services. Denormalized columns can improve performance by allowing indexes to be placed directly on an intersection table, rather than on its master or detail table. However, if this is configured improperly, then the data in the denormalized column can become out of sync with its source. This can result in several problems ranging from inconsistent sorting to corrupt data. Contact your Oracle sales representative to request assistance from Oracle Advanced Customer Services.

Example: Reusing NAME and LOC in S_ORG_EXT Table

The columns NAME and LOC of the S_ORG_EXT table are denormalized into ACCNT_NAME and ACCNT_LOC in the S_ACCNT_POSTN table.

When sorting accounts by name and location in views where the Visibility Applet Type property is set to Sales Rep, the Siebel application uses the denormalized columns ACCNT_NAME and ACCNT_LOC of the S_ACCNT_POSTN table. Doing so allows the use of an index.

If the account name and location were stored in extension columns (for example, X_NAME and X_LOC), then these columns would have to be used for sorting instead of NAME and LOC. Even if these extension columns were indexed, the application could not use an existing index to create the necessary joins and sort the data, because the index is on S_ORG_EXT and not on S_ACCNT_POSTN. Therefore, the result would be a significant decrease in performance.

Query Plan for My Accounts View

The first SQL statement is generated by the standard My Accounts view. The query plan shows that the database uses numerous indexes to execute the statement.

```
SELECT
 T1.LAST_UPD_BY,
 T1.ROW_ID,
 T1.CONFLICT_ID,
 .
 .
```

```
 .
 T10.PR_EMP_ID,
 T2.DUNS_NUM,
 T2.HIST_SLS_EXCH_DT,
 T2.ASGN_USR_EXCLD_FLG,
 T2.PTNTL_SLS_CURCY_CD,
 T2.PAR_OU_ID
FROM
 SIEBEL.S_PARTY T1
 INNER JOIN SIEBEL.S_ORG_EXT T2 ON T1.ROW_ID = T2.PAR_ROW_ID
 INNER JOIN SIEBEL.S_ACCNT_POSTN T3 ON (T3.POSITION_ID = ?, 0.05)
 AND T2.ROW_ID = T3.OU_EXT_ID
 INNER JOIN SIEBEL.S_PARTY T4 ON (T4.ROW_ID = T3.POSITION_ID, 0.05)
 LEFT OUTER JOIN SIEBEL.S_PRI_LST T5 ON T2.CURR_PRI_LST_ID = T5.ROW_ID
 LEFT OUTER JOIN SIEBEL.S_INVLOC T6 ON T2.PR_FULFL_INVLOC_ID =
 T6.ROW_ID
 LEFT OUTER JOIN SIEBEL.S_ORG_EXT T7 ON T2.PAR_OU_ID = T7.PAR_ROW_ID
 LEFT OUTER JOIN SIEBEL.S_ORG_EXT_SS T8 ON T1.ROW_ID = T8.PAR_ROW_ID
 LEFT OUTER JOIN SIEBEL.S_INT_INSTANCE T9 ON T8.OWN_INST_ID =
 T9.ROW_ID
 LEFT OUTER JOIN SIEBEL.S_POSTN T10 ON T2.PR_POSTN_ID = T10.PAR_ROW_ID
 LEFT OUTER JOIN SIEBEL.S_USER T11 ON T10.PR_EMP_ID = T11.PAR_ROW_ID
 LEFT OUTER JOIN SIEBEL.S_ADDR_ORG T12 ON T2.PR_ADDR_ID = T12.ROW_ID
 LEFT OUTER JOIN SIEBEL.S_INDUST T13 ON T2.PR_INDUST_ID = T13.ROW_ID
 LEFT OUTER JOIN SIEBEL.S_ASGN_GRP T14 ON T2.PR_TERR_ID = T14.ROW_ID
 LEFT OUTER JOIN SIEBEL.S_POSTN T15 ON T3.POSITION_ID = T15.PAR_ROW_ID
 LEFT OUTER JOIN SIEBEL.S_USER T16 ON T15.PR_EMP_ID = T16.PAR_ROW_ID
 LEFT OUTER JOIN SIEBEL.S_ORG_SYN T17 ON T2.PR_SYN_ID = T17.ROW_ID
 LEFT OUTER JOIN SIEBEL.S_ORG_BU T18 ON T2.BU_ID = T18.BU_ID AND
 T2.ROW_ID = T18.ORG_ID
 LEFT OUTER JOIN SIEBEL.S_PARTY T19 ON T18.BU_ID = T19.ROW_ID
 LEFT OUTER JOIN SIEBEL.S_ORG_EXT T20 ON T18.BU_ID = T20.PAR_ROW_ID
WHERE
 ((T2.INT_ORG_FLG != 'Y' OR T2.PRTNR_FLG != 'N') AND (T3.ACCNT_NAME >= ?))
ORDER BY
 T3.POSITION_ID, T3.ACCNT_NAME
Query plan : T3(S_ACCNT_POSTN_M1),T2(S_ORG_EXT_P1),T1(S_PARTY_P1),T15(S_POSTN_
U2),T10(S_POSTN_U2),T4(S_PARTY_P1),T12(S_ADDR_ORD_P1),T13(S_INDUST_P1),T7(S_ORG_
EXT_U3),T16(S_USER_U2),T11(S_USER_U2),T17(S_ORG_SYN_P1),T6(S_INVLOC_P1),T5(S_PRI_
LST_P1),T14(S_ASGN_GRP_P1),T18(S_ORG_BU_U1),T19(S_PARTY_P1),T20(S_ORG_EXT_
U3),T8(S_ORG_EXT_SS_U1),T9(se)
```

Query Plan for My Accounts View (Different ORDER BY Clause)

The second SQL statement generated in My Accounts, below, has a different `ORDER BY` clause. Even though the columns `NAME` and `LOC` of `S_ORG_EXT` are indexed, the database cannot use this index. Performance decreases from the use of a temporary table. The same behavior occurs if the `ORDER BY` clause uses the columns `X_NAME` and `X_LOC` instead of `NAME` and `LOC`.

The following example shows a different `ORDER BY` clause than the previous example query plan.

```
WHERE
 ((T2.INT_ORG_FLG != 'Y' OR T2.PRTNR_FLG != 'N') AND)
 T3.ACCNT_NAME >= ?))
ORDER BY
 T3.ACCNT_NAME, T3.POSITION_ID
Query plan : TEMPORARY TABLE
T3(S_ACCNT_POSTN_M1),T2(S_ORG_EXT_P1),T1(S_PARTY_P1),T15(S_POSTN_U2),T10(S_POSTN_
U2),T4(S_PARTY_P1),T12(S_ADDR_ORG_P1),T13(S_INDUST_P1),T7(S_ORG_EXT_U3),T16(S_
USER_U2),T11(S_USER_U2),T17(S_ORG_SYN_P1),T6(S_INVLOC_P1),T5(S_PRI_LST_P1),T14(S_
ASGN_GRP_P1),T18(S_ORG_BU_U1),T19(S_PARTY_P1),T20(S_ORG_EXT_U3),T8(S_ORG_EXT_SS_
U1),T9(se)
```

Limiting Extension Columns

This topic is part of *Guidelines for Data Objects Layer*.

Adding extension columns to base tables can also affect performance, depending on the data type and length, the joins and queries used, the number of columns, and other considerations. For example, it is recommended not to add more than three columns of the data type CLOB to a base table. See also *Reusing Standard Columns*.

Case Insensitivity and Performance

This topic is part of *Guidelines for Data Objects Layer*.

Case-sensitive queries perform better than case-insensitive queries, where queries include wildcards. However, you can support case-insensitive queries and reduce the performance impact through appropriate configuration. Siebel CRM applications are case-sensitive by default. You can enable case insensitivity for specified columns. End users can force case-sensitive or case-insensitive queries.

You can use the Case Insensitivity Wizard to configure Siebel database columns to support case-insensitive queries. For best performance, it is recommended to accept the recommendations of this wizard. For each specified column, the wizard creates a case-insensitive column populated with upper-case characters and creates a case-insensitive index. Before you run the wizard, verify that the columns that are to support case-insensitive queries are already indexed, to improve performance for case-insensitive queries (by avoiding table scans).

The Case Insensitivity Wizard formats the search criteria for the applicable columns into upper case and appends a wildcard to the search string. The wildcard is used to match data in the specialized column populated with upper-case characters.

A trailing wildcard causes performance issues with Oracle Database and IBM DB2. However, you can resolve this problem by setting the user property Use Literals For Like. Instead of the host variable, a literal string like 'SMITH%' is sent to the database server, which determines that the wildcard is in the last position and uses the appropriate index.

If you enable case insensitivity for a user key column that is denormalized to related visibility tables, then also enable case insensitivity for the denormalized column. For example, if you enable case-insensitivity for `Account.Name`, then also enable it for the `S_ACCNT_POSTN` and `S_ORG_BU` denormalized columns for `Account.Name`. After you enable case insensitivity for user key fields like Last Name, the Contacts View is slower. For more information, see 536211.1 (Article ID) on My Oracle Support.

For more information about configuring case insensitivity and about different ways of using the Case Insensitivity Wizard, see *Configuring Siebel Business Applications* , *Siebel Database Upgrade Guide* , and *Siebel Global Deployment Guide* . For more information about performing case-insensitive queries, see *Siebel Fundamentals Guide* and *Siebel Applications Administration Guide* .

Before you configure case insensitivity, a thorough review of business requirements and performance criteria is highly recommended. In addition, if the feature is enabled, then conduct a performance test with a full copy of the production database. It is also recommended that Oracle Advanced Customer Services be engaged to optimize the configuration and review requirements. Contact your Oracle sales representative to request assistance from Oracle Advanced Customer Services.

Related Books

Configuring Siebel Business Applications

Siebel Database Upgrade Guide

Siebel Global Deployment Guide

Siebel Fundamentals Guide

Siebel Applications Administration Guide

Guidelines for Business Objects Layer

This topic describes guidelines for configuring selected elements in the business objects layer for optimal performance. It contains the following information:

- *Using the Cache Data Property to Improve Business Component Performance*
- *Limiting the Number of Active Fields*
- *Guidelines for Using Calculated Fields*
- *Using Properties to Improve Picklist Performance*
- *Using Primary ID Fields to Improve Performance*
- *How the Check No Match Property Impacts Performance*

Using the Cache Data Property to Improve Business Component Performance

This topic is part of *Guidelines for Business Objects Layer*.

To cache on the Siebel Application Object Manager the content of a business component for subsequent use in the same user session (given the same query and search specification), the property Cache Data property must be set to TRUE for the business component. Setting Cache Data to TRUE is appropriate for semi-static data that might be subject to repetitive queries, but that is unlikely to change during the user session.

For some business components, Cache Data is set to TRUE by default. This is done, for example, for the Internal Product business component.

Cache Data must be FALSE for business components that represent transactional data that might change within a user session.

> **Note:** It is recommended that you do not set the Cache Data property of a business component to TRUE and also set the Use Primary Join property of a multi-value link to TRUE. If Siebel CRM modifies the primary record of a multi-value group business component in this situation, then it might not update data for the cached parent.

See also *Using Properties to Improve Picklist Performance*.

Limiting the Number of Active Fields

This topic is part of *Guidelines for Business Objects Layer*.

Field object definitions are instantiated for each business component when the business component is instantiated, such as by a user navigating to a view containing an applet based on the business component. All such instantiated fields are included in the SELECT statements in generated SQL that is issued to the Siebel database, even fields that are not represented in the user interface with a corresponding list column or other field control.

The set of fields that is instantiated includes those for which the Force Active property is set to TRUE. The Force Active setting of TRUE indicates to the system that it must obtain data for the field every time the business component is accessed, even if the field is not displayed in the current applet; this adds the field to the SQL query each time.

When Force Active is set to TRUE, there is an associated performance cost. Force Active affects performance more significantly when fields are based upon MVLs or joins, because the Siebel application has to create the relationships in the SQL query to retrieve data for these columns.

In most cases, the Force Active property is not required. In general, do not set Force Active to TRUE unless strictly necessary.

Use Force Active only when the field must be included in generated queries, but the field does not appear in the user interface.

Guidelines for Using Calculated Fields

This topic is part of *Guidelines for Business Objects Layer*.

Calculated fields provide a convenient way to access and display data in the user interface that is not directly stored in a table. However, calculated fields have a cost associated with them. Consequently, it is important to use them appropriately to fulfill your requirements, and not to misuse them.

Each calculated field is evaluated whenever the business component is queried to provide a value for the field. Extensive use of calculated fields, or usage in certain contexts, can impact performance. Some guidelines are as follows:

- Use calculated fields sparingly. Be sure there is a valid business case for their usage.
- Minimize the complexity of the expressions defined in your calculated fields.
- Minimize the use of calculated fields that perform Sum, Count, Min, or Max calculations, such as for detail records in an MVG business component. In particular, avoid using such fields in list applets, or in More Info form applets. The cost of using such expressions can be significant depending on the number of detail records.

  Whenever data is totaled there are performance implications. It is important to limit the number of records being totaled. For example, totaling the line items in a Quote or Expense report is not resource-consuming. However, summing the expected revenue for all Opportunities is resource-consuming. The latter occurs when you generate a chart. However, charts tend not to be generated frequently. Accessing the Opportunities list view for routine searches and data entry is done frequently.

  > **CAUTION:** Never put a sum([MVfield]) in a list column. Doing so requires that a separate query be executed for each record in the list, which is a significant performance issue.

- Avoid defining calculated fields using complex expressions that provide different values depending on the current language.
- Avoid using a calculated field to directly copy the value of another field.
- Avoid including calculated fields in search specifications, particularly if the calculated fields use functions that are not supported by the underlying RDBMS.
    - If the RDBMS supports the function, then it will have algorithms for performing the calculations efficiently and will return the calculated values with the result set. However, if functions such as EXISTS, Max, or Count are included, then multiple subqueries can be performed, impacting performance. In this case, the calculations can take place before the results are returned.
    - If the function is *not* supported in the RDBMS, then the Siebel application might have to rescan the entire result set to perform the desired calculation, considerably increasing the time it takes to obtain the results of the query. In this case, the calculations must to be performed in memory, on the Siebel Application Object Manager or client

> **Note:** Even if the calculated field is supported at the RDBMS level, there can be other reasons why a search specification on a calculated field might result in poor performance, such as the lack of an index (for example, when using the LIKE function) supporting the search specification. See *Managing Database Indexes in Sorting and Searching*.

Using Properties to Improve Picklist Performance

This topic is part of *Guidelines for Business Objects Layer*.

To cache the content of certain picklists for subsequent use in the same user session, the Cache Data property must be set to TRUE for the PickList Generic business component. By default, this property is FALSE.

> **Note:** Picklists based on PickList Generic display LOV data, which is unlikely to change during the user session, and are thus suitable for caching. Picklists based on other business components display data that could change during a user's session and is thus generally unsuitable for caching.

