Siebel Performance Tuning Guide

Version 8.0

December 2006

ORACLE

Contents

Chapter 5: Tuning Siebel Web Client for Performance

Chapter 6: Tuning Siebel Communications Server for Performance

Chapter 7: Tuning Siebel Workflow for Performance

Chapter 8: Tuning Siebel Configurator for Performance

Chapter 9: Tuning Siebel EAI for Performance

Chapter 10: Tuning Siebel EIM for Performance

Index

1 What's New in This Release

What's New in Siebel Performance Tuning Guide, Version 8.0

Table 1 lists changes described in this version of the documentation to support Version 8.0 of Siebel Business Applications from Oracle.

Table 1. New Features in Siebel Performance Tuning Guide, Version 8.0

Topic	Description
"Configuring Database Connection Pooling for AOMs" on page 34	Revised topic. When you enable database connection pooling, multiple user sessions no longer share one default database connection; they share a pool of database connections.
"Administering the Siebel Configurator Cache" on page 101	Revised topic. This release allows you to refresh the Siebel Configurator cache with changes made to customizable products, product classes, and attribute definitions in one operation.
"Database Guidelines for Optimizing EIM" on page 133	Revised topic. This topic contains a number of new recommendations for optimizing EIM performance on the IBM DB2 UDB database.
"Tuning Siebel Remote for Performance" on page 145	Revised topic. This chapter contains a number of new recommendations for optimizing the performance of Siebel Remote.
"Monitoring Siebel Application Performance with Siebel ARM" on page 191	New chapter. This chapter provides an overview of the Siebel Application Response Measurement (Siebel ARM) feature.
"Analyzing Siebel ARM Data" on page 199	New chapter. This chapter describes how to use a new command-line tool, the Siebel ARM Query Tool, to analyze Siebel ARM data. Information that appeared in previous editions of this guide describing how to use the Siebel ARM Analyzer Tool also appears in this chapter.

2 Siebel Architecture and Infrastructure

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

Cross-references are provided to other chapters of this guide on how to configure specific areas of Siebel Business Applications. Optimally tuning these areas achieves a balance between performance and scalability.

For more information and details about the Siebel Business Applications architecture and infrastructure, see the following documentation on the *Siebel Bookshelf*:

- *Siebel Deployment Planning Guide*
- *Siebel Installation Guide* for the operating system you are using
- *Siebel System Administration Guide*
- *Configuring Siebel Business Applications*

NOTE: Every implementation of Siebel Business Applications is unique. Your Siebel application architecture, infrastructure, and configurations may differ depending on your business model.

About Performance and Scalability

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 may include the time required to log into 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 machines (vertical scalability) or new Siebel Server machines (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 further definitions of terminology related to performance and scalability, see "Performance Tuning Terminology" on page 19.

About Siebel Architecture and Infrastructure

Figure 1 on page 15 shows a generic representation of the architecture and infrastructure of a Siebel Business Applications 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 the *Siebel Installation Guide* for the operating system you are using.

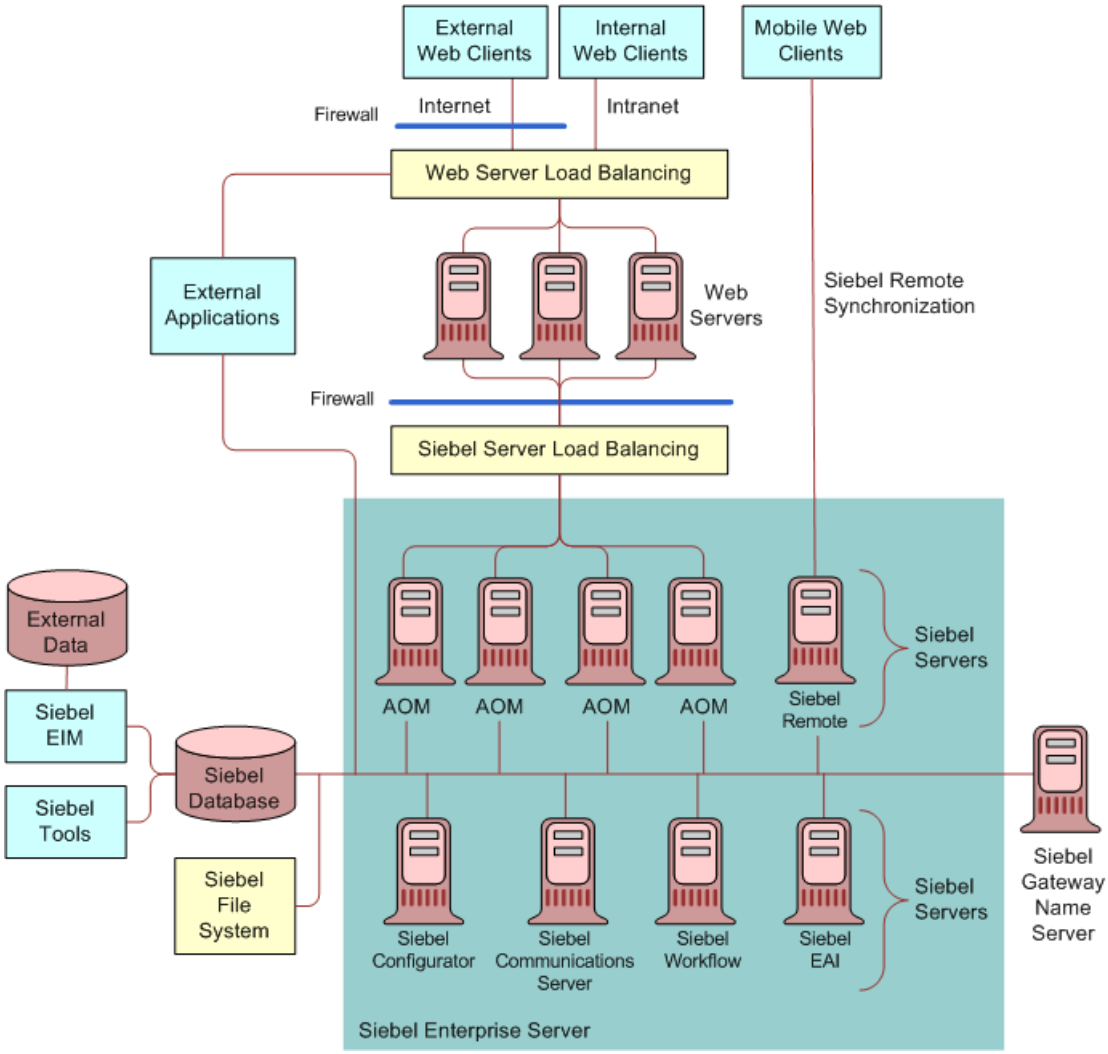

Figure 1. Generic Architecture of Siebel Business Applications

Siebel Architecture and Infrastructure Areas for Tuning

The following list provides details on tuning 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 Chapter 14, "Monitoring Siebel Application Performance with Siebel ARM."

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

  Running AOM components has significant performance and scalability implications. In general, the goal for tuning an AOM is to maximize scalability with little or no performance degradation as more users use the system.

  Although AOM components can be tuned for optimal performance, capacity for this and all other Siebel Server components is ultimately limited by Siebel Server machine resources such as CPU and memory.

  For details on tuning this area, see Chapter 3, "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.

  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 AOM, network bandwidth and latency, Web server, Siebel Database, and the Siebel application configuration (represented in the Siebel repository file). It is also subject to local machine resources and settings, including browser settings such as those for caching.

  For details on tuning this area, see Chapter 5, "Tuning Siebel Web Client for Performance." See also Chapter 12, "Tuning Customer Configurations for Performance."

- **Siebel Communications Server.** Siebel Communications Server provides an infrastructure to support several kinds of communications activities for Siebel application users, including session communications (such as voice calls) and inbound and outbound communications (such as email).

  Siebel Communication Server processing may affect end user response time, and may demand additional AOM resources to support user sessions. Performance and scalability is subject to third-party server configuration and capacity and Siebel Server machine resources and configuration.

  For details on tuning this area, see Chapter 6, "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 may affect end user response time (for synchronous requests), and may demand additional AOM resources to support user sessions. Performance and scalability is subject to Siebel Server machine resources and configuration.

  For details on tuning this area, see Chapter 7, "Tuning Siebel Workflow for Performance."

- **Siebel Configurator.** Siebel Configurator supports order management and product configuration functions for Siebel applications.

  Siebel Configurator processing may affect end user response time (for configuration sessions), and may demand additional AOM resources to support user sessions. Performance and scalability is subject to Siebel Server machine resources and configuration.

  For details on tuning this area, see Chapter 8, "Tuning Siebel Configurator for Performance."

- **Siebel Enterprise Application Integration (Siebel EAI).** Siebel EAI provides components for integrating Siebel Business Applications with external and internal applications, and provides inbound and outbound interfaces to and from a Siebel application.

  Siebel EAI processing may affect end user response time (for real-time interfaces), and may demand additional AOM resources to support user sessions. Performance and scalability is subject to Siebel Server machine resources and configuration.

  For details on tuning this area, see Chapter 9, "Tuning Siebel EAI for Performance."

- **Siebel Enterprise Integration Manager (Siebel EIM).** Siebel EIM is a server component in the Siebel EAI component group that transfers data between the Siebel database and other corporate data sources.

  For details on tuning this area, see Chapter 10, "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 details on tuning this area, see Chapter 11, "Tuning Siebel Remote for Performance."

- **Siebel Tools.** Siebel Tools is an integrated development environment 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 details on tuning this area, see Chapter 12, "Tuning Customer Configurations for Performance."

- **Operating systems.** For details on tuning your Microsoft Windows or UNIX operating system, see Chapter 13, "Tuning Operating Systems for Performance."