Also set the Long List property to TRUE for each applicable Pick List object definition. When Long List is TRUE, the focus is not maintained on the current picklist record, thus improving performance for picklists with many records. The default setting of Long List varies for each Pick List object definition.

Using Primary ID Fields to Improve Performance

This topic is part of *Guidelines for Business Objects Layer*.

MVGs configured without Primary ID fields require separate queries to display each parent record and each set of child records. For example, for a list applet that displays 10 records and two MVGs per record, a total of 21 queries would be required to populate the applet: one query to populate the parent records and 20 additional queries (two per parent record) to populate the MVGs. The number of queries executed is many times the number actually required.

You can avoid unnecessary queries by configuring a Primary ID field on the master business component. The Primary ID field serves as a foreign key from a parent record to one primary child record in the detail business component. This allows the application to perform a single query using an SQL join to display values for the parent record and the

primary child record in the applet. In other words, it defers having to perform additional queries for the MVG until the user opens the MVG and displays a list of all child records.

List applets receive the most performance benefit from using Primary ID fields because list applets typically access a large number of records and each record can have one or more MVGs associated with it. The Primary ID field avoids having to submit queries for each MVG for every parent record.

Form applets can also benefit from Primary ID fields, even though in form applets only one parent record is accessed at a time. A Primary ID field allows the application to submit a single query for each new parent record displayed, rather than having to perform multiple queries for every MVG on the form applet. This can improve performance as the user moves from one record to another.

In some circumstances, configuring a Primary ID field is not desirable or feasible:

- When Microsoft SQL Server is being used, and the creation of the primary join would create a double-outer-join situation prohibited by the Microsoft software.
- When the only purpose of the multi-value field is to sum detail record values.

For information on how to configure Primary ID fields, see *Configuring Siebel Business Applications* .

How the Check No Match Property Impacts Performance

This topic is part of *Guidelines for Business Objects Layer*.

In most cases, the Check No Match property of a Multi Value Link object definition (used to implement Primary ID fields) must be set to `FALSE`. Setting the Check No Match property to `TRUE` could negatively impact performance, especially in situations where most parent records do not have child records defined in an MVG.

The Check No Match property defines whether a separate query must be used to populate an MVG when no child record is found through a primary join.

- When Check No Match is set to `FALSE`, the application does the following:
  - If a parent record's Primary ID field is invalid or has the value of NULL, then a secondary query is performed to determine whether there are child records in the MVG. If there are no child records, then the Primary ID field is set to the value NoMatchRowId.
  - If a parent record's Primary ID field has the value NoMatchRowId, then the application does not perform a secondary query, because NoMatchRowId indicates that there are no child records in the MVG. Avoiding these extra SQL queries improves performance.

  > **Note:** NoMatchRowId is not a permanent setting; the Primary ID field can be updated after it is set to NoMatchRowId.

- When Check No Match is set to `TRUE`, a separate SQL query is executed for each parent record in which the primary join did not find a primary child record. Doing this ensures that the multi-value field does not appear blank unless there are no child records. But executing these extra SQL queries decreases performance.

  It is appropriate to set the Check No Match property to `TRUE` in the following cases:
  - When the multi-value group allows records to be added without having to go through the MVG. For example, account addresses might actually be inserted through the Business Address multi-value group on the Contact business component instead of the Account business component.
  - When records can be added to a detail business component through Siebel EIM.

For more information about configuring Multi Value Link object definitions, see *Configuring Siebel Business Applications* .

Guidelines for User Interface Objects Layer

This topic describes guidelines for configuring selected elements in the user interface objects layer for optimal performance. It contains the following information:

- *Addressing Performance Issues Related to Grid Layout*
- *Maintaining Performance When Using Applet Toggles*

Addressing Performance Issues Related to Grid Layout

This topic is part of *Guidelines for User Interface Objects Layer*.

The grid layout feature allows developers to create effective and usable form applets for Siebel views. However, performance can be adversely affected by certain applet design choices.

Typically, such performance problems relate to the alignment of user interface controls such as labels and fields, and stem from the total number of cells in the grid-based form applet, including spacer cells. Performance impact will depend on the number of user interface elements, the applet size, and other factors.

You can optimize user interface performance by:

- Making stacked sets of labels or fields the same width. Doing so can reduce the number of adjacent spacer cells that you require.
- Aligning stacked sets of labels consistently.
- Making labels the same height as the adjacent fields.
- Eliminating horizontal or vertical spacer cells that you deem unnecessary.

> **Note:** Weigh all optional measures against possible usability concerns. Judicious use of spacing in your view layouts is generally appropriate for optimal usability.

For more information about using the grid layout, see *Configuring Siebel Business Applications* .

Maintaining Performance When Using Applet Toggles

This topic is part of *Guidelines for User Interface Objects Layer*.

Applet toggles are a useful feature where multiple applets based on different business components occupy the same location in a view. Which applet displays at one time depends on a field value in a parent applet (dynamic toggle) or on a user selection (static toggle).

Dynamic toggle applets are based on the same business component, while static toggle applets can be based on different business components.

In general, when configuring applet toggles for your Siebel application, particularly dynamic toggles, you can reduce memory and CPU usage for user application sessions by minimizing the number of applet toggles and fields per applet.

It is important to be aware of potential performance impact of using applet toggles, particularly dynamic toggles:

- When a user selects a record in a parent applet for a dynamic applet toggle, the business component and fields for all of the applet toggles are instantiated and cached in memory, and all of these fields are queried.

  This query is used to populate other applet toggles that might be displayed when the user changes the relevant field value in the parent record. However, each time the user selects a different record in the parent applet, all of the fields in the toggle business component are required.

  Also note that browser caching is not performed for views containing dynamic applet toggles.
- When a user navigates to a view containing a static applet toggle, the business component and fields for the default displayed applet is instantiated and cached in memory, and these fields are queried. Other business components are instantiated and cached, and other queries performed, when the user navigates to the other applets in the toggle.

In each case, cached objects remain in memory until the user navigates to a different screen.

13 Tuning Operating Systems and Databases for Performance

Tuning Operating Systems and Databases for Performance

This chapter describes tuning steps designed to improve the performance and scalability of your Siebel Enterprise Server or Siebel Application Interface installation. It also contains information about tuning your Siebel database on IBM DB2. It contains the following topics:

- *Tuning Microsoft Windows for Enhanced Siebel Server Performance*
- *Tuning the Siebel Server for All UNIX and Linux Operating Systems*
- *Tuning the Siebel Application Interface Computer for All Applicable UNIX and Linux Operating Systems*
- *Tuning the Siebel Application Interface for All UNIX and Linux Operating Systems*
- *Tuning Siebel CRM for AIX*
- *Tuning Siebel CRM for HP-UX*
- *Tuning Siebel CRM for Oracle Solaris*
- *Tuning Siebel CRM for IBM DB2*

Before doing any of the procedures in this chapter, you must have completed the minimum necessary steps described in the chapters about installation and configuration tasks for the Siebel Gateway, Siebel Enterprise, Siebel Server and Siebel Application Interface, as described in *Siebel Installation Guide* . See also *Siebel System Administration Guide* and *Siebel System Monitoring and Diagnostics Guide* .

> **Note:** The settings provided in this chapter are based on a controlled lab environment using a standard Siebel application, such as Siebel Call Center. The degree of performance gained by using these settings at your site depends on your implementation. Contact your vendor for additional tuning recommendations for your supported operating system. Information is provided where tuning recommendations apply. In general, it can be assumed that tuning is not required if no recommendations are provided. For example, no information is provided about tuning the kernel on Linux operating systems, because such tuning is not necessary in those cases.

Tuning Microsoft Windows for Enhanced Siebel Server Performance

This topic describes how you can configure settings for your Microsoft Windows operating system to optimize the performance of Siebel CRM. It contains the following information:

- *Maximizing Data Throughput*
- *Turning on the 4GT RAM Tuning Feature*

Maximizing Data Throughput

This topic is part of *Tuning Microsoft Windows for Enhanced Siebel Server Performance*.

For a server computer on which you install Siebel CRM software, changing the data throughput setting from *Maximize data throughput for file sharing* (default) to *Maximize data throughput for network applications* can result in the following benefits:

- Better symmetrical multi-processing (SMP) scalability
- Improved networking performance
- Allocation of more physical memory for your Siebel applications

Note: Where the Siebel database is on a server computer running Microsoft SQL Server, the setting *Maximize data throughput for network applications* is set by default, and is also generally recommended for optimal performance. Whether to keep or change this default depends on how you are using the server computer.

For more information about these settings, see Microsoft's documentation.

Turning on the 4GT RAM Tuning Feature

You can expand the per-process address limit from 2 GB to 3 GB. This setting reduces the amount of physical RAM available to the operating system from 2 GB to 1 GB. The difference (1 GB) is allocated to your applications. This feature is referred to as 4GT RAM tuning. For information about how to configure this setting, see Microsoft's documentation.

Note: Each Siebel process (Siebel Application Object Manager) cannot use more than 2 GB of RAM.

Tuning the Siebel Server for All UNIX and Linux Operating Systems

For all Siebel Server computers running on supported UNIX or Linux operating systems, setting the environment variables described in this topic can help you to manage your server resources appropriately and stay within appropriate CPU-usage limits. It contains the following information:

- *Environment Variable for Siebel Assert Creation*
- *Environment Variable for Operating System Resource Limits*
- *Environment Variables for Operating System Latches*
- *Tuning Kernel Settings*

Environment Variable for Siebel Assert Creation

This topic is part of *Tuning the Siebel Server for All UNIX and Linux Operating Systems*.

For Siebel Server computers or Siebel Application Interface computers, the environment variable SIEBEL_ASSERT_MODE determines whether assert files are created. With the default setting of 0, the creation of assert files is disabled, which conserves disk space and improves performance.

Set this variable to a nonzero value only if you are performing system diagnostics, and only in consultation with Oracle Global Customer Support. For more information about this variable, see *Siebel System Monitoring and Diagnostics Guide* .

Environment Variable for Operating System Resource Limits

This topic is part of *Tuning the Siebel Server for All UNIX and Linux Operating Systems*.

Set the environment variable SIEBEL_OSD_MAXLIMITS using one of the following methods (define the variable in the applicable profile for the Siebel Server):

- C Shell:

```
setenv SIEBEL_OSD_MAXLIMITS 1
```

- Korn Shell or Bourne Shell:

```
SIEBEL_OSD_MAXLIMITS=1;export SIEBEL_OSD_MAXLIMITS
```

Setting this variable to 1 specifies that operating system maximum values for resources will apply. Such resources might include coredumpsize, cputime, filesize, descriptors, maxmemory, and others.

Environment Variables for Operating System Latches

This topic is part of *Tuning the Siebel Server for All UNIX and Linux Operating Systems*.

Depending on the total number of tasks on the Siebel Server, you might need to set the environment variables described here in order to manage these loads. SIEBEL_OSD_NLATCH controls named latches and SIEBEL_OSD_LATCH controls unnamed latches. Latches, which are similar to mutexes (mutual exclusion objects), are used for communication between processes.

If SIEBEL_OSD_NLATCH and SIEBEL_OSD_LATCH are not defined, the values are 5000 and 1000, respectively. If these values are sufficient for the total number of tasks on the Siebel Server, then you do not need to set these variables. Do not set these variables to values lower than their default values.

> **Note:** Before changing these variables, stop the Siebel Server using the stop_server command, then run the cleansync utility.

Set SIEBEL_OSD_NLATCH and SIEBEL_OSD_LATCH on the Siebel Server computer based on the following formulas (define the variables in the applicable profile for the Siebel Server):

- SIEBEL_OSD_NLATCH = (7 times (cumulative MaxTasks for all components)) plus 1000
- SIEBEL_OSD_LATCH = 1.2 times (cumulative MaxTasks for all components)

Assume, for example, that you have enabled two multithreaded server components on the same Siebel Server: SCCObjMgr_enu and WfProcMgr. For SCCObjMgr_enu, `MaxTasks` = 700 and, for WfProcMgr, `MaxTasks` = 150. In this example, the parameter values must be set as follows:

- `SIEBEL_OSD_NLATCH` = 6950 = 7 times (700 plus 150) plus 1000
- `SIEBEL_OSD_LATCH` = 1020 = 1.2 times (700 plus 150)

> **Note:** Although the formulas presented here are expected to work for most deployments, in certain cases you might need to increase (for example, double) the values of these variables, and then monitor system behavior with the adjusted values. (An error message such as `SBL-OSD-00217: Error exceeded maximum number of latches` can sometimes be generated even when the variables are set according to the recommended formulas.

Tuning Kernel Settings

This topic is part of *Tuning the Siebel Server for All UNIX and Linux Operating Systems*.

To change the kernel settings

1. Using a text editor such as `vi`, open the `/etc/rc.net` file for editing.
2. Check the settings for all User Limits (ulimit) and make sure that they are set to -1 (unlimited), as follows:

   ```
   ulimit -a
   ```

   > **Note:** Work with your system administrator to change the set limits as appropriate for your UNIX or Linux operating system. On AIX, for example, update the `/etc/security/limits` file by changing all `ulimit` parameter values to -1 (unlimited).

3. Save your changes and exit the editor.
4. Restart the server computer to have the new settings take effect.

Tuning the Siebel Application Interface Computer for All Applicable UNIX and Linux Operating Systems

This topic describes how to configure the thread stack size for a Siebel thread on the Siebel Application Interface computer.

The default thread stack size is 512 KB, or 524288 bytes, for the Siebel threads in all UNIX and Linux operating systems supported for the computer on which you install the Siebel Application Interface.

In many cases, the default thread stack size for Siebel thread might be larger than necessary. With a larger thread stack size, the httpd process consumes more resources, and thus fewer users can be supported for each httpd process than with smaller sizes.

If you determine that a smaller thread stack size for the Siebel thread is more suitable for your Siebel applications, then you can set the `SIEBEL_OSD_PTHREAD_STACK_SIZE` environment variable at the operating system level to specify the size that you require, in bytes. Some Siebel CRM applications might require only the minimum thread stack size, while others

might require a size larger than this. Setting `SIEBEL_OSD_PTHREAD_STACK_SIZE` is at the customer's discretion, based on the nature of the Siebel applications running on each Siebel Application Interface.

> **Note:** The minimum Siebel thread stack size is 16 KB, or 16384 bytes. If you set `SIEBEL_OSD_PTHREAD_STACK_SIZE` to a value lower than 16384, then the effective value is 16384.

This topic contains the following information:

- *Configuring the Siebel Thread Stack Size on the Siebel Application Interface Computer*

Configuring the Siebel Thread Stack Size on the Siebel Application Interface Computer

This topic is part of *Tuning the Siebel Application Interface Computer for All Applicable UNIX and Linux Operating Systems*.

Use the following procedure to configure the thread stack size on the Siebel Application Interface computer.

To configure the Siebel thread stack size on the Siebel Application Interface computer

1. Open a new shell and execute a command similar to the following (this example sets the thread stack size to 64 KB):

   C Shell

   ```
   setenv SIEBEL_OSD_PTHREAD_STACK_SIZE 65536
   ```

   Korn or Bourne Shell

   ```
   export SIEBEL_OSD_PTHREAD_STACK_SIZE=65536
   ```
2. Stop the Siebel Application Interface.
3. Start the Siebel Application Interface.

Tuning the Siebel Application Interface for All UNIX and Linux Operating Systems

You might need to tune the Siebel Application Interface to run Siebel CRM on UNIX platforms, for performance reasons. As part of that tuning, you might need to configure additional anonymous users or specify settings such as the guest session time-out, for example. For more information, see *Siebel Installation Guide* and see *Siebel Security Guide* .

Related Topics

Specifying Static File Caching on the Siebel Application Interface

Tuning Siebel Application Object Manager Instances for Oracle Solaris

Related Books

Siebel Installation Guide

Siebel Security Guide

Tuning Siebel CRM for AIX

This topic provides instructions for configuring and tuning operating system settings and Siebel Enterprise Server components so that you can run Siebel applications on AIX. It contains the following information:

- *Tuning the Siebel Server for AIX*
- *Tuning Kernel Settings for AIX*

Tuning the Siebel Server for AIX

This topic is part of *Tuning Siebel CRM for AIX*.

AIX provides several environment variables that can be tuned to optimize Siebel Server performance. These environment variables and their values are used as start parameters when the Siebel Server is started. The following tables describe each of these environment variables and their recommended settings.

> **Note:** For more information about tuning the Siebel Server, see *Tuning the Siebel Server for All UNIX and Linux Operating Systems*.

Environment Variable	Value	Description
AIXTHREAD_SCOPE	S	Controls contention scope. S signifies system-based contention scope (1:1).
AIXTHREAD_MNRATIO	1:1	Controls the M:N ratio of number of kernel threads that must be employed to handle runnable pthreads.
AIXTHREAD_MUTEX_DEBUG	OFF	Maintains a list of active mutexes for use by the debugger.
AIXTHREAD_RWLOCK_DEBUG	OFF	Maintains a list of read-write locks for use by the debugger.
AIXTHREAD_COND_DEBUG	OFF	Maintains a list of condition variables for use by the debugger.

Environment Variable	Value	Description
SPINLOOPTIME	1000	Controls the number of times to retry a busy lock before yielding to another processor.

Environment Variable	Value	Description
YIELDLOOPTIME	>=*n* A number greater than or equal to the number of processors.	Controls the number of times to yield the processor before blocking on a busy lock (only for libpthreads). Set this variable, at the minimum, equal to the number of processors on the computer.
MALLOCOPTIONS	buckets,considersize,mu ltiheap:4	Include in the value an integer representing the number of processors on this computer. In this case, the value 4 would be used for a four-processor computer.
LDR_CNTRL	Example values: **LOADPUBLIC@MAXDATA=0x5000000** (this value, which specifies five segments, sets a maximum process size of 1.25 GB) **LOADPUBLIC@MAXDATA=0x6000000** (this value, which specifies six segments, sets a maximum process size of 1.5 GB)	The **LOADPUBLIC** option directs the system loader to load all modules requested by an application into the global shared library segment. Set **LDR_CNTRL** in the environment of the user, or, preferably, in the shell script that launches the executable program that needs the extra memory.

Tuning Kernel Settings for AIX

This topic is part of *Tuning Siebel CRM for AIX*.

There are several AIX kernel settings that you can tune for optimal Siebel Server or Siebel Application Interface performance under AIX. These include the Virtual Memory Management and TCP settings. You must have root privileges to modify these settings. Use the `vmo`, `ioo`, and `no` commands to tune the AIX kernel. For more information about AIX kernel settings, including several that are not mentioned in this guide, see 1097858.1 (Article ID) on My Oracle Support. See also your operating system vendor's documentation.

To change the kernel settings

1. Using a text editor such as `vi`, open the `/etc/rc.net` file for editing.
2. Modify settings using `ioo` and `no`, following the guidance of 1097858.1 (Article ID) on My Oracle Support.

   Note: Use default values for settings for `vmo`.

3. Save your changes and exit the editor.
4. Make sure that the rpc.statd and rpc.lockd daemons run on the Siebel Server computer and on the server computer where the Siebel File System is located. Then set the number of threads for the rpc.lockd daemon on each applicable server computer.

   It is recommended to increase the number of rpc.lockd daemon threads from the default. If possible, use the maximum number of threads, which is 511. System degradation can occur and logins might be blocked if the rpc.lockd daemon is not configured to handle a large number of lock requests. For example, you might execute commands like this:

   ```
   chssys -s rpc.lockd -a 511
   stopsrc -s rpc.lockd; startsrc -s rpc.lockd
   ```

5. Restart the server computer to have the new settings take effect.

Tuning Siebel CRM for HP-UX

This topic provides instructions for configuring and tuning operating system settings and Siebel Enterprise Server components so that you can run Siebel applications on HP-UX. It contains the following information:

- *Tuning Kernel Settings for HP-UX*
- *Setting Permissions for the HP-UX Scheduler*

Tuning Kernel Settings for HP-UX

This topic is part of *Tuning Siebel CRM for HP-UX*.

Modify the HP-UX kernel parameters to values like those shown below (suggested guidelines). Use the HP-UX System Administration Manager (SAM) tool to make these changes.

```
nproc 4096 - 4096
ksi_alloc_max 32768 - (NPROC*8)
max_thread_proc 4096 – 4096
maxdsiz 0x90000000 - 0X90000000
maxdsiz_64bit 2147483648 - 2147483648
maxfiles 4000 - 4000
maxssiz 401604608 - 401604608
maxssiz_64bit 1073741824 - 1073741824
maxtsiz 0x40000000 - 0X40000000
msgmap 4098 - (NPROC+2)
msgmni 4096 - (NPROC)
msgtql 4096 - (NPROC)
ncsize 35840 - (8*NPROC+2048+VX_NCSIZE)
nfile 67584 - (16*NPROC+2048)
ninode 34816 - (8*NPROC+2048)
nkthread 7184 - (((NPROC*7)/4)+16)
nproc 4096 - 4096
nsysmap 8192 - ((NPROC)>800?2*(NPROC):800)
nsysmap64 8192 - ((NPROC)>800?2*(NPROC):800)
semmni 1024 - 1024
semmns 16384 - ((NPROC*2)*2)
semmnu 2048 - 2048
semume 256 - 256
shmmax 0x40000000 Y 0X40000000
shmmni 1024 - 1024
shmseg 1024 Y 1024
vps_ceiling 64 - 64
```

Setting Permissions for the HP-UX Scheduler

This topic is part of *Tuning Siebel CRM for HP-UX*.

Siebel CRM will have better performance on HP-UX if you make the following changes, which allow the Siebel Server to execute the HP-UX scheduler upon startup. You must have root privileges to make these changes.

To set permissions for the HP-UX scheduler

1. Add the following line to the `/etc/privgroup` file, creating it if necessary:

   ```
   -g RTSCHED
   ```

2. Save the file and exit.
3. Execute the following command:

   ```
   setprivgrp -f /etc/privgroup
   ```

4. Verify that global `RTSCHED` permissions are set by executing the following command:

   ```
   getprivgrp
   ```

   If the command is successful, then the system will respond:

   ```
   global privileges: RTSCHED
   ```

Tuning Siebel CRM for Oracle Solaris

This topic provides instructions for configuring and tuning operating system settings and Siebel Enterprise Server components so that you can run Siebel applications on Oracle Solaris. It contains the following information:

- *Tuning Kernel Settings for Oracle Solaris*
- *Tuning Siebel Application Object Manager Instances for Oracle Solaris*

Tuning Kernel Settings for Oracle Solaris

This topic is part of *Tuning Siebel CRM for Oracle Solaris*.

To run Siebel Servers or Siebel Application Interface in an Oracle Solaris environment, you need to set Oracle Solaris kernel parameters to specific recommended values for particular releases of Oracle Solaris servers. To learn the specific parameter recommendations for Siebel Servers or Siebel Application Interface running on Oracle Solaris, contact Oracle Advanced Customer Services. Contact your Oracle sales representative to request assistance from Oracle Advanced Customer Services.

Several Oracle Solaris kernel parameter settings significantly affect performance of Siebel CRM in general, and the Siebel Server in particular.

Oracle Solaris kernel parameters reside in the configuration file `/etc/system`. To change the settings for these parameters, you must manually edit this file, save your changes, and reboot the system.

Normally, the Oracle Solaris kernel memory parameter settings are relatively low. However, for large memory-model applications like the Siebel Server applications, it is recommended that you increase the values assigned to several of these parameters.

> **CAUTION:** If you use the default Oracle Solaris kernel parameters, or lower, to run a Siebel Server in an Oracle Solaris environment, then there is a risk of serious performance problems, resulting in SIGABRT or SIGSEV errors for some Siebel Server components.

To tune the Oracle Solaris kernel settings for Siebel Server

1. Using an editor such as `vi`, open the `/etc/system` file for editing.
2. Add or modify the following lines, which are general settings:

```
set rlim_fd_cur = 65536
set kernel_cage_enable = 1
set rlim_fd_max = 65536
```

3. Save your changes and exit the editor.
4. Restart the server computer to have the new settings take effect.

Tuning Siebel Application Object Manager Instances for Oracle Solaris

This topic is part of *Tuning Siebel CRM for Oracle Solaris*.

Oracle Solaris computers running more than 50 Application Object Manager instances (multithreaded processes for Siebel Application Object Manager) might experience a situation where one or more of the processes do not start correctly, while the rest start and function normally. The log files for the processes that do not start will indicate that they have not started correctly. If you experience these symptoms, then change the `tcp_conn_req_max_q` and `tcp_conn_req_max_q0` default values, using the `ndd` command.

To change TCP values

1. Log in as `root`.
2. Issue the `ndd` command:

   Note: The responses are noted in bold.

```
ndd /dev/tcp
name to get/set ? tcp_conn_req_max_q
value ? 1024
name to get/set ? tcp_conn_req_max_q0
value? 4096
```

3. Add the following lines to the `/etc/system` file, using any text editor such as vi:

```
set tcp:tcp_conn_req_max_q = 1024
set tcp:tcp_conn_req_max_q0 = 4096
```

4. Save your changes and exit the editor.

Tuning Siebel CRM for IBM DB2

This topic provides instructions for tuning your Siebel database on IBM DB2. These steps improve the performance of Siebel workspace queries on IBM DB2, for Siebel developers using Siebel Tools or Siebel Web Tools. Your database administrator can apply these configurations directly on your DB2 database. Testing for the following configurations was done on IBM DB2 v11.1. Results might vary on IBM DB2 v10.5.

To tune Siebel CRM for IBM DB2

1. Check the value of the NUMDB parameter. If it is too high, then set it to a lower number to provide each database with sufficient memory.
2. Increase the buffer pool allocation to 5 GB. To do so, enter the following commands:

   > **Note:** Before you increase buffer pool allocation, it is important to identify the buffer pool for the table spaces used by your workspace queries. For example, for table spaces TBS_16K and TBS_4K, you might configure the buffer pools BUF_16K and BUF_4K, respectively.

   ```
   db2 alter bufferpool buf_4k immediate size 1310720 automatic # 5 GB
   db2 alter bufferpool buf_16k immediate size 327680 automatic # 5 GB
   ```

   The value AUTOMATIC allows the size to be adjusted dynamically if more memory is available. Alternatively, each buffer pool could also be set to 80% of DATABASE_MEMORY.

   > **Note:** The preceding configuration changes would require the database server to be restarted, requiring some downtime. For maximum performance benefit, perform steps 1 and 2 together.

3. Enable parallel execution. To do so, enter the following commands:

   ```
   db2stop force
   db2 update dbm cfg using intra_parallel on
   db2start
   db2 deactivate db <dbname>
   db2 update db cfg for <dbname> using dft_degree any
   db2 activate db <dbname>
   ```

4. Increase SORTHEAP size. To do so, enter the following commands:

   ```
   db2 update db cfg for <dbname> using SHEAPTHRES_SHR 200000
   db2 update db cfg for <dbname> using SORTHEAP 100000
   ```

Related Books

For additional database configuration settings for IBM DB2 and other RDBMS, see *Siebel Installation Guide* .

14 Monitoring Siebel Application Performance with Siebel ARM

Monitoring Siebel Application Performance with Siebel ARM

This chapter provides an overview of performance monitoring using the Siebel Application Response Measurement (Siebel ARM) feature. It contains the following topics:

- *About Siebel Application Response Measurement*
- *About Siebel ARM Parameters and Variables*
- *Enabling and Configuring Siebel ARM*
- *Guidelines for Converting Siebel ARM File*

For information about how to analyze the data that the Siebel ARM feature collects, see *Analyzing Siebel ARM Data*.

About Siebel Application Response Measurement

Siebel ARM is a framework for capturing critical performance data in Siebel CRM. This data is saved in binary file format. For information about how to analyze the data in these files, see *Analyzing Siebel ARM Data*

Siebel ARM captures response times at key monitoring points within the Siebel Server infrastructure. These Siebel ARM monitoring points are classified in the following distinct areas within the Siebel infrastructure:

- **Infra-Network Time.** Time duration between a request from the Siebel Application Interface and the Siebel Server (including the network time).
- **Siebel Server Time.** Time duration for the request to be processed by the Siebel Server and the Siebel database server (time between Server Thread (SMI) and any database-layer calls).
- **Database Time.** Time for any Siebel database-layer calls.
- **Application-Specific Time.** Time duration spent in application-specific areas of the infrastructure.

The Siebel ARM feature monitors system performance in the infrastructure and application-specific areas in the following list. The areas in the following are listed as they appear in Siebel ARM output. The name in parentheses after the area name represents the area symbol, which also appears in Siebel ARM output. In this table, MWC represents the Mobile Web Client.

Area Monitored by Siebel ARM	Area Monitored by Siebel ARM
SARM Framework (SARM)	Assignment Manager (AM)
Web Engine (SWE)	Fulfillment Engine (FSFULFILL)

Area Monitored by Siebel ARM	Area Monitored by Siebel ARM
Build Web Page (SWEPAGE)	Preventative Maintenance Engine (FSPREVMNT)
Universal Inbox (UINBOX)	Siebel Loyalty (LOY)
Database Connector (DBC)	Handheld Sync (HHSYNC)
Application Server (INFRA)	SmartScript (SMARTSCRIPT)
Workflow (WORKFLOW)	Siebel Anywhere (SIEBANYWHERE)
eScripts (SCRIPT)	Communications Channel Manager (CSMM)
Request Broker (SRB)	Communications Server Service (CSS)
File System Manager (FSM)	Customer/Order Management - Configurator (COMCFG)
Business Service (BUSSRVC)	EAI Transports (EAITRANSP)
Email Response (EMR)	MWC Profiler (MWC)
Security / Authentication (SEC)	Communications Outbound Manager (COM)
Object Manager (OBJMGR)	Siebel Repository (SRF)

Each area listed contains one or more subareas, which further define the timing and performance of their respective area. The number of areas and subareas present in Siebel ARM files depends on the granularity level, which is configured by the parameter `SARM Granularity Level`. For more information about this parameter, see *About Siebel ARM Parameters and Variables*. For more information about the format of Siebel ARM files, see *About Siebel ARM Files*.