About Siebel User Request Flow

Figure 2 illustrates how a user request is processed within the Siebel Business Applications 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*.

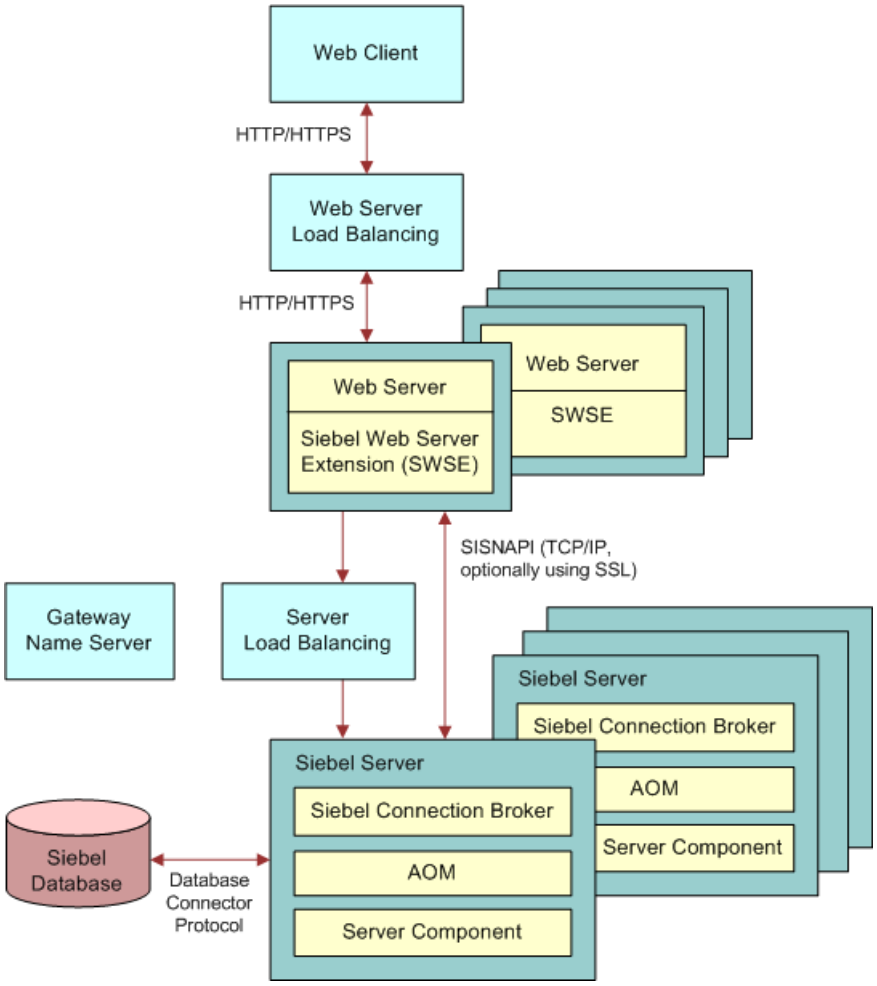

Figure 2. Generic User Request Flow in Siebel Business Applications

A typical Siebel 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 an existing or new HTTP connection. The request may go through a network router, proxy server, cache engine, or other mechanism.

3 If present, Web server load balancing software evaluates the request and determines the best Web server to forward the request to. It then forwards the request to a Web server.

4 The Web server receives the HTTP request, determines that it is a Siebel application request, and forwards the request to the Siebel Web Server Extension (SWSE) installed on the Web server.

5 The SWSE parses the HTTP message and generates a SISNAPI message, based on the content of the HTTP message. SWSE also parses the incoming cookie or URL to obtain the user session ID.

   - If using Siebel load balancing, SWSE forwards the request to a Siebel Server in round-robin fashion.
   - If using a third-party HTTP load balancer, SWSE forwards the request to the load balancer. The load balancer uses user-configured routing rules to forward the request to a Siebel Server.

   SISNAPI (Siebel Internet Session application programming interface) is a messaging format that runs on top of the TCP/IP protocol. It is used for network communication between Application Object Managers (AOMs) and SWSE.

6 On the Siebel Server, an AOM receives and processes the SISNAPI message. If a database query is needed to retrieve the information, the AOM 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 AOM. The AOM forwards the message to the Web server that originated it. If using a third-party HTTP load balancer, the message may go through the load balancer before reaching the Web server.

8 The SWSE on the Web server receives the SISNAPI message, and translates it back to HTTP. It then forwards the HTTP message to the Web server. The message is now in the form of Web page content.

9 The Web server load balancer, 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

Table 2 provides definitions of specific terms related to performance and tuning Siebel Business Applications. For definitions of *performance* and *scalability*, see "About Performance and Scalability" on page 13.

For more information about some of these terms and concepts (including concurrent users and think time) in the context of tuning Application Object Manager (AOM) components, see "Performance Factors for AOM Deployments" on page 23.

Table 2. Performance Tuning Terminology

Term	Definition
Concurrent users	The number of application users actively using and accessing the Siebel application, or a particular element such as an AOM process, at a particular time.
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, the think time in this case is 10 seconds. Average think time is a critical element in performance and scalability tuning, particularly for AOM. 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 AOM consists of multiple OS processes, referred to as multithreaded processes.
Multithreaded process (or MT server)	A process running on a multithreaded Siebel Server component that supports multiple threads (tasks) per process. AOM 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 may 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.

3 Tuning the Siebel Application Object Manager for Performance

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

For more information about the Siebel Server and AOM 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* for the operating system you are using

About the Application Object Manager

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

A different AOM component is provided for each base application among the Siebel Business Applications or Siebel Industry Applications. For example:

- Call Center Object Manager (SSCObjMgr_enu) is the AOM for Siebel Call Center in a U.S. English environment.
- Sales Object Manager (SSEObjMgr_enu) is the AOM for Siebel Sales in a U.S. English environment.
- eService Object Manager (eServiceObjMgr_enu) is the AOM for Siebel eService in a U.S. English environment.

NOTE: Separate AOMs are provided for each installed language in which you may run your Siebel applications. For example, Call Center Object Manager for French is SCCObjMgr_fra.

When configured appropriately, AOM components on your Siebel Servers can use memory and CPU resources efficiently, and can communicate efficiently with the Siebel Database, the Siebel Web Server Extension (SWSE), and other components in the Siebel Enterprise.

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

The overall performance of the AOM contributes significantly to the response time as experienced by your end users.

AOM Infrastructure

An AOM 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 AOM configuration.

Each process can host multiple user sessions (as tasks), which in turn are implemented as threads within the process. These threads may be dedicated to particular user sessions, or they may 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 into the system, additional processes may 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 AOM" on page 42.
- 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 AOM parameters MaxMTServers and MinMTServers refer to multithreaded processes.

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

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

AOM Communications with Other Modules

Each AOM task uses Siebel Server infrastructure capabilities to communicate with the Siebel Database, the Web server (through the SWSE), and other Siebel Enterprise Server components.

- 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 details on configuring database connection pooling, see "Configuring Database Connection Pooling for AOMs" on page 34.

- Communication with Siebel Connection Broker (SCBroker) uses mechanisms internal to the operating system. SCBroker receives each SISNAPI connection request from the SWSE and forwards the connection request to the AOM multithreaded process with the fewest running tasks. Once the connection has been forwarded, requests flow directly from SWSE to AOM.

  For more information about tuning SCBroker, see the load balancing sections in *Siebel Installation Guide* for the operating system you are using and the *Siebel System Administration Guide*.

- Communication with the Siebel Web Server Extension uses SISNAPI (Siebel Internet Session API), a messaging format that runs on top of the TCP/IP protocol. SISNAPI connections can be configured to use encryption and authentication based on Secure Sockets Layer (SSL).

  For details on tuning SISNAPI communications, see "Configuring SISNAPI Connection Pooling for AOM" on page 45.

- Communication with other Siebel Enterprise Server components (including other Siebel Servers) also uses SISNAPI, going through Server Request Broker (SRBroker).

  For more information about tuning SRBroker, see "Tuning Server Request Broker (SRBroker)" on page 47.

About Tuning the AOM

Tuning activities directly or indirectly applicable to AOM components may involve any or all of the following:

- Configuring parts of your system using the Siebel Enterprise Server configuration utility.
- Using the Siebel Server Manager to tune parameters for the Enterprise Server, the Siebel Server, or the AOM component. These parameters are stored in the siebns.dat file in a directory on the Siebel Gateway Name Server.
- Selectively enabling component groups and components on each Siebel Server. Only enable the component groups and components you need.
- Tuning parameters in the eapps.cfg file on the Siebel Web Server Extension. This file is located in the bin subdirectory of the Siebel Web Server Extension installation directory, on the Web server machine.
- Tuning parameters in the application configuration file, such as uagent.cfg for Siebel Call Center. This file is located in the bin/*language* subdirectory of the Siebel Server installation directory. Parameters in certain sections of this file, such as [InfraUIFramework], are read by the relevant AOM, such as SCCObjMgr_enu for Siebel Call Center in a U.S. English environment.

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

Performance Factors for AOM Deployments

In planning to deploy AOMs, or in troubleshooting performance for existing AOM deployments, you must consider several factors that determine or influence performance.

Factors that are central to the task of configuring the AOM are also called *performance drivers*. Performance drivers for AOM 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 sections provide information and guidelines to help you achieve and maintain optimal performance and scalability.

These factors are critical in initially configuring your AOMs, particularly when specifying values for the AOM component parameters MaxTasks, MaxMTServers, and MinMTServers, which are discussed in "Tuning AOM Components for CPU and Memory Utilization" on page 27.