About Siebel ARM Parameters and Variables

The following parameters on the Siebel Server enable and configure the Siebel ARM feature. The Siebel ARM parameters and environment variables are equivalent in function and similar in naming convention.

See the following table for a listing of each Siebel ARM parameter and its equivalent environment variable. Descriptions of each parameter and environment variable follow the table. For more information about enabling Siebel ARM using these parameters and environment variables, see *Enabling and Configuring Siebel ARM*.

Parameter Display Name	Parameter Alias	Environment Variable Name
SARM Granularity Level	SARMLevel	SIEBEL_SARMLevel

Parameter Display Name	Parameter Alias	Environment Variable Name
SARM Buffer Size	SARMBufferSize	SIEBEL_SARMBufferSize
SARM Period	SARMPeriod	SIEBEL_SARMPeriod
SARM Max Number of Files	SARMMaxFiles	SIEBEL_SARMMaxFiles
SARM Data File Size	SARMFileSize	SIEBEL_SARMFileSize

SARM Granularity Level

This parameter or equivalent environment variable specifies the amount of response measurement detail logged to Siebel ARM files, and effectively enables or disables the Siebel ARM feature. `SARM Granularity Level` has the following settings:

- **0 (OFF).** This setting is the default value and disables Siebel ARM.
- **1 (ARM).** This setting captures general application performance and is based on the application response measurement (ARM) standard. At this level, Siebel ARM collects information such as process and component boundaries, third-party software calls, database measurements, workflow execution, and script performance. Use this level for general performance monitoring.
- **2 (Detail).** This setting captures the information at level 1 as well as detailed information such as steps of workflow execution, construction of large objects, reading of large files, and crossing significant architectural areas. Use this level for problem diagnostics.

SARM Buffer Size

The Siebel ARM framework uses a buffered data generation mechanism. Siebel ARM collects data and stores it in memory. After the in-memory data size reaches a threshold defined by the `SARM Buffer Size` parameter or equivalent environment variable, Siebel ARM outputs the stored data to file on a physical disk. `SARM Buffer Size` is specified in bytes. The default value is 5000000 (5,000,000 bytes, approximately 5 MB). Valid settings correspond to values ranging from 100,000 bytes to 50,000,000 bytes.

> **Note:** Siebel ARM also outputs stored data to file based on elapsed time, which is defined by the `SARM Period` parameter or environment variable. The `SARM Period` setting can determine the size of the data saved to file rather than the threshold value defined by `SARM Buffer Size`.

For example, if the setting of `SARM Buffer Size` is 5 MB and there are five instances (processes) of the component, then the total memory used is 25 MB.

SARM Period

Siebel ARM collects data and stores it in memory. The time period specified by the `SARM Period` parameter or equivalent environment variable determines when Siebel ARM outputs the stored data to file on a physical disk regardless of the value set for `SARM Buffer Size`. The value of `SARM Period` is specified in minutes, and has a default value of 3 minutes. The valid settings range from 1 minute to 60 minutes.

SARM Max Number of Files

This parameter or equivalent environment variable specifies the maximum number of Siebel ARM files created per component instance. The default value is 4, and there is no predefined upper limit to the number of files Siebel ARM creates. (`SARM Data File Size` configures how large a file becomes before a new file is stored on the physical disk.)

The number of active Siebel ARM files per component process is the value of `SARM Max Number of Files` plus 1. Siebel ARM removes the oldest file for that process only after the file representing `SARM Max Number of Files` plus 1 reaches the value of `SARM Data File Size`. For an example of how to calculate memory usage, see *SARM Data File Size*.

SARM Data File Size

This parameter or equivalent environment variable specifies how large a file becomes before Siebel ARM stores data in a new file on the physical disk. The value of `SARM Data File Size` is specified in bytes. The default value is 15000000 (15,000,000 bytes, approximately 15 MB), and there is no predefined upper limit to file size.

Until the specified size is reached, Siebel ARM continues to append file segments to the current file. When the file limit is reached, Siebel ARM creates a new file. (`SARM Max Number of Files` configures the number of files maintained by Siebel ARM.)

When Siebel ARM reaches the file number specified by `SARM Max Number of Files` (that is, the number of files of size `SARM Data File Size` has reached the value of `SARM Max Number of Files`), Siebel ARM removes the first file (that is, the oldest file) when the next file reaches the `SARM Data File Size` limit.

Therefore, the maximum amount of disk space used is approximately `SARM Max Number of Files` plus 1 times the number of bytes represented by the value of `SARM Data File Size`. This amount of memory is per-process (per component instance).

For example, if `SARM Data File Size` is 15 MB, `SARM Max Number of Files` is 4, and there are 5 instances (processes) of the component, then the maximum amount of disk space consumed is approximately 375 MB; that is, 15 MB per file, times 5 files per process, times 5 processes (instances of the component).

Enabling and Configuring Siebel ARM

Enabling and configuring Siebel Application Response Measurement (Siebel ARM) involves setting Siebel ARM parameters on the Siebel Server, as outlined in the following procedure. By default, Siebel ARM is disabled.

Note: If any of the Siebel ARM parameters that you want to set are not visible, then make sure that the parameter `Show Advanced Objects` (alias `ShowAdvancedObjects`) is set to `TRUE` for the server component Server Manager (alias ServerMgr), or click Advanced to view the parameters in a server configuration view.

To enable and configure Siebel ARM on the Siebel Server

1. Set the parameter `SARM Granularity Level` (alias `SARMLevel`) to a value of 1 or 2 to enable Siebel ARM on the Siebel Server.

   You can enable Siebel ARM at either the enterprise, Siebel Server, or server component level. For more information about this parameter and its settings, see *About Siebel ARM Parameters and Variables*.
2. Set the other Siebel ARM-related parameters to configure the Siebel ARM file characteristics on the Siebel Server.

   You can configure Siebel ARM at the Siebel Server or server component level. For more information about these parameters, see *About Siebel ARM Parameters and Variables*.

For more information about setting Siebel Server parameters using the Server Manager GUI or command-line interface, and for background information about parameter administration, see *Siebel System Administration Guide* .

Guidelines for Converting Siebel ARM File

Review the following information as recommendations when converting Siebel ARM files.

- Set the Siebel ARM granularity to level 1 for monitoring production deployments; set the granularity to level 2 for diagnostic purposes.
- Set the `SARM Max Number of Files` parameter to 0 in order to disable Siebel ARM file creation. This scenario can be useful when you are enabling Siebel ARM for use with other third-party response measurement tools.
- Make sure that the Siebel ARM feature has flushed data to the Siebel ARM file before converting the file. The Siebel ARM feature creates an empty Siebel ARM file before data is flushed to the file. For more information about this process, see the descriptions for `SARM Data File Size` and `SARM Period` in *About Siebel ARM Parameters and Variables*.
- Change the value of `SARM Memory Size Limit` or `SARM Period` to a lower setting if the Siebel ARM files remain empty on a consistent basis. For more information about this process, see the descriptions for `SARM Data File Size` and `SARM Period` in *About Siebel ARM Parameters and Variables*.
- Make sure that the Siebel ARM file name and path name, as necessary, are correct when you reference the Siebel ARM files in the commands.
- (Siebel ARM Analyzer Tool only) If the Siebel ARM Analyzer Tool cannot convert large Siebel ARM files or the output file is too large, then split the Siebel ARM file by using the `-p` flag with the tool. For more information about the `-p` flag, see *About the Siebel ARM Analyzer Tool*.
- Concatenate Siebel ARM files to increase the amount of performance data for a given process. For example, as the Siebel ARM feature can save numerous Siebel ARM binary files for each process, concatenate these files to view performance data for multiple requests for this process. For more information about the number of files saved, see the description for `SARM Max Number of Files` in *About Siebel ARM Parameters and Variables*.

  Tip: Use a third-party utility to concatenate Siebel ARM files on Windows. Use the command `cat list_of_files > filename.sarm` to concatenate Siebel ARM files on UNIX.

> **Note:** Only concatenate Siebel ARM files of the same process.

- Gather performance analysis data on your Siebel application before customizing the application. These baseline measurements provide a good reference when monitoring the performance of your Siebel application after any customizations.
- Run a user session trace analysis if there are performance problems for an individual user during a particular session. The user trace session data identifies each request that the user made and identifies which request required the longest time when compared to a base line.
- Use the performance aggregation data to diagnose performance at a given point in time or for a certain process. Reviewing the data by group can diagnose the area that is performing poorly. After reviewing a high-level view of the performance data, extrapolate a more detailed review by running the comma-separated value analysis. For more information about running this analysis, see *Running Siebel ARM Data CSV Conversion*.
- Compile performance aggregation data over a period of time to determine a trend analysis.

15 Analyzing Siebel ARM Data

Analyzing Siebel ARM Data

This chapter describes how to analyze Siebel Application Response Measurement (Siebel ARM) data. It contains the following topics:

- *About Siebel ARM Files*
- *Analyzing Siebel ARM Files Using the Siebel ARM Query Tool*
- *Analyzing Siebel ARM Files Using the Siebel ARM Analyzer Tool*

For more information about the features of Siebel ARM, including how to collect Siebel ARM data, see *Monitoring Siebel Application Performance with Siebel ARM*.

About Siebel ARM Files

When enabled, the Siebel ARM feature saves binary Siebel ARM files in the following locations:

- Siebel Server `log` subdirectory on Windows: `SIEBSRVR_ROOT\log`
- Siebel Server `log` subdirectory on UNIX: `SIEBSRVR_ROOT/enterprises/EnterpriseServerName/SiebelServerName/log`

For information about Siebel ARM, see *About Siebel Application Response Measurement*.

Siebel ARM names the binary data files as in the following example:

```
T201707081744_P001768_N0006.sarm
```

where:

- `T` is a constant value, indicating that timing convention information follows.
- `201707081744` indicates the date and time of the Siebel ARM file. This example indicates that this file was saved on July 8th, 2017 at 17:44.
- `P` is a constant value, indicating that process ID information follows.
- `001768` indicates the process ID on which Siebel ARM collects data.
- `N` is a constant value, indicating that Siebel ARM ID information follows.
- `0006` indicates the Siebel ARM log ID number for the listed process ID. Starts at 0000 and increments until it reaches 9999, at which point it wraps around to 0000.
- `.sarm` is the Siebel ARM file extension.

To analyze the data contained in the binary Siebel ARM files, use one of the following tools, which present the collected data in a readable format:

- **Siebel ARM Query Tool.** This command-line tool allows you analyze the performance data collected by Siebel ARM. The Siebel ARM Query Tool is the recommended tool for analyzing Siebel ARM data. For more information about the Siebel ARM Query Tool, see *Analyzing Siebel ARM Files Using the Siebel ARM Query Tool*.

- **Siebel ARM Analyzer Tool.** This command-line tool allows you analyze the performance data collected by Siebel ARM. However, the Siebel ARM Query Tool is recommended for use over the Siebel ARM Analyzer Tool. For more information about the Siebel ARM Analyzer Tool, see *Analyzing Siebel ARM Files Using the Siebel ARM Analyzer Tool*.

Analyzing Siebel ARM Files Using the Siebel ARM Query Tool

This topic describes how to analyze Siebel ARM files using the Siebel ARM Query Tool. This tool converts binary Siebel ARM files into readable output for analysis.

The following topics provide more information about using the Siebel ARM Query Tool:

- *About the Siebel ARM Query Tool*
- *General Commands for the Siebel ARM Query Tool*
- *Configuring the Siebel ARM Query Tool*
- *Configuring Input for the Siebel ARM Query Tool*
- *Configuring Output from the Siebel ARM Query Tool*
- *Using Selection Filters with the Siebel ARM Query Tool*
- *Aggregating Siebel ARM Data with the Siebel ARM Query Tool*
- *Generating Histograms with the Siebel ARM Query Tool*
- *Using Macros with the Siebel ARM Query Tool*

About the Siebel ARM Query Tool

This topic is part of *Analyzing Siebel ARM Files Using the Siebel ARM Query Tool*.

The Siebel ARM Query Tool is a performance analysis command line tool that processes the binary Siebel ARM data produced by the Siebel Server. This tool provides many command-line options that allow you to create complex queries.

Comparison with Siebel ARM Analyzer Tool

The Siebel ARM Query Tool is recommended for use over the Siebel ARM Analyzer Tool, which is described in *Analyzing Siebel ARM Files Using the Siebel ARM Analyzer Tool*. The Siebel ARM Query Tool processes data faster than the Siebel ARM Analyzer Tool, allows you to create more complex queries, and offers many other benefits summarized in this topic.

Summary of Features of the Siebel ARM Query Tool

Some of the features of the Siebel ARM Query Tool are described as follows:

- **Source data specification.** You can specify individual Siebel ARM files or directories as input for the Siebel ARM Query Tool. For more information about the parameters that allow you to specify input data, see *Configuring Input for the Siebel ARM Query Tool*.
- **Multiple output formats.** You can specify that multiple types of output be produced simultaneously. Supported output formats include TXT, XML and CSV. For more information about how to specify the output format, see *Configuring Output from the Siebel ARM Query Tool*.

- **Data filtering capability.** You can specify selection filters that include or exclude data from analysis by the Siebel ARM Query Tool. Selection filters provide many ways to specify the types of records that you are interested in. Filters are generally not mutually exclusive. That is, you do not need to choose just one filter; you can use multiple filters, in any combinations that you require. For more information about using selection filters, see *Using Selection Filters with the Siebel ARM Query Tool*.
- **Aggregating data.** You can specify the order of aggregation for your output. This includes rollup calculations. Aggregation is the grouping of Siebel ARM records that share common attributes and the computing of statistics on the group. For more information about aggregation, see *Aggregating Siebel ARM Data with the Siebel ARM Query Tool*.
- **Histograms.** Histograms are a special type of aggregation. You can use a histogram to aggregate the results that the Siebel ARM Query Tool retrieves when you submit a query on Siebel ARM data that can return too many values if you do not use the histogram. For more information about using histograms, see *Generating Histograms with the Siebel ARM Query Tool*.
- **Macro language.** The Siebel ARM Query Tool supports the use of macros. For more information about using macros, see *Using Macros with the Siebel ARM Query Tool*.
- **Configurable memory for statistical accuracy.** By default, approximately 20 MB of memory is used by the Siebel ARM Query Tool's internal buffer used for statistical calculation. You can increase this amount of memory in order to increase the statistical accuracy of results. For example, increasing the amount of memory for the tool to 500 MB can reduce the statistical error level from 1% to 0.2%. For more information about configuring memory allocation for the Siebel ARM Query Tool, see *Configuring the Siebel ARM Query Tool*.

For more command-line options for the Siebel ARM Query Tool, see *General Commands for the Siebel ARM Query Tool* and *Configuring the Siebel ARM Query Tool*.

General Commands for the Siebel ARM Query Tool

This topic is part of *Analyzing Siebel ARM Files Using the Siebel ARM Query Tool*. It describes the general options for use with the Siebel ARM Query Tool.

Siebel ARM Query Tool options include commands to display online help and to display progress information to the command window. The following table describes these commands.

Flag	Description
-help	Prints help.
-copyright	Prints copyright information about the Siebel ARM Query Tool.
-tips	Prints the command-line syntax to accomplish common aggregations, reports, and conversions.
-macrosyntax	The Siebel ARM Query Tool supports a macro language. This flag prints the syntax of the macro language. For more information about using macros, see *Using Macros with the Siebel ARM Query Tool*.
-planonly	Prints an execution plan for a query, without executing the query.
-quiet	Prints results only to the output console. No progress information appears when you specify this flag.

Flag	Description
	You can also specify to save progress information to a file, as noted in *Configuring the Siebel ARM Query Tool*. The following example saves progress information to the file verbose.txt: > sarmquery –output verbose=verbose.txt –input data.sarm -aggregate time=1

Configuring the Siebel ARM Query Tool

This topic is part of *Analyzing Siebel ARM Files Using the Siebel ARM Query Tool*. It describes how you can configure the Siebel ARM Query Tool. An example of configuring the Siebel ARM Query Tool is modifying the amount of memory that the Siebel ARM query uses, to improve statistical accuracy.

The following table describes the flags that you can use to configure the Siebel ARM Query Tool. All of the listed options must be preceded by the option **`-config`**.

Flag	Description
file=*macro_file_name*	Specifies the file that contains a macro. For more information about using macros, see *Using Macros with the Siebel ARM Query Tool*.
macro=*macro_name*(*string*)	Executes the specified macro and passes the specified string as the first argument to the macro. Before you can use this flag, you must specify the file that contains the macro, by using the **`file`** flag: -config file=*macro_file_name*
gmt=0	Parse all time stamps (on the command line) and displays all time in Greenwich Mean Time (GMT) to the command window. If you do not specify this flag, then the Siebel ARM Query Tool uses local time.
gmt=[+-]*HHMM*	Parses and reports all times as an offset from GMT. Offsets are specified in the *HHMM* notation, where *HH* is the hours (0023) and *MM* is the minutes (00 to 59). For example, to report Pacific Standard Time (PST), use the **`gmt`** flag as follows: –config gmt=-0800
datalimit=*limit*	Specifies a maximum number of records to return, where *limit* is the maximum number. When the maximum number is exceeded, an error message (data limit exceeded) appears, and the Siebel ARM Query Tool terminates.
timelimit=*seconds*	Specifies the maximum number of seconds that the Siebel ARM Query Tool can execute. When the maximum number of seconds elapses, the tool exits. The Siebel ARM Query Tool processes in two phases: **File collection phase.** The tool identifies all required files. The value that you specify for the timelimit flag does not affect this phase; the time that the Siebel ARM Query Tool takes cannot be constrained. **Data processing phase.** The value that you specify for the **`timelimit`** flag affects the length of this phase.

Flag	Description
memlimit=*memory*	Specifies the amount of memory, in megabytes, that the Siebel ARM Query Tool can allocate to an internal buffer which the tool uses to compute aggregated statistics. The more memory that the internal buffer can use, the more accurate are the statistics returned. The default memory limit is 20 MB. You can allocate a value between 5 MB and 500 MB. The value that you specify does not affect performance; it only applies to the buffer that the Siebel ARM query uses for statistical calculation.

Configuring Input for the Siebel ARM Query Tool

This topic is part of *Analyzing Siebel ARM Files Using the Siebel ARM Query Tool*. It describes how you can specify the Siebel ARM file(s) that the Siebel ARM Query Tool uses. The Siebel ARM Query Tool converts binary Siebel ARM files into readable output for analysis.

The following table describes the available input options. All of the listed options must be preceded by the option **-input**.

Flag	Description
sarmfile	Specifies a binary Siebel ARM file (.sarm).
directory	Specifies a directory that contains Siebel ARM files. The Siebel ARM Query Tool processes all of the Siebel ARM files that it finds in the specified directory.
stdin	A literal keyword that tells Siebel ARM query to read a list of Siebel ARM file names from standard input. You specify one Siebel ARM file or a directory name per line. The following are examples of valid input using **stdin** or other input flags: > sarmquery . > sarmquery –input d:\sarmdata > sarmquery d:\sarmdata\srvr1 d:\sarmdata\srvr2 sarmfile1.data > dir /s /b *.sarm | sarmquery –input stdin

Configuring Output from the Siebel ARM Query Tool

This topic is part of *Analyzing Siebel ARM Files Using the Siebel ARM Query Tool*.

This topic discusses the output options that are available to you when you use the Siebel ARM Query Tool. The following table describes the flags that you can use to configure output.

The Siebel ARM Query Tool can simultaneously write output to files in the TXT, XML and CSV formats. The following example shows using the Siebel ARM Query Tool to write different types of information to different files.

```
> sarmquery -input d:\sarmdata -select subarea=infra_entry -select resp=1000
-output header=hdr.csv -output sarm=sarm.csv -aggregate area -output agg=agg.xml
```

The preceding command writes:

- Information about the header metadata from the Siebel ARM files to the file hdr.csv.
- Siebel ARM data to the file sarm.csv.
- Aggregate information about the area to the file agg.xml.

You can also specify a maximum number of lines to write to a file. When the file contains the maximum number of lines specified, the Siebel ARM Query Tool creates a new file with the same filename plus *N*, where *N* equals a number. The following example illustrates the use of this option:

```
> sarmquery -output sarm=sarmdata.csv#20000
```

This command writes Siebel ARM data in comma-separated values (CSV) format to the file named sarmdata.csv. The optional value #20000 writes a maximum of 20,000 lines to sarmdata.csv. Once sarmdata.csv contains 20,000 lines, the Siebel ARM Query Tool writes to a new file with the name sarmdata_0002.csv, and so on, until all of the applicable data is output.

All flags in the following table must be preceded by the option `-output`.

Flag	Description
fdr=*filename*	Converts all FDR files (specified using `-input`) to CSV format and write them to the specified file. **Note:** For more information about FDR (flight data recorder) files, see *Siebel System Monitoring and Diagnostics Guide* .
fdrhdr=*filename*	Converts all of the FDR headers to CSV and write them to the specified file.
error=*filename*	By default, the Siebel ARM Query Tool writes output to the file stderr.txt. This command redirects error messages to the specified file.
debug=*filename*	By default, Siebel ARM query does not write debug messages. This command enables the Siebel ARM Query Tool to write debug messages to the file specified file.
dbglines=*filename*	This command enables the Siebel ARM Query Tool to write more debug information.
verbose=*filename*	By default, the Siebel ARM Query Tool writes to the file stderr.txt. This command redirects verbose output to the specified file.
map=*filename*	Creates a file which lists all areas and subareas for Siebel ARM.

header=*filename*
 	
appheader=*filename*
 	
cliheader=*filename*
	The available flags are:
	
	`header`
	
	Writes all header information to the specified file.
	
	`appheader`
	
	Writes header information from Siebel ARM files generated by the Siebel Server.

Flag	Description
	cliheader Writes header information from the Siebel ARM files generated by clients. The following example writes information about all headers to all.csv and then writes information for each specific part to a different file: > sarmquery –input dir –output header=all.csv –output appheader=app.csv –output cliheader=cli.csv
sarm=*filename* appsarm=*filename* clisarm=*filename*	The available flags are: **sarm** Writes all Siebel ARM data to the specified file. **appsarm** Writes all Siebel ARM data generated by the Siebel Server to the specified file. **clisarm** Writes all Siebel ARM data generated by clients. The following example writes Siebel ARM data generated by the Siebel Server: > sarmquery –output appsarm=app.csv
agg=*filename* iagg=*filename* sagg=*filename*	The available flags are: **agg** Writes the aggregation report to the specified file. **sagg** Writes a subset of the aggregation report to the specified file. This flag specifies exclusive metrics. **iagg** Writes a subset of the aggregation report to the specified file. This flag specifies inclusive metrics. To use the options, you must have aggregated Siebel ARM data by using the flag **-aggregate**.
avginclresp=*filename* pctcount=*filename* pctinclresp=*filename* pctselftime=*filename*	The available flags are: **avginclresp** Writes the average response time to the specified file. **pctcount** Writes the percentage of server requests that complete within the specified time to the specified file. The following example writes the percentages of server requests that complete within 100 milliseconds, between 100 milliseconds to 500 milliseconds, and so on: > sarmquery –histogram resp=100,500,1000,2000,5000 – out pctcount=stdout.txt

Flag	Description
	pctselftime Writes the percentage of time spent in internal parts of the Siebel Server. This is also known as SLA fingerprint.

Using Selection Filters with the Siebel ARM Query Tool

This topic is part of *Analyzing Siebel ARM Files Using the Siebel ARM Query Tool*. It describes how to use selection filters with the Siebel ARM Query Tool to constrain the data that the tool retrieves.

Siebel ARM files contain one or more segments. Each segment has a header and a body section. The header section contains metadata describing the data contained in the body section. For example, the metadata describes the time range when the data in the body section was collected and the amount of data in the body section.

You can use selection filters to filter the metadata and the actual performance data.

- The following table describes the filters that you can use for the metadata in Siebel ARM files
- The following table describes the filters that you can use for data in Siebel ARM files

Note the following points when you formulate queries:

- String filter searches are case insensitive and accept wild cards. For example, the following query retrieves Siebel ARM records associated with the users *johndoe*, *JOHNDOE*, and *JoHnDoE*:

  ```
  > sarmquery -select user=JohnDoe
  ```

- String filters also accept leading and trailing wild cards. For example, each of the following queries retrieves records associated with the same users, and possibly others:

  ```
  > sarmquery -select user="*doe"
  > sarmquery -select user="john*"
  > sarmquery -select user="*hndo*"
  ```

  > **Note:** Wild cards can only be leading or trailing. Wild cards in the middle of a pattern do not retrieve results. For example, the pattern `jo*ndoe` does not retrieve *johndoe*.

- You can combine selection filters to retrieve records that match multiple conditions. For example, the following selection filter retrieves all script execution records that executed for at least 5 seconds:

  ```
  > sarmquery -select area=script -select resp=5000
  ```

All of the options listed in the following table must be preceded by the option `-select`.

Flag	Description
component=*componentname*	Selects headers from Siebel ARM files that were generated by the named component.
fillfactor=*percent*	Use this flag to specify a percentage value; the Siebel ARM Query Tool retrieves headers that are *percent* full.

Flag	Description
Or: fillfactor=*minpct*,*maxpct*	Alternatively, you can specify two arguments, a minimum percentage *minpct* and a maximum percentage *maxpct* to retrieve a range of values. Generally, headers have a capacity to hold a certain number of Siebel ARM records. The percentage value is the ratio of actual number of records to that capacity.
host=*hostname*	Selects headers from Siebel ARM files generated on the named host.
procid=*integer*	Selects headers from Siebel ARM files generated by a process whose ID (process ID) is *inter*.
segcapacity=*nrecs* Or: segcapacity=*min*,*max*	Selects headers whose capacity to hold Siebel ARM records matches the number of records specified by *nrecs*. Alternatively, you can specify two arguments, a minimum number of records and a maximum number of records, to retrieve a range of values.
segduration=*nsecs* Or: segduration=min,*max*	Selects headers whose duration matches the number of seconds specified. A header duration is the number of seconds the file segment was in memory before it was flushed. Alternatively, you can specify two arguments, *min* and *max*, to retrieve a range of values.
segid=min Or: segid=min,*max*	Specify a single argument to retrieve the headers with an internal segment ID of at least *min*. Alternatively, you can specify two arguments, *min* and *max*, to retrieve a range of values.
segsize=*size* Or: segsize=min,*max*	A segment size is the size of the header and body in bytes. Specify a single argument of *size* to retrieve all headers whose segment size is *size* bytes. Alternatively, you can specify two arguments, *min* and *max*, to retrieve a range of values.
server=*servername*	Retrieve headers from Siebel ARM files generated on the specified Siebel Server *servername*.
sourcetype=*value*	Retrieves headers from Siebel ARM files generated by the specified type of server or process. Specify one of the following parameters in place of *value* to retrieve the headers: **`app`** Retrieves headers generated by Siebel Servers. **`cli`** Retrieves headers generated by other client programs such as the Siebel Mobile Web Client.
threshold=min Or: threshold=min,*max*	A threshold is a value represented in milliseconds. In the Siebel ARM framework, any performance record whose total response time duration is less that the threshold amount is discarded. The threshold setting at the time the Siebel ARM file was generated is saved in the header. Specify a single argument to retrieve all headers that contain Siebel ARM records whose threshold was least *min* milliseconds.

Flag	Description
	Alternatively, you can specify two arguments, *min* and *max*, to retrieve a range of values.
starttime=*start*	Specify a start time to retrieve headers that contain Siebel ARM records that ended after the specified start time. **Note:** Time filters are compared against the generation start time for Siebel ARM records. You can specify the start time in the following ways: String in the format *YYYY-MM-DD hh:mm:ss*. For example: > sarmquery -select starttime="2014-02-13 17:05:00" A number interpreted in Universal Time Coordinated (UTC). For example: sarmquery -select starttime=1108083900 A negative number to indicate the number of seconds from the current time. For example, this command retrieves headers that contain data generated less than 300 seconds ago: > sarmquery -select starttime=-300
endtime=*end*	Specify an end time to retrieve headers that contain Siebel ARM records that were generated before the specified time *end*. **Note:** Time filters are compared against the generation end time for Siebel ARM records. You can specify the end time in the following ways: String in the format *YYYY-MM-DD hh:mm:ss*. For example: > sarmquery -select endtime="2014-02-13 17:05:00" A number interpreted in Universal Time Coordinated (UTC). For example: > sarmquery -select endtime=1108083900 A negative number to indicate the number of seconds from the current time. For example, this command retrieves headers that contain data generated less than 300 seconds ago: > sarmquery -select endtime=-300 A positive number to indicate the number of seconds after the start time. For example, this command retrieves headers that contain data generated more than 600 seconds after the start time: > sarmquery -select starttime=+600

The following table describes the filter options that you can use when you formulate a query to retrieve Siebel ARM data. All of the options must be preceded by the option `-select`.

Flag	Description
foreign	Use this flag to retrieve records whose parent records are from a different process. This is frequently the case for Siebel ARM data generated from batch mode components.