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

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

The maximum number of concurrent users per Siebel Server—assuming, for example, that a particular Siebel Server machine is dedicated to running AOM 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 AOM is limited by the value of the MaxTasks parameter. The effective maximum is also limited by the number of multithreaded processes for this AOM and by your hardware resources.

Depending on the average think time and other factors, each multithreaded process (process within the AOM) 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 (part of Siebel Industry Applications) and Siebel Configurator typically support about 25 concurrent users. For more information about the Object Manager for Siebel Configurator (Siebel Product Configuration Object Manager), see Chapter 8, "Tuning Siebel Configurator for Performance."

Think Time

The 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.

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 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 AOM Components for CPU and Memory Utilization" on page 27.

The ratio of 100 is based on having approximately three users running operations at the exact same time (100/30 = 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 may 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 * 3.3 = approximately 50, or 50/15 = approximately 3.3).

Nature of Siebel Application Deployment

Which Siebel applications and other modules you are using, how you have configured your Siebel applications, how you have deployed your applications, and other such factors also affect AOM performance and how many concurrent users 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? Typically, employee applications use high interactivity and customer applications use standard interactivity.
- 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 you do using Siebel Tools? For more information, see Chapter 12, "Tuning Customer Configurations for Performance."

  The number of concurrent tasks you can support varies based on the level of customization or the use of process automation for the application the AOM 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 may be relatively complex and demand more resources than most other operations.
- Will you use specialized functionality such as offered by Siebel 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 machine, particularly CPU and memory, are a factor in how many concurrent users can be supported for each AOM component. For example, a four-way machine has twice the resources of a two-way machine and can potentially support twice as many concurrent users. Key hardware resources for AOM performance include:

- **CPU.** The CPU rating and the number of CPUs per server machine.
- **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 AOM tuning. They do significantly affect performance for the Siebel Database and the Siebel File System.

The total number of machines you can devote to supporting AOM components will determine the total number of concurrent users.

Topology Considerations for AOM Deployments

Your Siebel applications can be deployed using a variety of topologies, or system layouts. Although AOMs 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 machines you will run Siebel Server, and on how many of these you will run AOM components. In some cases, you may choose to run multiple components on the same Siebel Server.

NOTE: AOM components are typically the major resource consumers for your Siebel Server machines. Tuning considerations discussed in this chapter generally assume that you are not running additional components on an AOM machine that will significantly contend for available resources.

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

Best Practices for AOM Tuning

Using your hardware resources optimally and configuring your system appropriately can help you to achieve your performance goals. You should 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 AOM Deployments" on page 23.

Review the following subsections for more details on AOM tuning:

- "Tuning AOM Components for CPU and Memory Utilization" on page 27
- "Tuning Parameters for AOM Caches" on page 31
- "Additional Parameters Affecting AOM Performance" on page 32

■ "Memory Consumers in AOM" on page 33

Tuning AOM Components for CPU and Memory Utilization

This section provides background information and guidelines for tuning your AOM 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 AOM Deployments" on page 23, which determine the true capacity of the server.

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

About MaxTasks, MaxMTServers, and MinMTServers

The AOM 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 "AOM Infrastructure" on page 22.

- **MaxTasks (Maximum Tasks).** Specifies the total number of tasks (threads) that can run concurrently on this AOM, 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 AOM. 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 AOM when the parent process is started. The parent process may 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 AOM 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 AOM load-balancing behavior, up to the maximum number of tasks and maximum number of multithreaded processes. For details, see "Effect of AOM Parameter Settings" on page 28.

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 AOM 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/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 AOM Deployments" on page 23.

Effect of AOM Parameter Settings

This section illustrates how an AOM behaves given particular example settings for the MaxTasks, MaxMTServers, and MinMTServers parameters. More realistic examples may be found in "Formulas for Calculating AOM Parameter Values" on page 29.

For example, if MaxTasks = 500, and MaxMTServers = 5, then the ratio MaxTasks/MaxMTServers = 100. This means that, at most, 100 threads (tasks) can run in a multithreaded process on this AOM.

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 AOM. This is a form of load balancing internal to the AOM component.
- If the number of concurrent threads reaches 400, and a new request is received, a fifth multithreaded process starts for this AOM. The AOM now distributes threads among five multithreaded processes for this AOM.
- If the AOM reaches 500 concurrent threads, no more client session requests can be handled, because the existing multithreaded processes can start no more threads, and the AOM can start no more multithreaded processes. The AOM can be said to be "maxed out."

If AOM 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 may also time out and stop running; this can happen only when MaxMTServers is greater than MinMTServers.

Guidelines for Configuring AOM Parameters

This section provides formulas and guidelines for setting the MaxTasks, MaxMTServers, and MinMTServers parameters for your AOM components.

NOTE: All elements in the two formulas shown in "Formulas for Calculating AOM Parameter Values" on page 29 vary according to your deployment. The number of concurrent users an AOM can support depends on factors such as the number of processors, session timeout settings, and the average think time.

Typically, the AOM is the only component using significant resources on the Siebel Server machine. If you run multiple server components, or run non-Siebel modules, then an AOM on this machine may 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 machine running AOM. In the formulas outlined below, this is the target number of users plus the number of anonymous browser users, where applicable.
- Determine how many Siebel Server machines are needed to support your concurrent users. This is typically done by Siebel Expert Services or by platform vendors.
- Plug your values into the formulas below, 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/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 browser users required, then adjust settings further as necessary.

Formulas for Calculating AOM Parameter Values

Use the formulas below for calculating parameter values for your AOM components:

- MaxTasks = *target_number_of_users* + *anon_browser_users*
- MaxMTServers = (*target_number_of_users* + *anon_browser_users*)/100
- MinMTServers = MaxMTServers

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

NOTE: The figure of 100 in the MaxMTServers formula represents the ratio of concurrent tasks per multithreaded process. The value of 100 is a rule of thumb only. For details, see below.

Variables in the above formulas are described below:

- *target_number_of_users* = The maximum number of concurrent user sessions your AOM will support (for users who are logged into the application).

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

- *anon_browser_users* = The number of sessions on the AOM dedicated to anonymous browser users (threads that support users who do not log in).

  - For high interactivity applications (typically, employee applications like Siebel Call Center), anonymous browser users are not supported, so this is not a factor.
  - For standard interactivity applications (typically, customer applications like Siebel eService), anonymous browser users may be approximately 25% of the target number of users.

- 100 = The approximate maximum number of concurrent threads each multithreaded process on the AOM 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 AOM usage scenarios. However, if your deployment involves a shorter think time than 30 seconds, or a heavier than average load per thread, 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 AOM Deployments" on page 23.

Example Settings for AOM 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_browser_users*, which are described in the previous section. 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_browser_users* is not a factor. Your parameter values would be:

```
MaxTasks = 500

MaxMTServers = 500/100 = 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_browser_users* might be about 25% of *target_number_of_users* (or 125). Your preliminary parameter values would be:

```
MaxTasks = (500 + 125) = 625

MaxMTServers = (500 + 125)/100 = 6.25 = 7 (round up)

MinMTServers = MaxMTServers = 7
```

Adjust the value of MaxTasks. The variable X = the remainder of (625/7) = 2. Increment MaxTasks by (MaxMTServers - X): 625 + (7 - 2) = 625 + 5 = 630. Therefore, the final calculations for parameter values would be:

```
MaxTasks = 630

MaxMTServers = MinMTServers =7
```

Tuning Parameters for AOM Caches

The AOM uses several caches, which affect memory usage for the AOM. Tuning AOM caches affects AOM performance and memory usage. The following are some of the major caches used by AOM 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 AOM) with database connection pooling.

The value represents the number of SQL cursors per database connection. For an AOM for which the Siebel Server machine is more likely to reach its CPU capacity before it reaches its memory capacity (for example, for Siebel Employee Relationship Management), the default value of 16 for the DSMaxCachedCursors parameter may be appropriate. (Such an application is sometimes referred to as *CPU-bound*.)

For an AOM for which the Siebel Server machine 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 lower value, even to 0. (Such an application is sometimes referred to as *memory-bound*.)

In general, the value should reflect the CPU and memory resource availability on the Siebel Server machine running a particular AOM component. The trade-off in setting this parameter is that allocating memory to caching SQL cursors means 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 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 should 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 AOMs" on page 34.

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 an CPU-bound AOM (for example, for Siebel Employee Relationship Management), the default values for DSMaxCachedDatasetsPerProcess and DSMaxCachedDataSets may be appropriate.

For a memory-bound AOM (for example, for Siebel Call Center), you can set DSMaxCachedDatasetsPerProcess and DSMaxCachedDataSets to a lower value, even to 0.

In general, the values should reflect the CPU and memory resource availability on the Siebel Server machine running a particular AOM component. The trade-off in setting these parameters is that allocating memory to caching SQL data sets means 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 AOM Performance

This section provides guidelines for setting additional parameters that affect AOM performance.

- **MemProtection.** Setting the MemProtection parameter to FALSE for your AOM component may improve performance.

  When this parameter is TRUE (the default), each transaction issues a large number of serialized mprotect statements, the total effect of which may degrade performance on your Siebel Server machines.

  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 AOM 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.

In addition to improving performance, you may also improve scalability if you set the MemProtection parameter to FALSE.

NOTE: Support for setting MemProtection to FALSE is restricted to the AIX and HP-UX platforms.

- **DSPreFetchSize and DSMaxCursorSize.** These parameters should be set *only* for Siebel implementations on IBM DB2 UDB for z/OS and OS/390. For more information on setting these parameters, see *Implementing Siebel Business Applications on DB2 UDB for z/OS and OS/390*.

  For all other databases, these parameters should be set to -1.

- **EnableCDA.** If an AOM component does not need to support Siebel Advisor or browser-based Siebel Configurator, set this parameter to FALSE in the [InfraUIFramework] section of the application configuration file, such as uagent.cfg for Siebel Call Center.

Memory Consumers in AOM

In addition to the caches described earlier, this section discusses major memory consumers in AOM components. For more information on some of these topics, see Chapter 12, "Tuning Customer Configurations for Performance."

- **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 AOMs" on page 34.
- **Scripts.** A script defined on a business component, applet, or business service is loaded into AOM 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.

Other memory consumers in AOM are the following:

- **Navigation pattern.** Numerous scenarios that can be used to navigate in the application can make using global caches ineffective.
- **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 may mean more frequent logins.

- **Users per AOM.** More users per AOM means more sharing of global resources between the users. While the amount of memory used *per user* on this AOM 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 AOMs

This section describes database connection configuration options for AOMs, particularly database connection pooling. It includes the following topics:

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 AOM. RDBMS performance and usage of database connections by non-Siebel components are outside the scope of this section.

About Database Connections for AOM

This section provides an overview of database connections for AOM components, including nonpooled connections and pooled connections. Subsequent sections provide guidelines and instructions for configuring different types of database connection pooling.

About Nonpooled Database Connections

If you do not pool database connections, the number of database connections corresponds to the number of AOM sessions—that is, database connections are not pooled. No special AOM 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 may 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 may alternatively be pooled, as described later in this section.

- **Nonpooled specialized database connections.** If, during a session, specialized functionality is invoked that uses the external transaction management capabilities of the AOM, 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 may alternatively be pooled, as described later in this section.

About Pooled Database Connections

Optionally, you can configure your AOM 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 details, see "Database Connection Pooling Usage Guidelines" on page 36 and "Configuring Pooling for Default Database Connections" on page 38.

- **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 details, see "Database Connection Pooling Usage Guidelines" on page 36 and "Configuring Pooling for Specialized Database Connections" on page 40.

Database Connection Pooling Usage Guidelines

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

For more information about configuring the AOM parameters for database connection pooling mentioned below, see "Configuring Pooling for Default Database Connections" on page 38 and "Configuring Pooling for Specialized Database Connections" on page 40.

When to Consider Using Database Connection Pooling

You should 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 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 machine on which the AOM is running. Pooling default database connections for shared use can reduce AOM 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) may provide a significant benefit for such components.
  - For Siebel Configurator, if you are using the component Siebel Product Configuration Object Manager (alias eProdCfgObjMgr_jpn in a Japanese locale), it is highly recommended to configure persistent connection pooling. For more information about Siebel Configurator, see Chapter 8, "Tuning Siebel Configurator for Performance."

    NOTE: Separate Object Managers are provided for each installed language in which you may 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 in sessionless mode), it may also be helpful to configure persistent connection pooling.

  NOTE: If session caching is configured for a component (by setting the parameter ModelCacheMax), persistent connection pooling may 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" on page 88.

Guidelines for Using Database Connection Pooling

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

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

  NOTE: If you plan to use a ratio higher than 3:1, it is recommended that you consult Siebel Expert Services.

- If you have short (aggressive) average think times in your user scenarios, use a smaller ratio of MaxTasks/MaxSharedDbConns. Longer think times may 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 section will also determine practical limits.

  For more information about think time, see "Performance Factors for AOM Deployments" on page 23.

- Minimize long-running queries. A database connection can process only one database operation at any particular moment. Therefore, if user sessions use all available connections in the database connection pool to execute long-running queries, a new user session that requests a database connection is blocked until a database connection becomes available.

  For example, if a long-running query is run which takes, for example, three seconds, then, for this duration, database requests from another user who requires the same database connection are queued (blocked) until the query operation completes.

  A long-running query may continue to run on the RDBMS even if the user who initiated it has killed the browser.

  When using database connection pooling**,** it is critical to optimize database access in your environment. If long-running queries cannot be avoided, monitor the overall database response time and use a small MaxTasks/MaxSharedDbConns ratio to achieve a satisfactory response time.

  Alternatively, long-running queries may be minimized or avoided by adjusting indexes to include fields that may be sorted or queried by end users, by configuring the application user interface so non-indexed fields are not exposed, or 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" on page 165.

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

  If your deployment uses database authentication, 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, 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 may or may not be shared. For pooled default database connections where connections are persistent but not shared, set MaxSharedDbConns = MaxTasks - 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 = -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 = -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 may return to the connection pool.
- Siebel database connection pooling cannot be used simultaneously with MTS or the Multiplexing features of Oracle 10g.

Configuring Pooling for Default Database Connections

Default database connections can be used by most AOM operations.

Configuring Parameters for Pooling Default Connections

This section 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 - 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 the AOM 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 AOM usage patterns, you may 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/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/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 - 1.
- To disable connection pooling, set MaxSharedDbConns and MinSharedDbConns to -1 (this is the default value).

MaxSharedDbConns and MinSharedDbConns are defined per AOM 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/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" on page 40.

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 AOM 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 AOM, 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 AOM's connection manager checks the connection pool for SQL statements. If this connection pool is empty, the connection manager adds a connection.

Each time a user initializes a 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 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, 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

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

Configuring Parameters for Pooling Specialized Connections

This section 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 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 -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 AOM component, on an enterprise basis (this parameter is included in named subsystems of type InfraDatasources). The database connections 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 AOM processes while MinTrxDbConns specifies the number of specialized database connections per AOM process. For details, see "Configuring Pooling for Default Database Connections" on page 38.

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" on page 39:

```
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 AOM, there would be a minimum of 25 specialized connections for this AOM.

How Pooled Specialized Connections Are Assigned

When the AOM starts up, the specialized connection pool is empty. When a request is made to start a transaction, the AOM requests a database connection from the specialized connection pool. If one is available, 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, 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 AOM terminates.

Using Thread Pooling for AOM

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

Using AOM thread pooling can improve performance and scalability on multiple-CPU machines that are under heavy load—for example, machines using eight or more CPUs that are running at more than 75% CPU capacity.

NOTE: AOM thread pooling is not recommended for smaller server machines or for machines that run under a relatively lower capacity.

About Thread Pooling for AOM

The pool size per multithreaded process for an AOM is determined by the combined settings of the parameters UseThreadPool, ThreadAffinity, MinPoolThreads, and MaxPoolThreads.

AOM thread pooling reduces some of 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.

NOTE: Using thread pooling introduces its own overhead, however, 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 UseThreadPool = TRUE, but ThreadAffinity = TRUE).

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

Configuring AOM Thread Pooling

To use thread pooling, you set the following parameters on the AOM:

- UseThreadPool = TRUE (default is FALSE)
- ThreadAffinity = FALSE (default is FALSE)
- MinPoolThreads = *min_number_threads_in_pool* (default is 0)

  where *min_number_threads_in_pool* represents the minimum number of threads in the AOM thread pool.
- MaxPoolThreads = MinPoolThreads (default is 0)

NOTE: You must specify a value for MaxPoolThreads that is equal to or greater than the value of MinPoolThreads. Other than this requirement, the specific value you provide does not matter.

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

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

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

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

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

```
MinPoolThreads = MaxPoolThreads = (525/5)/ 5 = 105/5 = 21
```

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

```
MinPoolThreads = MaxPoolThreads = (725/5)/ 5 = 145/5 = 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

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

For more information about the Siebel Server and AOM 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* for the operating system you are using
- *Siebel System Administration Guide*

Configuring SISNAPI Connection Pooling for AOM

This section provides information about how to manage SISNAPI connections for your Siebel Server.

SISNAPI (Siebel Internet Session application programming interface), a messaging format that runs on top of the TCP/IP protocol, is used for network communication between AOM and Siebel Web Server Extension (SWSE), installed on the Web server. SISNAPI connections can be configured to use encryption and authentication based on Secure Sockets Layer (SSL).

Each multithreaded process for an AOM component uses a pool of SISNAPI connections managed by the SWSE. 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:

- Web server process terminates
- AOM component terminates
- 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 an AOM is determined by the combined settings of MaxTasks, MaxMTServers, and SessPerSisnConn.

SessPerSisnConn determines how many sessions can be multiplexed over a single SISNAPI connection. By default, SessPerSisnConn is set to 20 for AOM components. This figure is suitable for most deployments and generally does not need to be changed. You may set this differently, depending on how you have calculated think time. For details, see "Performance Factors for AOM Deployments" on page 23.

For more information about configuring MaxTasks and MaxMTServers, see "Tuning AOM Components for CPU and Memory Utilization" on page 27.

The number of actual SISNAPI connections per multithreaded process for the AOM is determined by the following formula:

(MaxTasks/MaxMTServers)/SessPerSisnConn = *SISNAPI_conn_per_process*

where *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:

```
(600/5)/20 = 120/20 = 6
```

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