Flag	Description
orphan	Select records that are not root records and whose parent record was not found in the input Siebel ARM file(s).
level=*level* Or: level=min,*max*	Specify one argument to retrieve records whose Application Program Interface (API) level is at least equal to *level*. Alternatively, specify two arguments to retrieve a range of values between the minimum and maximum level specified. The API level indicates the importance of a Siebel ARM record as follows: *level* = 1 Equivalent to **`SARMLevel=1`** and indicates high level information. *level* = 2 Equivalent to **`SARMLevel=2`** and indicates detailed performance information. *level* = 3 Equivalent to **`SARMLevel=3`** and indicates debug or internal performance information.
parentid=*ID*	Selects Siebel ARM records whose parent ID is *ID*.
rootid=*ID*	Selects Siebel ARM records whose root ID (also known as request ID in interactive mode components) is *ID*.
sarmid=*sarmid* Or: sarmid=*procid.sarmid* Or: sarmid=*segid.procid.sarmid*	Retrieves Siebel ARM records primary ID where: *sarmid* is a number uniquely identifying this performance record within a process or component, *procid* is the process ID (same as **`-select procid=`** procid), and *segid* is the segment ID (same as **`-select segid=`** segid **`,`** segid)
sessionid=*sessid*	All session based component performance data contain an attribute known as the session ID. All performance data from all processes that have the same session ID are assumed to belong to a single session of a single user. The Siebel ARM Query Tool does a case-insensitive search using *sessid*. Wild cards are acceptable.
taskid=*taskid*	Selects all records that associated with task *taskid*.
threadid=*threadid*	Selects all records that were created by the thread whose operating system thread ID is *threadid*.
user=*username*	Selects all records that were created by user named *username*, which is typically the login name. The search is case-insensitive and wild cards are acceptable.
area=*area* area=*areacode* subarea=*sub*	Two filters, **`area`** and **`subarea`**, identify each Siebel ARM record. The area and subarea are logical sections of the Siebel Server. For example, the Siebel Web Engine (SWE) area creates Web pages. Siebel ARM records associated with the SWE describe Web page creation performance. Similarly, the database connector (DBC) area represents the connection to the

Flag	Description
subarea=*subcode* pararea=*parea* pararea=*pareacode* parsubarea=*psarea* parsubarea=*pscode*	enterprise database. Siebel ARM records associated with DBC indicate the performance of database queries. For example, the following command retrieves all Siebel ARM records associated with database queries: > sarmquery –select area=DBC You can retrieve a complete list of areas, subareas, and descriptions from the Siebel ARM Query Tool. For example, the following command saves the complete list to the file map.csv: > sarmquery –output map=map.csv If you know the numeric area or subarea codes, then you can use them directly in this command. Otherwise, you can use the string form of the symbol. When using the string form, you do not need to use the entire text string. You can use a partial text string, provided that it uniquely identifies the area or subarea. The filters **`pararea`** and **`parsubarea`** are similar to area and subarea except that they select Siebel ARM records whose parent area and subarea (respectively) are *parea* and *psarea*. For example, the following command retrieves all Siebel ARM records whose parent area is SWE: > sarmquery –select pararea=swe
children=0	Use this flag to retrieve Siebel ARM records that have no child records.
children=*count* Or: children=*min*,*max*	Specify one argument to retrieve Siebel ARM records that have child records equal to at least *count*. Specify two arguments to retrieve the records that have child records between *min* and *max*.
cputime=*ms* Or: cputime=min,*max*	Specify one argument to retrieve Siebel ARM records that spent at least *ms* milliseconds on the CPU. Specify two arguments to retrieve the Siebel ARM records that spent between *min* and *max* milliseconds on the CPU.
depth=*depth* Or: depth=*min*,*max*	Specify one integer argument to retrieve Siebel ARM records that are no more than *depth* from the root node. Specify two arguments to retrieve a range of records.
descendants=*count* Or: descendants=*min*,*max*	Specify one integer argument to retrieve Siebel ARM records that have at least *count* descendants. Specify two arguments to retrieve a range of values.
instance=*string* detail=*string* int1=*int*	Use these filters to retrieve Siebel ARM records where the filter is as follows: **`instance`** Retrieves Siebel ARM records where instance metadata equals *string*. Instance metadata typically contains the names of things, such as the view names, screen names, user name, workflow name and script name.

Flag	Description
	`detail` Retrieves Siebel ARM records where the detail metadata equals *string*. Detail metadata typically contains identification information. For example, it might contain a database row ID or the name of a business component method. **`int1`** Retrieves Siebel ARM records where int1 filter equals *int*. The **`int1`** metadata typically contains counter values, task IDs or other unspecified information. **Note:** The values that the preceding metadata fields reference depend on the associated area or subarea.
memusage=*excl* Or: memusage=min,*max*	Specify one argument to retrieve all Siebel ARM records where memory allocated or deallocated was larger than *excl* bytes. Specify two arguments to retrieve a range of Siebel ARM records that record the memory allocation or deallocation events within the specified range. For example, the following command retrieves all Siebel ARM records where the associated event recorded a memory allocation of 1 MB (or larger) or a deallocation of 1 MB or larger > sarmquery –select memusage=1000000
pctcpu=*percent* Or: pctcpu=min,*max*	Specify a single argument to retrieve Siebel ARM records that indicate a CPU consumption of the percentage *percent*. Specify two arguments to retrieve a range between *min* and *max* percent.
resptime=*ms* Or: resptime=min,*max*	Specify one argument to retrieve Siebel ARM records where the inclusive wall clock time consumed is *ms* milliseconds. Specify two arguments to retrieve a range between *min* and *max*. An inclusive wall clock time is the time spent in a specific part of the architecture and includes the time spent in all of its descendants. For example, the response time of a SWE area would be the time spent in constructing a Web page, including the time to evaluate business component events and database access.
selftime=ms Or: selftime=min,*max*	With one argument, select Siebel ARM records where the exclusive time consumed is *ms* milliseconds. With two arguments, *min* and *max* specify a range. An exclusive time is time spent in an area, and only in that area, excluding the time spent in its descendants. Hence, the self time of a SCRIPT area would be the time spent in the scripting engine, and not including time spent in the database, object manager, workflow, and so on.
starttime=*start* Or: endtime=*end*	The semantics and syntax are similar to the time filters described for headers, except that the filters select Siebel ARM records based on the time stamps.
tree=all parent=parent	These are a specialized set of selection filters that do not just operate on a single record basis but on record sets. In most cases, they serve to replace a performance record with an entire set.

Flag	Description
tree=ancestor tree=children tree=descendent	In order to understand these operators, it is important to realize that Siebel ARM records form a tree of an execution thread with parent and children nodes: **`tree=all`** Replaces a selected record by all records in its tree. **`tree=parent`** Replaces a selected record by its parent. The selected record is not included. **`tree=ancestor`** Replaces a selected record by all Siebel ARM records leading from the selected record to the root of the tree. The selected record is also included in the set. **`tree=children`** Replaces a selected record by its immediate children. The selected record is not included. **`tree=descendent`** Replaces a selected record by all of its descendents. The selected record is also included in the set.

Aggregating Siebel ARM Data with the Siebel ARM Query Tool

This topic is part of *Analyzing Siebel ARM Files Using the Siebel ARM Query Tool*. It describes the aggregation options available to you when you use the Siebel ARM Query Tool. Aggregation is the grouping of Siebel ARM records that share a common attribute and the computing statistics on the group.

Multiple **`-aggregate`** options can be specified, which are interpreted as sub-aggregations.

The Siebel ARM Query Tool can compute the total, maximum, minimum, average, and the count of contributing records for the following items:

- Inclusive Response Time
- Exclusive Response Time (self time)
- Inclusive CPU Time
- Exclusive CPU Time
- Inclusive Memory Usage
- Exclusive Memory Usage

Note: Aggregation is computed on Siebel ARM records that are retrieved by the selection filters.

All of the options listed in the following table must be preceded by the option **`-aggregate`**.

Flag	Description
area	Siebel ARM records that have the same value for **`area`** are grouped together.

Flag	Description
subarea	Siebel ARM records that have the same value for **subarea** are grouped together. Note that it is usually more common to also group by area when grouping by subarea, or filter on the area, as with the following examples: > sarmquery -aggregate area -aggregate subarea > sarmquery –select area=DBC -aggregate subarea
instance	Siebel ARM records that have the same value of instance metadata are grouped together. Instance metadata typically contains names of things such as scripts, workflows, and views. This means that the value for the instance metadata depends on the area or subarea. As a result, aggregate instance must either be preceded by an aggregation of area or subarea or a filter on area or subarea, as with the following example: > sarmquery -aggregate area -aggregate instance > sarmquery –select area=script -aggregate instance
server component host procid	Aggregates Siebel ARM records that have the same value for: Server Component Host Procid
user	Aggregates Siebel ARM records that have the same value for user name. User name is case sensitive.
sessionid	Aggregates Siebel ARM records that have the same value for session ID. **Tip:** This flag can be useful when used in conjunction with the flag for **user**, as in the following examples: > sarmquery -aggregate user -aggregate sessionid > sarmquery –select user=andy -aggregate sessionid
clickid	Aggregates Siebel ARM records that have the same value for **clickid**. Typically in an interactive component, a user has multiple sessions and a session has multiple requests. Sometimes a single user action on the client (browser) results in multiple requests. In such cases, each of those requests is associated with the same unique ID, known as the clickid. This ID is generated and associated with requests even if a user action results in a single request. **Tip:** Use this flag in conjunction with **user** and **sessionid**, as in the following examples: > sarmquery -aggregate user -aggregate sessionid -aggregate clickid > sarmquery –select user=andy -aggregate sessionid -aggregate clickid
time=*interval*	Aggregates Siebel ARM records by their time stamps over the interval specified by *interval*, where *interval* is a value in minutes.

Flag	Description
	For example, if you specify 5, then Siebel ARM records with time stamps of 12:00 to 12:05 form one aggregation, those with time stamps of 12:05 to 12:10 form another aggregation, and so on. The following example command displays a report to the command window that shows the average response time of the Siebel Server over intervals of 15 minutes: > sarmquery –select source=app –select subarea=infra_entry -aggregate time=15

Generating Histograms with the Siebel ARM Query Tool

This topic is part of *Analyzing Siebel ARM Files Using the Siebel ARM Query Tool*.

Histograms are a special type of aggregation. You can use a histogram to aggregate the results that the Siebel ARM Query Tool retrieves when you submit a query on Siebel ARM data that might return too many values if you do not use the histogram.

For example, note the following query:

```
> sarmquery -aggregate resptime
```

This query might retrieve millions of rows as the first row returns the requests that completed in one millisecond, the second row returns the number of requests that completed in two milliseconds, and so on.

A more effective query generates a histogram as in the following example:

```
> sarmquery –histogram resptime=100,200,300,400,500
```

This query returns six rows. The first row aggregates the number of requests that complete in less than 100 milliseconds, the second row aggregates the number of requests that completed between 100 and 200 milliseconds, and so on.

In the preceding example, the values 100, 200, 300 and so on are known as *bin endpoints*.

The following table describes the flag options that you can use with the histogram parameter. All of the listed options must be preceded by the option `-histogram`.

Flag	Description
resptime=*val1*,*val2* … *valN*	Creates a histogram of response (inclusive) times, where the arguments *val1* and *val2* specify the bin endpoints. *val1* and *val2* are expressed in milliseconds.
selftime=val1,val2 … *valN*	Creates a histogram of self time (exclusive) times, where *val1* and *valN* specify the endpoints. The values are expressed in milliseconds.
pctcpu=val1,val2 … *valN*	Creates a histogram of the percentage of CPU time consumed. The arguments are expressed as integers between 0 and 100.
children=val1,val2 … *valN*	Creates a histogram of the width of the execution trees.
depth=val1,val2 … *valN*	Creates a histogram of the depth of the execution trees.

Flag	Description

Using Macros with the Siebel ARM Query Tool

This topic is part of *Analyzing Siebel ARM Files Using the Siebel ARM Query Tool*.

The Siebel ARM Query Tool supports the use of macros. A macro allows you to save complex queries to a file, which you can subsequently use as input for the Siebel ARM Query Tool.

To create a macro, you need to:

- Write the syntax of your query in a CFG file (for example, macro.cfg).

  Each part of the query must appear on a separate line in the CFG file.
- Define a name for the macro, which you insert before the first line of your query in the CFG file.

For example, the following query, which retrieves data on the average login time for users, can be input as one entry from the command line:

```
> sarmquery -input d:\sarmdata -select source=app -select subarea=objmgr_sess_
relogin -select tree=all -select depth=1 -aggregate user -output
avginclresp=stdout.txt
```

However, in the CFG file the preceding query is assigned the macro name `[Login]` and appears as follows:

```
[Login]
-input d:\sarmdata
-select source=app
-select subarea=objmgr_sess_relogin
-select tree=all
-select depth=1
-aggregate user
-output avginclresp=stdout.txt
```

For the Siebel ARM Query Tool to use a macro, you need to specify two parameters:

- The location of the CFG file that contains the macro

  > **Note:** A CFG file can contain multiple macros.

- The name of the macro to execute

The following example requests that the Siebel ARM Query Tool execute the Login macro, which is saved in the macro.cfg file.

```
> sarmquery -config file=macro.cfg -config macro=Login
```

The Siebel ARM Query Tool reads the input arguments for the macro as if you had specified the individual parts of the macro on the command line. This means that you can specify additional arguments that might not be in your macro. The following example executes the Login macro and in addition specifies a time slice:

```
> sarmquery -config file=macro.cfg -config macro=Login -select
starttime="2017-06-10 10:00:00" -sel endtime="2017-06-11 09:59:59"
```

For more information about configuration commands, see *Configuring the Siebel ARM Query Tool*.

For a complete description of the macro language syntax and other options, execute the following command from the command line:

```
> sarmquery -macrosyntax
```

Analyzing Siebel ARM Files Using the Siebel ARM Analyzer Tool

This topic describes how to analyze Siebel ARM files using the Siebel ARM Analyzer Tool. This tool converts binary Siebel ARM files into readable output for analysis.

> **Note:** The Siebel ARM Query Tool, which is described in *Analyzing Siebel ARM Files Using the Siebel ARM Query Tool*, is recommended for use over the Siebel ARM Analyzer Tool.

The following topics provide general information about the Siebel ARM Analyzer Tool and describe how to generate different types of analysis output using the Siebel ARM Analyzer Tool:

- *About the Siebel ARM Analyzer Tool*
- *Running Performance Aggregation Analysis*
- *Running Call Graph Generation*
- *Running User Session Trace*
- *Running Siebel ARM Data CSV Conversion*

The following topics provide information about the different types of analysis output that the Siebel ARM Analyzer Tool generates. The analyzer tool produces output in either XML or CSV format, based on the type of conversion analysis. For more information, see:

- *About Call Graph Generation Analysis and Data*
- *About User Session Trace Analysis and Data*
- *About Siebel ARM to CSV Conversion Data*

About the Siebel ARM Analyzer Tool

This topic is part of *Analyzing Siebel ARM Files Using the Siebel ARM Analyzer Tool*.