```
(540/5)/18 = 108/18 = 6
```

In each case, six SISNAPI connections will be created and pooled for each AOM multithreaded process, from each SWSE. Each Web server and SWSE may potentially have its own set of six connections, so the maximum total number of connections into an AOM process is six times the number of Web servers. 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 may need to modify how you define MaxTasks, MaxMTServers, and SessPerSisnConn.

Some Object Manager components are not AOM components. Configuration issues for such components may differ from that applicable to AOM components. For information about the EAI Object Manager, see Chapter 9, "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 an AOM to a batch component. SRBroker also handles requests between batch components. SRBroker is used whether the components run on the same machine or on different machines.

Server requests originating with an AOM component always go the SRBroker component to determine what to do with the request:

- If the destination component is running on the same Siebel Server, SRBroker passes the request to this component. If multiple instances of the destination component are running, 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, SRBroker passes the request to SRBroker running on another machine. If the destination component runs on multiple Siebel Servers, 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/MaxMTServers threads. MaxMTServers and MinMTServers should be kept at their default values of 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 may result in request failures. See the discussion of the HonorMaxTasks parameter in the following section for more information about how requests submitted to batch components may be handled. (HonorMaxTasks has no effect when set on the SRBRoker or Server Request Processor (SRProc) components.)

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

About HonorMaxTasks Parameter for Batch Components

By default, the HonorMaxTasks parameter for batch components, such as Workflow Process Manager, is set to FALSE (this setting is recommended). With this setting, if requests are routed by SRBroker to a batch component that has reached the maximum task capacity, the requests will be queued in memory, and processed 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.

You may consider setting HonorMaxTasks to TRUE for batch components in the following scenarios:

- For batch components handling asynchronous requests, consider changing HonorMaxTasks to TRUE if servers running these components have different resource levels and are therefore configured with different MaxTasks values for these components. In this case, larger servers would be forced to handle more requests. (However, if components are not running at maximum task capacity, this effect may be hard to observe.)
- If batch components are suffering from crash or hang issues, it may be undesirable to queue requests in component memory. If HonorMaxTasks is TRUE, success or failure status of each request will be correctly reported. (This optional usage is a temporary measure only. Work with Siebel Technical Services to resolve any component crash or hang issues.)

See also "Tuning Workflow Process Manager for Performance" on page 87.

5 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 includes the following topics:

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

- *Siebel Deployment Planning Guide*
- *Siebel Installation Guide* for the operating system you are using
- *Siebel System Administration Guide*
- *Siebel Security Guide*
- *System Requirements and Supported Platforms* on Siebel SupportWeb

The following sections in this book also relate to Siebel Web Client performance:

- For performance considerations for Application Object Manager (AOM), see Chapter 3, "Tuning the Siebel Application Object Manager for Performance."
- For performance considerations related to configuring Siebel applications, see Chapter 12, "Tuning Customer Configurations for Performance."

About Siebel Clients

A Siebel client is a computer that operates Siebel Business Applications, accessing data and services by way of one or more servers. The Siebel clients allow users to access information managed by Siebel applications. All Siebel deployments include one or more of the Siebel client types. You can deploy a mixture of clients.

The Siebel Business Applications client type covered in this book is the Siebel Web Client. This client runs in a standard third-party browser on the end user's client computer, and does not require any additional persistent software installed on the client.

Using HTTP, the browser connects to a Web server over a WAN, LAN, or VPN, or over the Internet. Through the Web server, the Siebel client connects to an Application Object Manager (AOM) component on a Siebel Server, which executes Siebel application business logic and accesses data. Data is accessed from the Siebel Database and may also be accessed from other data sources using virtual business components and a variety of integration methods.

Only the user interface layer of the Siebel Business Applications architecture resides on the client computer.

For more information about the Siebel Web Client and other client types, and about supported browsers and browser settings, see the *Siebel Installation Guide* for the operating system you are using and the *Siebel System Administration Guide*.

Performance Factors for Siebel Web Clients

Some performance considerations for Siebel applications involve processing or tuning activities on servers only, and do not affect Siebel client performance. However, many other such factors either directly or indirectly affect Siebel client performance. This chapter highlights some of the factors most directly related to the performance of the Siebel Web Client.

The performance of the Siebel client depends on many factors, some of which are summarized below. For additional information on these topics, refer to applicable documents on the *Siebel Bookshelf* or Siebel SupportWeb.

About Supporting Multiple Siebel Modules

Employee applications and customer applications have different requirements and characteristics and may use different browsers and other related technologies.

- Employee applications, such as Siebel Call Center, use high interactivity mode and run in supported Microsoft Internet Explorer browsers only.
- Customer applications, such as Siebel eService or Siebel eSales, use standard interactivity mode and may run in a wider range of browsers and browser versions.

All Siebel applications have many architectural elements in common. Multiple applications can use the same Siebel repository file (SRF). Each application uses its own AOM component. You may 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 or custom browser scripts. See "Following Configuration Guidelines" on page 54.

Client performance will also vary according to which Siebel modules 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 may affect performance.

About Local Machine Resources

The resources available on each user's local machine should meet or exceed the recommendations outlined in *System Requirements and Supported Platforms* on Siebel SupportWeb. Some performance enhancement measures depend directly on the available resources.

Best Practices for Siebel Web Client Tuning

You should 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 Chapter 12, "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 as 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 you perform to achieve performance and scalability goals may include the following:

- Adjusting your system topology
- Configuring the Siebel application in Siebel Tools
- Configuring Siebel Server components, particularly the AOM
- Adjusting hardware resources available on local machines
- Adjusting operating system settings on server or client machines
- Adjusting Web server settings or network settings
- Adjusting Web browser settings
- Setting user preferences for Siebel applications

Providing Sufficient Web Server and Network Capacity

Make sure that your Web server is appropriately configured to meet your performance requirements. See also "Specifying Static File Caching" on page 55.

Make sure that your network capacity (bandwidth) meets your performance requirements. Several 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 Web server to the client. If bandwidth is an issue, 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. For the high interactive client, the user can decide which columns are required in list applets. Rather than assuming a specific set, let users adjust it as necessary. Provide the minimal number of columns required in the base configuration.

  For more information, see Chapter 12, "Tuning Customer Configurations for Performance."

- **View layout caching.** In high interactivity mode, administrators can determine the number of views to be cached locally. If the hardware supports a greater number of views to be cached, adjust the value accordingly.

  When a view is cached, subsequent visits will require a data update, but the Web templates need not be reloaded. This provides a substantial improvement in overall usability.

  For more information, see "Improving Performance Using View Layout Caching" on page 57.

- **Login.** The first login is generally the most expensive operation for the high interactivity 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

Siebel Expert Services offers general guidance based on information known about the characteristics of the configured Siebel application. However, customer testing is advised, 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 you are testing with warm views (already visited and cached) if this is how the application will be used most of the time—if users log in and use the application for 4-8 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 time.

For more information about performance monitoring, see Chapter 14, "Monitoring Siebel Application Performance with Siebel ARM."

Providing Sufficient Client Hardware Resources

For best client performance for high interactivity applications, provide sufficient or generous hardware resources to your end users (typically, employees). Requirements may vary according to your deployment.

The more memory that is available on your client machines, the greater the number of views that can be cached. For more information, see:

- "Managing the Browser Cache" on page 54
- "Specifying Static File Caching" on page 55
- "Improving Performance Using View Layout Caching" on page 57

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

For best performance for the high interactivity client, which is used by employee applications like Siebel Call Center, it is generally recommended to include the latest supported version of Microsoft Internet Explorer in your testing. More recent versions often include fixes and performance enhancements.

For best performance for the standard interactivity client, which is used by customer applications like Siebel eService, you must determine the minimum capabilities of customer environments, such as browser to support, processor speed, or expected Internet connection speed. Customer applications must support a wide range of customer environments. Accordingly, you should generally minimize the complexity of such applications.

For Siebel client hardware and other platform requirements and recommendations, see *System Requirements and Supported Platforms* on Siebel SupportWeb.

For information about browser settings for Siebel applications, see *Siebel System Administration Guide*.

Tuning System Components

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

Most performance tuning involving Siebel Server components should focus on the AOM. For more information, see Chapter 3, "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 may 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, may improve both client performance and application scalability on Siebel Servers.

For more information about Siebel ARM, see Chapter 14, "Monitoring Siebel Application Performance with Siebel ARM."

Following Configuration Guidelines

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

Some application administration tasks may also affect application performance, and should also be carefully assessed.

Follow guidelines presented in Chapter 12, "Tuning Customer Configurations for Performance," or in other books on the *Siebel Bookshelf*.

Managing the Browser Cache

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

NOTE: When measuring performance, you should take into account view layout caching or other types of caching. For example, performance is better when a Siebel view layout is retrieved from a cache than it is when the view layout is not cached and must be retrieved from the system. For more information, see "Improving Performance Using View Layout Caching" on page 57.

Cache usage varies according to what browser is being used, what applications are running, and application settings. For example, high interactivity applications use the browser cache more than standard interactivity applications.

For high interactivity applications, it is generally recommended that users do not clear their browser cache, including when the browser is closed. The following settings for Microsoft Internet Explorer are recommended:

- Choose Tools > Internet Options. Click the Advanced tab. In the Security options, uncheck the setting Empty Temporary Internet Files Folder when browser is closed.

  NOTE: If you do not use the above setting, then persistent view layout caching and preloading, described in "Improving Performance Using View Layout Caching" on page 57, will not work.

- Choose Tools > Internet Options. Click the Advanced tab. In the Security options, uncheck the setting Do not save encrypted pages to disk.

  NOTE: If you do not use the above setting, then persistent view layout caching and preloading, described in "Improving Performance Using View Layout Caching" on page 57, will not work for encrypted views (views encrypted using SSL).

- Choose Tools > Internet Options. In the General tab, click Settings. For the option Check for newer versions of stored pages, use the setting Automatically.

- Browser caching is also subject to the size of the temporary Internet files folder. This setting is located in Tools > Internet Options. In the General tab, click Settings, then specify the amount of disk space to use for this folder.

  NOTE: Setting the size of the temporary Internet files folder to 0 disables persistent view layout caching and preloading, which are described in "Improving Performance Using View Layout Caching" on page 57.

For more information about browser settings for Siebel applications, see *Siebel System Administration Guide*.

Caching in the browser is also subject to Web server settings controlling static file caching. For details, see "Specifying Static File Caching" on page 55.

Specifying Static File Caching

Browser caching behavior is also subject to Web server 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.

Caching for Siebel Web template files is described in the persistent view caching section in "Improving Performance Using View Layout Caching" on page 57.

Because some static files may in fact be updated periodically, there is some risk that outdated versions of static files may be served from the cache. Therefore, some appropriate content expiration time should be specified. In general, setting an expiration time of 7 days may be appropriate.

If static files are rarely updated, you can specify a larger number, for less frequent expiration. If static files are updated more often, you can specify a smaller number, for more frequent expiration.

Instructions follow for specifying static file caching on Microsoft Internet Information Services (IIS), IBM HTTP Server (IHS), and Sun Java System Web Server. You must restart your Web server for the settings to take effect. For details, refer to your third-party Web server vendor documentation.

For more information about supported Web servers and versions, see *System Requirements and Supported Platforms* on Siebel SupportWeb.

Static File Caching for Microsoft IIS

For Microsoft IIS, follow the procedure below to specify static file caching and content expiration.

To specify static file caching on Microsoft IIS

1. On the Web server machine, choose Start > Settings > Control Panel > Administrative Tools.
2. Run Internet Service Manager.
3. In Internet Service Manager, right-click Default Web Site.
4. In Default Web Site Properties, click the HTTP Headers tab.

5 Check the Enable Content Expiration check box.

6 Select Expire After, and specify the value of 7 (to expire static files after 7 days), or another value appropriate for your deployment.

Static File Caching for IBM HTTP Server

For IBM HTTP Server (IHS), follow the procedure below to specify static file caching and content expiration.

To specify static file caching on IBM HTTP Server

1 On the Web server machine, open the file httpd.conf for editing. This file is located in the Web server installation directory.

2 Verify that the following line is included and not commented out:

```
LoadModule expires_module modules/mod_expires.so
```

3 Add the following lines, if not already present, to the file (below the line shown in Step 2). Or, instead of 7 days, specify another value appropriate for your deployment.

```
###################################################################
ExpiresActive On
<IfModule mod_expires.c>
ExpiresByType image/gif                  "access plus 7 days"
ExpiresByType image/jpeg                 "access plus 7 days"
ExpiresByType application/x-javascript   "access plus 7 days"
ExpiresByType text/css                   "access plus 7 days"
</IfModule>
###################################################################
```

4 Save the file.

Static File Caching for Sun Java System Web Server

For Sun Java System Web Server, follow the appropriate procedure to specify static file caching and content expiration. For example, for Sun Java System Web Server 6.0, follow the steps below.

To specify static file caching on Sun Java System Web Server

1 From a browser, connect to the Web server administration page (for example, http://*web_server_name*/8080).

2 Select the server, and click Manage.

3. Click the link for Class Manager, in the upper right area.
4. Among the horizontal tabs at the top, click Content Mgmt.
5. Click the link for Cache Control Directives, in the left tab area.
6. Under Cache Control Response Directives, select Maximum Age (sec), and input 604800 (seconds) for a valid cache of 7 days.
7. Click Apply to apply the change.

Improving Performance Using View Layout Caching

View layout caching in the browser (also referred to as *layout caching* or *view caching*) improves the performance of accessing views in a high interactivity application. It speeds up the rendering of views in a Siebel application session by caching the following on the browser:

- Static HTML (from the templates) used for interpreting the view.
- Dynamic HTML generated on the client for rendering controls.

Appropriate caching settings can optimize client performance and network utilization for Siebel client sessions. Caching behavior is subject to considerations described under "Managing the Browser Cache" on page 54.

Two kinds of view layout caching are used for the Siebel Web Client. These types of caching work together and should be configured as a system.

- **Caching in browser memory.** For details, see "View Layout Caching in Memory" on page 57.
- **Persistent caching (in the browser cache directory on the local disk).** For details, see "Persistent View Layout Caching" on page 59.

NOTE: Whether views can be cached depends on the underlying requirements described in "Determining If Views Are Available for Layout Caching" on page 61.

View Layout Caching in Memory

View layout caching in memory creates multiple HTML frames on a browser to store the layout for a view. The number of these frames represents the view cache size. When a view is displayed, the HTML frame containing the layout for that view will be sized to occupy all (100%) of the available browser space, while the other frames will be hidden (that is, sized to occupy 0% of the space).

For information on setting the view cache size, see "Setting the View Cache Size" on page 58.

View layout caching uses the following logic:

- If a user navigates to a view whose layout is already available in the browser memory cache, the HTML frame containing that view will be made visible and the currently visible frame will be hidden.

- If a user navigates to a view whose layout is not in the browser memory cache, one of the available HTML frames will be used to load the layout of the view into memory. The view layout will be loaded from the persistent cache, if possible. The view layout in memory will be cached subject to the View Cache Size setting.
- If the view layout is not currently stored in the persistent cache, then it is loaded from the server. It will also be stored in the persistent cache. The view layout in memory will be cached subject to the View Cache Size setting.

For more information, see "Persistent View Layout Caching" on page 59.

NOTE: The high interactivity framework separates the retrieval of the Siebel application user interface from the server and the retrieval of database records. Database records to be displayed in views are always retrieved from the server.

The memory cache contains the layouts of views that the user has visited and that are available for view caching. When the view cache is full and another view is visited, the first view visited is removed from the cache. The memory cache contents are thus managed on an LRU or least recently used basis.

The HTML frames are loaded into memory when a user navigates to the view. To cache a startup view (one that is cacheable), the user must visit the view twice—that is, visit it a second time after visiting another view.

NOTE: Views specifically created as home page views, such as Home Page View (WCC) for Siebel Call Center, are standard interactivity views and are not cacheable.

The view caching framework is designed so that if the frames containing cached views are deleted (for example, by performing a browser refresh, which removes any previously cached views), the framework begins reloading the layout cache, starting with the next cacheable view the user visits.

At startup, view layouts for recently visited views may be preloaded into the memory cache from the persistent browser cache on disk. This behavior is specified using the parameter ViewPreloadSize. For more information, see "Persistent View Layout Caching" on page 59 and "Preloading Cached Views into Memory" on page 59.

Setting the View Cache Size

For browser memory caching, the size of the view layout cache is controlled by the View Cache Size user preference setting for each user, as described below.

NOTE: Setting View Cache Size to 1 turns off view caching. This has the same effect as setting the EnableViewCache parameter to FALSE, as described in "Disabling View Layout Caching" on page 60.

To set the size of the view layout cache

1. From the application-level menu, choose View > User Preferences.
2. From the Show drop-down list, choose Behavior.
3. In the View Cache Size field, select a value from the drop-down list, or type in a value.

The default value for View Cache Size is 10. This value specifies that 10 HTML frames are cached in memory to represent Siebel view layouts. One of these frames is displayed at any one time.

- Using a figure that is too low may not provide enough caching if your users access many views and client machines have sufficient memory.
- Using a figure that is too high may impair performance by using more memory than is available on the machine.

Persistent View Layout Caching

Persistent layout caching stores the layout of certain views in a local client's browser cache on disk. The stored layout is then reused for subsequent visits to this view, in the same session or in a subsequent session. (For subsequent visits in the same session, the view layout is accessed from the browser memory cache, if available.)

Persistent view layout caching helps improve performance by reducing the number of pages that have to be generated from the server from session to session.

The parameter WebTemplateVersion determines whether the Siebel Web Engine will use a view layout stored in the browser's cache or build a new view layout. This parameter is located in the [InfraUIFramework] section of the application configuration file, such as uagent.cfg for Siebel Call Center. This file is located on the Siebel Server machine (running AOM).

When you modify Web templates for Siebel views, add the WebTemplateVersion parameter to the configuration file (if not already present), and set its value to 1. For example:

```
[InfraUIFramework]
WebTemplateVersion = 1
```

Subsequently, each time you change any of the Web templates, increment the value of the parameter by 1. Doing so forces loading view layouts from the Web templates on the server.

When a view is requested, the Siebel Web Engine includes in the URL a checksum value that encapsulates the value of the WebTemplateVersion parameter.

- If the parameter value and the value encapsulated in the URL match, then it is assumed that the view layout for this view has not been updated. If it is available, the view layout stored in the persistent cache can be used.
- If no match is found, then a new view layout is loaded from the server. The Web template on the server is presumably more current than the view layout stored in the browser's persistent cache.

Preloading Cached Views into Memory

For recently visited views, view layouts that are cached in the persistent cache on the browser may be preloaded into browser memory when the user logs in. The number of views that can be preloaded depends on the content of the persistent cache and is limited by the setting of the View Cache Size setting for each user.

For better performance at login time, it may be helpful to further limit the number of view layouts that are preloaded into memory during startup. To do this, use the parameter ViewPreloadSize.

NOTE: ViewPreloadSize only affects a user session when it is set to a positive integer value lower than the View Cache Size value. If the parameter is not set, the default behavior is to preload the number of view layouts corresponding to the View Cache Size value, minus one. (One of the frames specified using View Cache Size is reserved for the application startup view, where applicable.)

ViewPreloadSize should be added to the [InfraUIFramework] section of the application configuration file, such as uagent.cfg for Siebel Call Center. This file is located on the Siebel Server machine (running AOM). For example:

```
[InfraUIFramework]
ViewPreloadSize = 5
```

If ViewPreloadSize is set to 0, then no view layouts are preloaded into memory. In scenarios where users frequently log into the Siebel application, such as to access a single view, then log out again, login performance may be more important than precaching multiple views. In this case, you may choose to set this parameter to 0.

Disabling View Layout Caching

You can disable browser memory caching of view layouts for your application users by changing the parameter EnableViewCache to FALSE in the [InfraUIFramework] section of the application configuration file, such as uagent.cfg for Siebel Call Center. For example:

```
[InfraUIFramework]
EnableViewCache = FALSE
```

NOTE: In general, setting EnableViewCache to TRUE is recommended. If some users do not need view layout caching, they can set View Cache Size to 1, as described in "Setting the View Cache Size" on page 58.

Setting EnableViewCache to FALSE disables browser memory view layout caching only. It does not disable persistent view layout caching.

Determining How the Current View Layout Was Loaded

If you are running an application and want to determine how the current view was retrieved, go to the view, press SHIFT, and choose Help > About View. The Cache Mode identified for the current view indicates how the application retrieved the view layout. Possible values include:

- **Not Cached.** The view layout was not cached (and cannot be cached).
- **Memory.** The view layout was retrieved from the browser memory cache.
- **Server.** The view layout was retrieved from the Siebel Server and the Web server. If the view is cacheable, and you visit another view and then return to this one, the Cache Mode value changes to Memory.

- **Disk.** The view layout was retrieved from the browser disk cache (persistent caching). If the view is cacheable, and you visit another view and then return to this one, the Cache Mode value changes to Memory.

  The longer you go without clearing the cache, the more likely that a rarely visited view will be retrieved from the persistent cache on the browser, rather than from the server.

Determining If Views Are Available for Layout Caching

Not all Siebel views are available for layout caching. Views that contain applets that have dynamic layouts or controls that are data-dependent cannot be cached. Only applets that support high interactivity are available for view layout caching.

Layout caching is a feature of the C++ class that implements an applet. The ability to be cached is determined by a property of each applet's class object definition. Using Siebel Tools, check the value of the High Interactivity Enabled property of a class object definition to determine whether applets for this class support layout caching. For a view to be available for caching, the class objects for *all* of the applets in the view must have High Interactivity Enabled values of 2 or 4 (available for caching).

For detailed information about settings for the High Interactivity Enabled property for a class, see *Siebel Object Types Reference*.

View layout caching is also disabled for a view in the following cases:

- If personalization rules are defined for any of the applets
- If any of the applets are dynamic toggle applets
- If any of the applets are hierarchical list applets or explorer (tree) applets
- If HTML frames are used within the view template (for example, for explorer views)

Managing Performance Related to Message Bar

Employee applications such as Siebel Call Center include a message bar feature. The message bar requires network resources and local resources on the client machine to continually update the displayed text.

- If your deployment does not require it, turn off the message bar feature to save processing resources.
- If some of your users require the message bar, you can specify that users will be able to turn it off from Tools > User Preferences > Message Broadcasting.

For more information about message broadcasting using the message bar, see *Siebel Applications Administration Guide*.

Configuring the Busy Cursor for Standard Interactivity Applications

When the parameter EnableSIBusyCursor is set to TRUE (default), the Siebel application cannot accept new requests until it has finished serving the current request. An example of a new request is a user's attempt to drill down on a record.

The default setting of this parameter helps prevent the occurrence of JavaScript errors as the user cannot click on another link while a page is loading. Setting EnableSIBusyCursor to FALSE may improve network bandwidth usage for Siebel applications that use standard interactivity mode. It disables the appearance of the hourglass icon and allows the user to click on other links before the current request has been served.

You set the value for EnableSIBusyCursor in the [InfraUIFramework] section of your application configuration file.

6 Tuning Siebel Communications Server for Performance

This chapter describes some issues that affect the performance and throughput of selected functionality for Siebel Communications Server and related modules, and provides guidelines for tuning these modules to achieve and maintain optimal performance and scalability. It contains the following topics:

Functionality covered in this chapter includes session communications (typically, Siebel CTI) and Siebel Email Response. Other communications-related modules are not covered.

For more information about topics in this chapter, see the following documents on the *Siebel Bookshelf*:

- *Siebel Communications Server Administration Guide*
- *Siebel Email Response Administration Guide*
- *Siebel System Administration Guide*
- *Siebel Smart Answer Administration 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" on page 64 and subsequent sections.

For Siebel Email Response performance tuning information, see "Siebel Email Response Infrastructure" on page 74 and subsequent sections.

- **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, such as Siebel CTI Connect or third-party products
  - Receive inbound email messages (for Siebel Email Response)
- **Inbound communications.** Supports integrating with third-party email servers and processing inbound email (when using Siebel Email Response).
- **Outbound communications.** Supports integrating to 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 Communications Server 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 section and in:

- "Performance Factors for Session Communications" on page 66
- "Topology Considerations for Session Communications" on page 67
- "Best Practices for Session Communications Tuning" on page 68

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.

- **Application Object Manager (AOM).** 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 AOM.

  For more information about AOM, see Chapter 3, "Tuning the Siebel Application Object Manager for Performance."

- **Server Request Broker (SRBroker).** This server component handles communications between the AOM and certain other Siebel Server components, including CommSessionMgr.

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

  SRBroker is used whether CommSessionMgr runs on the same machine as the AOM, or on a different machine. For more information about such scenarios, see "Topology Considerations for Session Communications" on page 67.

  For more information about SRBroker, see "Tuning Server Request Broker (SRBroker)" on page 47.

Other Siebel Server Components

You may 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 details, see "Siebel Email Response Infrastructure" on page 74.
- **Communications Inbound Processor (CommInboundProcessor).** For details, see "Siebel Email Response Infrastructure" on page 74.
- **Communications Outbound Manager (CommOutboundMgr).** Sends outbound email or other types of messages.

Siebel Product Modules

In addition to Siebel CTI or Siebel Email Response, you may be using the following Siebel product modules for session communications:

- **Siebel CTI Connect.** This module consists of CTI middleware, communications driver, and sample communications configuration data. Siebel CTI Connect is based on third-party CTI middleware—Intel NetMerge, formerly Dialogic CT Connect. For Siebel CTI Connect, consult Intel documentation provided on the *Siebel Business Applications Third-Party Bookshelf*.
- **Siebel Smart Answer.** This module analyzes the content of email and search requests and returns an automatic response or suggests one or more responses to the user for approval.

  Siebel Smart Answer is based on third-party products from Banter. See *Siebel Smart Answer Administration Guide* and consult Banter documentation provided on the *Siebel Business Applications Third-Party Bookshelf*.

  For more information, see "Performance for Siebel Smart Answer" on page 78.

Third-Party Product Modules

You may be using third-party product modules—for example, CTI middleware, driver, and configuration; routing products; predictive dialers; interactive voice response modules; email servers; fax servers; and so on. For information about supported email servers, see *System Requirements and Supported Platforms* on Siebel SupportWeb.

NOTE: If you are not using Siebel CTI Connect, then, to use Siebel CTI, you must obtain a third-party CTI middleware package and work with your vendor to integrate this module.

Performance Factors for Session Communications

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

Depending on your deployment, your agents may be handling phone calls (Siebel CTI), email messages (Siebel Email Response), 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 Siebel Server. Think time is an important factor in overall system load. Estimation of think time should approximate actual user usage.

  For more information about think time and AOM tuning, see Chapter 3, "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 AOM.

  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 may be defined as agents but may receive only escalated work items, for example.

  For more information about concurrent users and AOM tuning, see Chapter 3, "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 should be realistic and the database needs to 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 may place limitations on the number of agents that may be served at a single contact center site.
- Integration with ACD queues, predictive dialers, or other modules may affect your configurations, affect network traffic, or have other impacts.
- The capacity of your telephony link (between the ACD switch and the CTI middleware) may affect performance.

Topology Considerations for Session Communications

Generally, Siebel Communications Server components for session communications, such as CommSessionMgr, should be run on the same Siebel Server machines as those running AOMs. In some cases, however, you must run CommSessionMgr on a different machine than the AOMs. These options are described in detail below.

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

Running CommSessionMgr on AOM Machines

Generally, Siebel Communications Server components for session communications should be run on the same Siebel Server machines as those running AOMs. Such a topology allows the AOM 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 machines.

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

Running CommSessionMgr on Dedicated Machines

Sometimes you *must* run CommSessionMgr on a different machine than the AOM components.

CommSessionMgr must run on the same machine where the communications driver for your CTI middleware is running. If your driver requires a particular operating system platform, then you must install Siebel Server and run CommSessionMgr on a machine of this platform. (Communications drivers are required to be able to run on one of the supported Siebel Server platforms, as described in *System Requirements and Supported Platforms* on Siebel SupportWeb.)

If your AOM components (Call Center Object Manager) run on machines using a different platform, then you set several parameters in the communications configuration, including CommSessionMgr and RequestServer, in order to designate the machine where CommSessionMgr is running. All communications session requests from an AOM supporting users for this communications configuration will be routed to the CommSessionMgr component on the dedicated machine.

For related information, see "Tuning the CommSessionMgr Component" on page 69. For more information about these parameters, see *Siebel Communications Server Administration Guide*.

Best Practices for Session Communications Tuning

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

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

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

- Adjusting your system topology. For more information, see "Topology Considerations for Session Communications" on page 67.
- Configuring the AOM component. For more information, see "Tuning the AOM Component" on page 69.
- Configuring CommSessionMgr and related components. For more information, see "Tuning the CommSessionMgr Component" on page 69.
- Modifying communications configurations, communications driver settings, and so on. Many of the activities described in the sections 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 may be required to run more AOM components and CommSessionMgr components to support the increase in volume of communications and in number of users.

Tuning the AOM Component

CommSessionMgr and CommConfigMgr components use a small percentage of the resources of the Siebel Server on which it runs. AOM performance has the greatest effect on overall system performance, even when CommSessionMgr or CommConfigMgr components are present.

AOM memory requirements for agent sessions depend on many factors. AOM 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).

AOM tuning also depends on your communications configuration caching methods. See also "Conserving AOM Server Resources Through Caching" on page 69.

For more information about AOM tuning, see Chapter 3, "Tuning the Siebel Application Object Manager for Performance."

Tuning the CommSessionMgr Component

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 AOM machine.

If you use a dedicated Siebel Server machine to run the CommSessionMgr component, then it may 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" on page 67.

Conserving AOM Server Resources Through Caching

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 AOM.

These caching mechanisms may be used together or separately. For more information, see *Siebel Communications Server Administration Guide*.

- **CommConfigCache parameter (AOM).** Setting the CommConfigCache parameter on the AOM to TRUE caches communications configuration data when the first agent logs in. Configuration data is cached until the AOM is restarted. For agents associated with the same communications configuration, each agent session uses the same cached data. See also "Tuning the AOM Component" on page 69.
  - Performance is improved for subsequent agent logins, because the configuration data is loaded from the cache rather than from the database.