The Siebel ARM Analyzer Tool parses the files created by the Siebel ARM feature and generates extensible markup language (XML) analytic results or comma-separated value (CSV) results. Run the Siebel ARM Analyzer Tool manually at the command-line. For an overview of the types of output this tool can generate, see *Analyzing Siebel ARM Files Using the Siebel ARM Analyzer Tool*.

This command-line utility resides in the `bin` subdirectory of the Siebel Server root directory as the executable program `sarmanalyzer.exe` on Microsoft Windows or `sarmanalyzer` on UNIX. The following table describes the parameters to use with this tool.

> **CAUTION:** Monitoring the Siebel application can result in large Siebel ARM files. In some cases, the Siebel ARM Analyzer Tool cannot allocate enough memory to convert extremely large binary Siebel ARM files. In this situation, use the `-p` flag with the Siebel ARM Analyzer Tool to split the Siebel ARM file into smaller files.

Flag	Description
-help	Use this flag with the Siebel ARM Analyzer Tool to list and describe the available flags.
-f	Use this flag with a Siebel ARM file argument to run a performance aggregation analysis. For more information, see *Running Performance Aggregation Analysis*.
-o	Use this flag to name the output path and file resulting from the analysis of the Siebel ARM binary file. Make sure to include the correct file extension based on the selected analysis, that is, either XML or CSV.
-d	Use this flag with the argument XML or CSV to indicate the type of output file format: extensible markup language (XML) or a comma-delimited list (CSV).
-a	Use this flag with the argument AREA or DETAILS when running a performance aggregation analysis. For more information about this analysis, see *Running Performance Aggregation Analysis*.
-i	Use this flag with a directory argument when running a user session trace analysis. For more information about this analysis, see *Running User Session Trace*.
-s	Use this optional flag to denote a start time for a user session trace. The format of the time argument is as follows: *yyyy-mm-dd hh:mm:ss*. Use this flag with the **-e** flag to create a time range.
-e	Use this optional flag to denote an end time for a user session trace. The format of the time argument is as follows: *yyyy-mm-dd hh:mm:ss*. Use this flag with the **-s** flag to create a time range.
-p	Use this optional flag to split large Siebel ARM files into smaller sizes. Use a value of 0 to 50 as the flag argument, which denotes the size, in megabytes, of the reduced files. The default value is 14 MB. The Siebel ARM Analyzer Tool uses the default value if the flag argument is 0. The split files are suffixed with **_s** *nnnn*, where *nnnn* is the split sequence number.

Running Performance Aggregation Analysis

This topic is part of *Analyzing Siebel ARM Files Using the Siebel ARM Analyzer Tool*.

Use the following procedure to obtain performance aggregation analysis output. For a description of the performance aggregation analysis and output, see *About Performance Aggregation Analysis and Data*. For more information about running the Siebel ARM Analyzer Tool, see *Analyzing Siebel ARM Files Using the Siebel ARM Analyzer Tool* and *About the Siebel ARM Analyzer Tool*.

To run a performance aggregation analysis

1. Navigate to the `bin` subdirectory within the Siebel Server root directory.
2. Run the Siebel ARM Analyzer Tool using the following command:

```
sarmanalyzer -o output_file_name.xml -a aggregate_argument -f sarm_file_
name.sarm
```

where:

output_file_name `.xml` is the name and path of the XML output file.

aggregate_argument is either AREA or DETAILS, depending on which area you want the Siebel ARM post-process tools to aggregate data from. For more information, see *About Performance Aggregation Analysis and Data*.

sarm_file_name `.sarm` is the name and path of the binary Siebel ARM file. Use a comma-delimited list to aggregate data from more than one Siebel ARM file.

3. Review the XML output in the file named *output_file_name*.xml. For more information about analyzing the performance aggregation analysis XML output, see *About Performance Aggregation Analysis and Data*.

Running Call Graph Generation

This topic is part of *Analyzing Siebel ARM Files Using the Siebel ARM Analyzer Tool*.

Use the following procedure to obtain call graph generation analysis output. For a description of the call graph generation analysis and output, see *About Call Graph Generation Analysis and Data*. For more information about running the Siebel ARM Analyzer Tool, see *Analyzing Siebel ARM Files Using the Siebel ARM Analyzer Tool*.

To run a call graph generation analysis

1. Navigate to the `bin` subdirectory within the Siebel Server root directory.
2. Run the Siebel ARM Analyzer Tool using the following command:

```
sarmanalyzer -o output_file_name.xml -d xml -f sarm_file_name.sarm
```

3. where:

   output_file_name `.xml` is the name and path of the XML output file.

   `-d xml` identifies the call graph generation analysis.

   sarm_file_name `.sarm` is the name and path of the binary Siebel ARM file.
4. Review the XML output in the file named *output_file_name*.xml. For more information about analyzing the call graph analysis XML output, see *About Call Graph Generation Analysis and Data*.

Running User Session Trace

This topic is part of *Analyzing Siebel ARM Files Using the Siebel ARM Analyzer Tool*.

Use the following procedure to obtain user session trace analysis output. Before running this analysis, manually collect Siebel ARM files and store them in a common directory. Use this directory as an argument with the Siebel ARM Analyzer Tool. For a description of the user session trace analysis and output, see *About User Session Trace Analysis and Data*. For more information about running the Siebel ARM Analyzer Tool, see *Analyzing Siebel ARM Files Using the Siebel ARM Analyzer Tool*.

Tip: To reduce the amount of data logged, use the time frame parameters (`-s` start time and `-e` end time).

To run a user session trace analysis

1. Navigate to the `bin` subdirectory within the Siebel Server root directory.
2. Run the Siebel ARM Analyzer Tool using the following command:

   ```
   sarmanalyzer -o output_file_name.xml -u user_name -i SARM_file_directory -s
   start_time -e end_time
   ```

   where:

   - output_file_name `.xml` is the name and path of the XML output file.
   - user_name is the User ID of the session you want to trace.
   - SARM_file_directory is the directory containing the Siebel ARM files of the Siebel Server.
   - start_time is an optional variable to define a start time of a time range for the user session trace. The argument format is as follows: *yyyy-mm-dd hh:mm:ss*.
   - end_time is an optional variable to define the end time of a time range for the user session trace. The argument format is as follows: *yyyy-mm-dd hh:mm:ss*.
3. Review the XML output in the file named *output_file_name*.xml. For more information about analyzing user session trace XML output, see *About User Session Trace Analysis and Data*.

Running Siebel ARM Data CSV Conversion

This topic is part of *Analyzing Siebel ARM Files Using the Siebel ARM Analyzer Tool*.

Use the following procedure to obtain a comma-separated value (CSV) analysis output. For a description of the CSV conversion analysis and output, see *About Siebel ARM to CSV Conversion Data*. For more information about running the Siebel ARM Analyzer Tool, see *Analyzing Siebel ARM Files Using the Siebel ARM Analyzer Tool*.

To run a Siebel ARM data to CSV conversion analysis

1. Navigate to the `bin` subdirectory within the Siebel Server root directory.
2. Run the Siebel ARM Analyzer Tool using one of the following commands:

   ```
   sarmanalyzer -o output_file_name.csv -d csv -f sarm_file_name.sarm
   ```

   where:

   - output_file_name `.csv` is the name and path of the CSV output file.
   - `-d csv` identifies the Siebel ARM data CSV conversion analysis.
   - sarm_file_name `.sarm` is the name and path of the binary Siebel ARM file or files.
3. Review the CSV output in the file named *output_file_name*.csv. For more information about analyzing CSV data, see *About Siebel ARM to CSV Conversion Data*.

> **Note:** Running a CSV conversion can create large output files that, in some cases, cannot be read by third-party software. Use the `-p` flag to split large Siebel ARM files. For more information about this flag, see *About the Siebel ARM Analyzer Tool*.

About Siebel ARM Analyzer Output Files

This topic is part of *Analyzing Siebel ARM Files Using the Siebel ARM Analyzer Tool*.

Running the Siebel ARM Analyzer Tool produces output files of either extensible markup language (XML) or comma-separated value (CSV) format, depending on the type of Siebel ARM file conversion. For more information about converting Siebel ARM files and running the Siebel ARM Analyzer Tool, see *Analyzing Siebel ARM Files Using the Siebel ARM Analyzer Tool*.

Use an XML editor or Web browser to view the XML output files, which result from several types of analyses. Values of timing measurements are included among the XML tags.

Use third-party software (for example, a spreadsheet program) to view the output files that result from the conversion of Siebel ARM files to CSV files. Tags and values of timing measurements are included.

Siebel ARM records all timings included in both the XML and CSV output in milliseconds.

About Performance Aggregation Analysis and Data

This topic is part of *Analyzing Siebel ARM Files Using the Siebel ARM Analyzer Tool*.

Performance aggregation analysis is a compilation of the data contained in a Siebel ARM binary file. Siebel ARM files group performance data based on the instrumented areas. For information and a listing of instrumented areas, see *About Siebel Application Response Measurement*. For more information about creating this format of Siebel ARM output, see *Running Performance Aggregation Analysis*.

Running a performance aggregation analysis of a Siebel ARM file results in an extensible markup language (XML) output file. This file contains timing data for the instrumented areas.

The amount of information contained in the performance aggregation analysis XML output depends on the argument used for the `-a` flag when you perform the analysis (either AREA or DETAILS) and on the setting for the **`SARM Granularity Level`** parameter. For information about this parameter, see *About Siebel ARM Parameters and Variables*.

The performance aggregation XML output file contains the following tag schema when the `-a` flag argument is set to DETAILS. If the `-a` flag argument is set to AREA when you run the analysis, then the tag schema is the same, minus the **`<NumberOfSubAreas>`** and **`<SubArea>`** information.

```
<Area>
 <Name>
 <Symbol>
 <NumberOfSubAreas>
 <Invocations>
 <Recursive>
 <NonRecursive>
 <ResponseTime>
 <Total>
 <Average>
 <StandardDeviation>
 +<Maximum>
 +<Minimum>
 <ExecutionTime>
 <Total>
 <Calls>
 <Average>
 <Maximum>
```

```
<Minimum>
<PercentOfResponse>
<RecursiveTime>
<Total>
<Calls>
<Average>
<Maximum>
<Minimum>
<PercentOfResponse>
<InclusiveMemory>
<Total>
<Average>
<StandardDeviation>
+<MaxAllocated>
+<MaxDeallocated>
<ExclusiveMemory>
<Total>
<Average>
<StandardDeviation>
+<MaxAllocated>
+<MaxDeallocated>
<SubArea>
<Name>
<Symbol>
<NumberOfInstances>
+<Invocations>
+<ResponseTime>
+<ExecutionTime
+<Memory>
+<Instance>
+<Parents>
+<Children>
<Parents>
<NumberOfParents>
<ParentArea>
<Name>
<Symbol>
+<<InvocationsFromParents>
+<ResponseTime>
+<Memory>
<Children>
<NumberOfChildren>
<ChildArea>
<Name>
<Symbol>
+<InvocationsOfChild>
+<ResponseTime>
+<Memory>
```

For descriptions of each of the tags, see the following table.

Tag	Description
Area	Specifies performance data captured for a specific area of the Siebel ARM architecture. There might be one or more areas captured with performance data. For more information about Siebel ARM areas, see *About Siebel Application Response Measurement*.
Name	Name of the area containing performance data. For a listing of area names, see *About Siebel Application Response Measurement*.

Tag	Description
Symbol	Symbol of the area containing performance data. For a listing of symbol names, see *About Siebel Application Response Measurement*.
NumberOfSubAreas	A count of subareas within the area that contain data. This figure also indicates the number of **<SubArea>** tags appearing under the particular **<Area>** tag.
Invocations	Number of times this area was called during the monitoring period. **Recursive** – One of the key features of Siebel ARM is the capability to handle recursion. An example of a recursive call is if a workflow step calls a Siebel Application Object Manager function, which also invokes another workflow step. When accounting for the number of times the workflow layer is called, Siebel ARM uses two metrics: Recursive and NonRecursive. In the previous example, Recursive is 1 and NonRecursive is also 1. When calculating the response time, only the root-level call is accounted for, that is, the first workflow call to the Siebel Application Object Manager function. When calculating execution time, both calls are accounted for. **Nonrecursive** – Number of times an instrumentation area is called. This tag helps identify how fast it takes a layer to respond to a request.
ResponseTime	Specifies the time spent for a request to enter and exit an instrumentation area (layer), including calls to other child areas. It is also called inclusive time in other commercial profiling tools. Other tags in this area include: **Total** – Total time spent by requests through this instrumentation area (layer). **Average** – Average response time for a request. **StandardDeviation** – The standard deviation value of request times through this area. **+<Maximum>** – The maximum time spent by a request in this area. Expand this tag to review further details on the specific Siebel ARM node where this time was spent. For more information about Siebel ARM node tags, see *About Call Graph Generation Analysis and Data*. **+<Minimum>** – The minimum time spent by a request in this area. Expand this tag to review further details on the specific Siebel ARM node where this time was spent. For more information about Siebel ARM node tags, see *About Call Graph Generation Analysis and Data*.
ExecutionTime	Specifies the total time spent in a particular instrumentation area, not including the time spent in the descendant layers. It is also called exclusive time in other commercial profiling tools. Other tags in this area include: **Total** – Total time spent for a request to enter and exit an instrumentation area (layer). **Calls** – Total number of calls including both recursive and nonrecursive calls. **Average** – Average time spent for a request to enter and exit an instrumentation area (layer). **Maximum** – Maximum time for a request to enter and exit an instrumentation area (layer). **Minimum** – Minimum time for a request to enter and exit an instrumentation area (layer). **PercentageofResponse** – Percentage of the total response time spent in the area.
RecursiveTime	Specifies the total time spent in recursive calls within this area. That is, the time spent in this area when it calls itself. Other tags in this area include: **Total** – Total time spent for recursive requests.

Tag	Description
	Calls – Number of recursive calls. **Average** – Average time spent for a recursive request. **Maximum** – Maximum time spent by a recursive request. **Minimum** – Minimum time spent by a recursive request. **PercentageofResponse** – Percentage of the total response time spent recursively in the area.
InclusiveMemory	Specifies the amount of memory used by requests that enter this area and any child or descendent areas. The memory value is recorded in bytes. Other tags in this area include: **Total** – Total memory usage by requests in this area. **Average** – Average memory usage by requests in this area. **StandardDeviation** – The standard deviation value of memory usage in this area. **+<MaxAllocated>** – Expand this tag to reveal further data on Siebel ARM node where maximum memory was allocated. **+<MaxDeallocated>** – Expand this tag to reveal further data on Siebel ARM node where memory was deallocated. For more information about Siebel ARM node tags, see *About Call Graph Generation Analysis and Data*.
ExclusiveMemory	Specifies the amount of memory used by requests that enter only this area. The memory value is recorded in bytes. Other tags in this area include: **Total** – Total memory usage by request in this area. **Average** – Average memory usage by a request in this area. **StandardDeviation** – The standard deviation value of memory usage in this area. **+<MaxAllocated>** – Expand this tag to reveal further data on Siebel ARM node where maximum memory was allocated. **+<MaxDeallocated>** – Expand this tag to reveal further data on Siebel ARM node where memory was deallocated. For more information about Siebel ARM node tags, see *About Call Graph Generation Analysis and Data*.
SubArea	Specifies performance data captured for a specific subarea of the given area. There can be one or more subareas captured with performance data under a given area. **Name** – Name of the subarea containing performance data. **Symbol** – Symbol of the subarea containing performance data. **NumberOfInstances** – A count of instances within the subarea that contain data. This figure also indicates the number of **<Instance>** tags appearing under the particular **<SubArea>** tag. An instance is a further level of detail defining the subarea. **Invocations** – Number of times this subarea was called during the monitoring period.

Tag	Description
	+<ResponseTime> – Specifies the time spent for requests to enter and exit the subarea. Expand this tag to review further timing details. These tags are the same as those defined for the area **ResponseTime** tag. **+<ExecutionTime>** – Specifies the time spent in the subarea. Expand this tag to review further timing details. These tags are the same as those defined for the area **ExecutionTime** tag. **+<InclusiveMemory>** – Specifies the amount of memory used by requests that enter this subarea and any child or descendent areas. The memory value is recorded in bytes. Expand this tag to review further memory details. The expanded tags are the same as those defined for the area **InclusiveMemory** tag. **+<ExclusiveMemory>** – Specifies the amount of memory used by requests that enter only this subarea. The memory value is recorded in bytes. Expand this tag to review further memory details. The expanded tags are the same as those defined for the area **ExclusiveMemory** tag. **+<Instance>** – An instance is another level of detail defining the subarea. Expand this tag to review further the instance's details. These tags are the same as those defined for the area tag. **+<Parents>** – Specifies the parents of the subarea; that is, those areas that called the subarea. Expand this tag to review further parent subarea details. These tags are the same as those defined for the area **Parents** tag. **+<Children>** – Specifies the children of the subarea; that is, those areas called by the subarea. Expand this tag to review further parent subarea details. These tags are the same as those defined for the area **Children** tag.
Parents	Specifies the parents of the subarea; that is, those areas that called the given area. This information helps identify the caller or callers of an area and the total time and number of calls the area contributed to its parent's response time. Other tags in this area include: **NumberOfParents** – A count of parent areas calling the given area. **ParentArea** – Specifies performance data captured for a specific parent area of the Siebel ARM architecture. There can be one or more parent areas captured with performance data. **Name** – Name of the parent area calling the given area. **Symbol** – Symbol of the parent area calling the given area. **+<InvocationsFromParents>** – Number of times the given area was called by the parent area. Expand this tag for further timing details. **+<ResponseTime>** – Specifies the time spent for a request to enter and exit the parent area. Expand this tag for further parent area response time details. **+<Memory>** – Specifies the amount of memory used by parent area. Expand this tag to review further parent subarea details.
Children	Specifies the areas called by a parent area; that is, those areas called by the given area. Expanding an area's children information determines response time break downs within each of the children. Other tags in this area include: **NumberOfChildren** – A count of child areas called by the given area. **ChildArea** – Specifies performance data captured for a specific child area of the Siebel ARM architecture. There can be one or more child areas captured with performance data. **Name** – Name of the child area called by the given area. **Symbol** – Symbol of the child area called by the given area.

Tag	Description
	+<InvocationsOfChild> – Number of times the child area was called by the given area. Expand this tag for further timing details. **+<ResponseTime>** – Specifies the time spent for a request to enter and exit the child area. Expand this tag for further child area response time details. **+<Memory>** – Specifies the amount of memory used by child area. Expand this tag to review further child subarea details.

About Call Graph Generation Analysis and Data

This topic is part of *Analyzing Siebel ARM Files Using the Siebel ARM Analyzer Tool*.

A call graph generation analysis constructs a map of call references. Each node in the call map represents an instrumentation instance, that is, response times for an individual request through an instrumented area. For information about instrumented areas, see *About Siebel Application Response Measurement*. For more information about creating this format of Siebel ARM output, see *Running Call Graph Generation*.

Running a call graph generation analysis of a Siebel ARM file results in an extensible markup language (XML) output file. For a given Siebel ARM file, the Siebel ARM Analyzer Tool constructs a map with call references. Each node in the call map represents an instrumentation instance. Use this option to generate an XML file containing all of the calls made by each component (if that component captures response time data).

The XML output file contains the following tag schema, which records the details of the calls.

```
<SarmNode>
 <SarmID>
 <TypeLevel>
 <RootID>
 <ParentSARMID>
 <ParentTimeID>
 <ParentProcID>
 <AreaCodeSymbol>
 <AreaDescription>
 <SubAreaCodeSymbol>
 <SubAreaDescription>
 <Count>
 <Duration>
 <PooledMemoryUsage>
 <PooledMemoryCalls>
 <SystemMemoryUsage>
 <SystemMemoryCalls>
 <AppInt1>
 <AppInt2>
 <AppString1>
 <AppString2>
 +<ChildNode>
</SarmNode>
```

For descriptions of each of the tags, see the following table.

Tag	Description
SarmNode	Data contained within this tag represents an instance of a Siebel ARM node, which is an instrumented area of the Siebel ARM architecture. Each Siebel ARM node can have zero to many nodes as its descendants.
SarmID	A unique number representing the Siebel ARM node.
TypeLevel	The granularity level at which Siebel ARM records the Siebel ARM node information. For more information about granularity level, see *About Siebel ARM Parameters and Variables*.
RootID	The Siebel ARM ID of the root Siebel ARM node.
ParentSARMID	The Siebel ARM ID of the parent Siebel ARM node from which the request traveled.
ParentTimeID	A unique ID number that generates from the starting time of the corresponding parent Siebel ARM node.
ParentProcID	The parent process ID, that is, the OS (operating system) process ID for the Siebel component.
AreaCodeSymbol	Symbol of the instrumentation area within the Siebel architecture. For information about Siebel architecture areas, see *About Siebel Application Response Measurement*.
AreaDescription	Name of the instrumentation area within the Siebel architecture. For information about Siebel architecture areas, see *About Siebel Application Response Measurement*.
SubAreaCodeSymbol	Symbol of the subarea within an area of the Siebel architecture. For information about Siebel architecture areas, see *About Siebel Application Response Measurement*.
SubAreaDescription	Name of the subarea within an area of the Siebel architecture. For information about Siebel architecture areas, see *About Siebel Application Response Measurement*.
Count	Number of times Siebel ARM accesses this Siebel ARM Node.
SubArea	Detailed instrumentation within an area of the Siebel architecture. For example, Siebel ARM captures response time for invoking a method (Invoke Method) or executing a step (Step Execution) within a Workflow execution.
StartTime	Internal representation of the Siebel ARM record time stamp.
Duration	Total time to execute the instrumented area.
PooledMemoryUsage	Amount of memory consumed from or released to the Siebel High Performance memory allocator.
PooledMemoryCalls	The number of calls made to the High performance memory allocator.
SystemMemoryUsage	Amount of memory consumed from or released to the operating system.

Tag	Description
SystemMemoryCalls	The number of calls made to the operating system.
UserInt1	Context information captured at the point of instrumentation. The value depends on the instrumented area.
UserInt2	Context information captured at the point of instrumentation. The value depends on the instrumented area.
UserString	Context information captured at the point of instrumentation. The value depends on the instrumented area. For example, name of the method invoked or workflow process initialized.
AppInt1 and **AppInt2**	Context integer value captured at the point of instrumentation. The value depends on the instrumented area.
AppString1 and **AppString2**	Context string value captured at the point of instrumentation. The value depends on the instrumented area. For example, name of the method invoked or workflow process initialized.
+<ChildNode>	Expand this tag to reveal performance details on descendent nodes of the given node. The descendent nodes are defined the same as the parent node, that is, the tag definitions are the same as above.

About User Session Trace Analysis and Data

This topic is part of *Analyzing Siebel ARM Files Using the Siebel ARM Analyzer Tool*.

Running a user session trace analysis using Siebel ARM files from the Siebel Server results in an extensible markup language (XML) output file. The XML output file contains detailed information on each of the SWE requests made by the user identified when running the Siebel ARM file conversion.

If the user logs onto the system multiple times, the output shows that there are multiple sessions. The SWE requests are grouped into specific login sessions and sorted by the time the requests were made. For more information about the Siebel ARM architecture, see *About Siebel Application Response Measurement*. For more information about creating this format of Siebel ARM output, see *Running User Session Trace*.

The XML output file contains the following tag schema, which records the details of user session trace. The user session trace data also contains the tag schema of the performance aggregation analysis. For more information about those tags, see *About Performance Aggregation Analysis and Data*.

```
<UserID>
 <Session>
 <SessionID>
 <UserActionID>
 <ID>
 <SWERequest>
 <ReqID>
 <TotalServerTime>
 <NetworkTime>
 <SiebServerTime>
 <DatabaseTime>
```

```
<DatabaseCalls>
+<SiebsrvrDetail>
```

For descriptions of each of the tags specific to the user session trace analysis, see the following table. For descriptions of the tags that are also a part of the performance aggregation analysis, see *About Performance Aggregation Analysis and Data*.

Tag	Description
UserID	User login name. For example, SADMIN.
Session	Specifies performance data captured for a specific user session contained within this tag.
SessionID	Refers to a unique user session ID in hexadecimal format. The first component of the Session ID refers to the Server ID, the second refers to the Process ID, and the last section to the Task ID. For example: `!1.2b40.182b` Server ID = `!1` Process ID = `2b40` (`2b40` is 11072 in decimal format and represents the Operating System Process ID number.) Task ID = `182b` (`182b` is 6187 in decimal format and represents the task ID number.)
UserActionID	Data contained within this tag represents a specific individual action or request of the user.
ID	Number that identifies the specific user action or request in sequence for that particular user session.
SWERequest	Specifies performance timing data for the specific user action or request.
ReqID	An incremental numeric ID number corresponding to a Siebel Application Interface request.
TotalServerTime	Total request time on the servers (includes Siebel Server and network time).
NetworkTime	The total time spent between the Siebel Application Interface and the Siebel Server. This time can also include some Siebel infrastructure time routing the request to the handling Siebel Server task.
SiebServerTime	The time spent on the Siebel Server.
DatabaseTime	The time spent on the network when communicating to the database.
DatabaseCalls	The number of calls to the Siebel Server database connector layer.
+<SiebsrvrDetail>	Response time and execution time for each of the architectural areas of instrumentation for a given session. For more information about these tags, see *About Performance Aggregation Analysis and Data* and *About Call Graph Generation Analysis and Data*.

About Siebel ARM to CSV Conversion Data

This topic is part of *Analyzing Siebel ARM Files Using the Siebel ARM Analyzer Tool*.

CSV format is a comma-separated file without any interpretation or aggregation. The CSV file contains data organized under column headers. Use third-party software tools to view this output, for example, a spread sheet. For more information about creating this format of Siebel ARM output, see *Running Siebel ARM Data CSV Conversion*. For a listing and description of these column headers, see the definitions of the tags for the call graph analysis in *About Call Graph Generation Analysis and Data*. Information can be reviewed and organized by these columns. See the following for an example of CSV data.

	B	C	D	E	F	G	H	I	J	K	L
1	ThreadID	IsRoot	Type(level)	RootID	ParentSarmID	ParentTimeID	ParentProcID	AreaCodeSymbol	AreaDesc	SubAreaCodeSymbol	SubAreaDesc
2	5300	N	Sarm(1)	1	1	1074205585	1848	Area_SWSE	Web Server Plugin	Sub_SWSE_SENDMSG	Send message to app se
3	5300	N	Sarm(1)	1	1	1074205585	1848	Area_SWSE	Web Server Plugin	Sub_SWSE_SENDMSG	Send message to app se
4	5300	N	Sarm(1)	1	1	1074205585	1848	Area_SWSE	Web Server Plugin	Sub_SWSE_SENDMSG	Send message to app se
5	5300	N	Sarm(1)	1	1	1074205585	1848	Area_SWSE	Web Server Plugin	Sub_SWSE_SENDMSG	Send message to app se
6	5300	N	Sarm(1)	1	1	1074205585	1848	Area_SWSE	Web Server Plugin	Sub_SWSE_SENDMSG	Send message to app se
7	5300	N	Sarm(1)	1	1	1074205585	1848	Area_SWSE	Web Server Plugin	Sub_SWSE_SENDMSG	Send message to app se
8	5300	N	Sarm(1)	1	1	1074205585	1848	Area_SWSE	Web Server Plugin	Sub_SWSE_SENDMSG	Send message to app se
9	5300	N	Sarm(1)	1	1	1074205585	1848	Area_SWSE	Web Server Plugin	Sub_SWSE_SENDMSG	Send message to app se
10	5300	N	Sarm(1)	1	1	1074205585	1848	Area_SWSE	Web Server Plugin	Sub_SWSE_SENDMSG	Send message to app se
11	5300	N	Sarm(1)	1	1	1074205585	1848	Area_SWSE	Web Server Plugin	Sub_SWSE_SENDMSG	Send message to app se
12	5300	N	Sarm(1)	1	1	1074205585	1848	Area_SWSE	Web Server Plugin	Sub_SWSE_SENDMSG	Send message to app se
13	5300	N	Sarm(1)	1	1	1074205585	1848	Area_SWSE	Web Server Plugin	Sub_SWSE_SENDMSG	Send message to app se
14	5300	N	Sarm(1)	1	1	1074205585	1848	Area_SWSE	Web Server Plugin	Sub_SWSE_SENDMSG	Send message to app se
15	5300	N	Sarm(1)	1	1	1074205585	1848	Area_SWSE	Web Server Plugin	Sub_SWSE_SENDMSG	Send message to app se
16	5300	N	Sarm(1)	1	1	1074205585	1848	Area_SWSE	Web Server Plugin	Sub_SWSE_SENDMSG	Send message to app se
17	5300	N	Sarm(1)	1	1	1074205585	1848	Area_SWSE	Web Server Plugin	Sub_SWSE_SENDMSG	Send message to app se
18	5300	N	Detail(2)	1	1	1074205585	1848	Area_SARM	SARM Framework	Sub_SARM_IO	Flush SARM Buffer To Di
19	5300	N	Sarm(1)	1	1	1074205585	1848	Area_SWSE	Web Server Plugin	Sub_SWSE_SENDMSG	Send message to app se
20	5300	N	Sarm(1)	1	1	1074205585	1848	Area_SWSE	Web Server Plugin	Sub_SWSE_SENDMSG	Send message to app se