  - AOM scalability is also improved because configuration data is shared in AOM memory across agent sessions, therefore conserving server resources even as the number of agent sessions increases.

- **CommConfigMgr server component and CommConfigManager parameter (AOM).** The CommConfigMgr server component caches communications configuration data when the first agent logs in. Setting the CommConfigManager parameter on the AOM 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 AOM component.
  - Although it is not required to use the CommConfigMgr component in conjunction with the CommConfigCache parameter for AOM, if you use them together, communications configuration data gets cached at the enterprise level rather than only for the AOM. Overall performance may be enhanced compared to using each of these mechanisms separately.

Improving Performance for Communications Configurations

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 you define, using the parameter DialingFilter.Rule*N*, may be appropriate for agents at a specific place, such as within the same country or area code. Other dialing filters may be suitable for a different set of agents.

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

It may 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 AOM memory or CPU.

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

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

Configuring Logging for Session Communications

Logging data may be analyzed as part of performance monitoring or tuning, as described in Chapter 14, "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 should be reduced to improve performance.

Log-related parameters applicable to session communications are summarized below. The AOM 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 AOM 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 details on these logging parameters, see *Siebel Communications Server Administration Guide*.

AOM Logging Parameters

AOM 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 should contain extra detail. Setting to FALSE provides better performance.
- **CommMaxLogKB.** Specifies the maximum size of log files.
- **CommReleaseLogHandle.** Specifies that the log file handle should 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 should contain extra detail. Setting to FALSE provides better performance.
- **MaxLogKB.** Specifies the maximum size of log files.
- **ReleaseLogHandle.** Specifies that the log file handle should be released periodically. The default setting of TRUE provides better performance.

Siebel CTI Connect Driver Logging Parameters

Siebel CTI Connect communications driver parameters that log session communications activity include:

- **Driver:DriverLogFile.** Specifies the name of the log file (ctc.log by default). A single log file is created for the driver session (for all users). Siebel CTI Connect events are also logged.

- **Service:ServiceLogFile.** Specifies the name of the log file (ctc_{@Username}.log by default). A separate log file is created for each agent session, in the form ctc_*username*.log. Siebel CTI Connect events for each agent session are also logged.
- **LogDebug.** Specifies whether log files should contain extra detail. Setting to FALSE provides better performance.
- **MaxLogKB.** Specifies the maximum size of log files.
- **ReleaseLogHandle.** Specifies that the log file handle should be released periodically. The default setting of TRUE provides better performance.

Improving Availability for Session Connections

When agents log into the Siebel application after experiencing browser failure or a dropped connection, session communications may 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 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 may 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.

  NOTE: If you are using the Push Keep Alive driver, you *must* also use the ChannelCleanupTimer parameter.

- **CommMaxMsgQ and CommReqTimeout parameters (AOM).** In addition to setting general application timeouts, setting the AOM 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 machine 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 Communications Server Administration Guide*.

Improving Screen Pop Performance

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 may 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" on page 70.

For more information about Siebel Web Client response time, see Chapter 5, "Tuning Siebel Web Client for Performance."

Improving Screen Pop Performance for Siebel CTI Connect

If you are using Siebel CTI Connect, you can use another method to improve screen pop performance. For general considerations, see "Improving Screen Pop Performance" on page 73.

A screen pop triggered by an agent answering a call generally involves a small lag time from when the agent answers the call to when the CTI middleware connects the agent with the caller. For Siebel CTI Connect, you can reduce this lag time.

The Siebel CTI Connect communications driver for Siebel CTI Connect includes the EventAnswerCall device event. This event is triggered whenever the AnswerCall device command is invoked by the Siebel CTI Connect driver—typically, when an agent answers a call using the communications toolbar. The EventAnswerCall event occurs before Siebel CTI Connect sends the corresponding TpAnswered event, and thus provides extra time in which to generate the screen pop.

To use this feature, create an event handler definition based on the EventAnswerCall device event. This event handler invokes an event response that generates a screen pop and invokes the appropriate event logs.

For more information, including examples, see *Siebel Communications Server Administration Guide*.

Reviewing Performance Impact of Activity Creation

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 section and in:

- "Performance Factors for Siebel Email Response" on page 75
- "Topology Considerations for Siebel Email Response" on page 76
- "Best Practices for Siebel Email Response Tuning" on page 76

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 may be using the following Siebel components or modules:

- **Siebel Smart Answer.** This module analyzes the content of email and search requests and returns an automatic response or suggests one or more responses to the user for approval.

  Siebel Smart Answer is based on third-party products from Banter. See *Siebel Smart Answer Administration Guide* and consult Banter documentation provided on *Siebel Business Applications Third-Party Bookshelf*.

  For more information, see "Performance for Siebel Smart Answer" on page 78.

- **Siebel Assignment Manager.** This module may be used 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 supported email servers, see *System Requirements and Supported Platforms* on Siebel SupportWeb.

Performance Factors for Siebel Email Response

This section 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 may 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.

  If you are deploying Siebel Smart Answer, you must also consider the size of the knowledge base.

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. Using Siebel Smart Answer, especially for auto-response capabilities, reduces the number of agents needed to handle incoming email and reduces corresponding demand on session-related computing resources such as AOM or CommSessionMgr.

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 machine.

If inbound message volume warrants it and if multiple server machines are available to run CommInboundRcvr, CommInboundProcessor, and other components, then you should consider running CommInboundRcvr and CommInboundProcessor on separate machines (or machines) from other Communications Server components. Topology options for these component are different for real-time and nonreal-time processing.

For more information about CommInboundRcvr and CommInboundProcessor, see *Siebel Communications Server Administration Guide* and *Siebel Email Response Administration Guide*.

CommOutboundMgr and Siebel Smart Answer (Smart Answer Manager) may be run together on a different machine (or machines), as appropriate.

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.

Best Practices for Siebel Email Response Tuning

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

Review information presented in *Siebel Email Response Administration Guide*, *Siebel Communications Server Administration Guide*, *Siebel Smart Answer 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 machine, 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 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 may 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 should be defragmented regularly.

For more information about email processing directories, see *Siebel Communications Server Administration Guide* and *Siebel Email Response 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 should generally run on the same Siebel Server machines 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 should be defragmented regularly.

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

Configuring Logging for Siebel Email Response

Logging data may be analyzed as part of performance monitoring or tuning, as described in Chapter 14, "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 should be reduced to improve performance.

An applicable parameter for the Internet SMTP/POP3 Server communications driver is LogDebug. For details, see *Siebel Communications Server Administration Guide*.

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

Performance for Siebel Smart Answer

Siebel Smart Answer analyzes the content of email and search requests and returns an automatic response or suggests one or more responses to the user for approval. Smart Answer has an internal AI (artificial intelligence) engine that reads inbound message content and determines the nature (category) of the message.

Key performance factors to consider are the following:

- **Complexity of the inbound message.** If inbound messages are complex or large in size, then Smart Answer will have to process more text. This will impact Smart Answer performance. Therefore, if the format of inbound messages is subject to your control, consider that smaller or simpler messages will allow Smart Answer to perform better.
- **The number of categories in the knowledge base (KB) file.** As the number of categories increases, Smart Answer has to look through more data to determine a category. It is recommended to keep a reasonable number of categories in the KB file.
- **Whether Smart Answer runs in standalone or master/slave mode.** Smart Answer supports a multiserver mode where several instances of Smart Answer can be running at the same time across multiple servers. However, one node is designated as a master node that "learns" from the email it reads and provides feedback to the KB. Smart Answer in slave mode, however, simply processes the email messages without providing feedback to the KB.
- **Number of Smart Answer instances.** By default, MaxMTServers is set to 1, which should be enough for most deployments.
- **Placement of Smart Answer relative to CommInboundRcvr or CommInboundProcessor.** Both Smart Answer and CommInboundRcvr or CommInboundProcessor process the text of inbound email messages, and both take up significant server resources. Therefore, as inbound email volume, the number of categories, or the complexity of inbound messages increases, you may want to consider running CommInboundRcvr, CommInboundProcessor, and Smart Answer on separate physical servers.

7 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:

For more information on 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 Policies and Workflow Processes 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 the policy conditions for a workflow policy are true, then the policy action occurs when all 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" on page 80.
- **Workflow Agent trace logs.** For details, see "Using Workflow Agent Trace Logs" on page 80.
- **Workflow Policies tables.** For details, see "Monitoring Workflow Policies Tables" on page 81.

Using the Policy Frequency Analysis View

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 the policies executed, as evidenced by a Workflow Monitor Agent process. The policy maker can monitor Workflow Agent process activity to determine if the current policies are adequate, if new policies need to be created, or if 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 Logs

Workflow Agent trace logs include the following:

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

- **Workflow Action Agent task log.** 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 logs.**
  - 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

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 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 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, 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, 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_ACTN_REQ, and S_ESCL_STATE tables by adjusting parameters related to storage, access, and caching. Refer to the database documentation for additional information on properly adjusting such parameters. Also, make sure 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

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, 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

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, you may 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 may 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

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

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 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 may be required is in the case of multiple users who may 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, 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, you may 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, investigate what other processes may be contending for CPU resources on the Siebel Server. You may 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

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, 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 sections:

- "Minimizing Usage of Parameter Search Specification" on page 85
- "Monitoring Conditions Based on Parent and Child Business Components" on page 85
- "Configuring Siebel Business Applications for Workflow Performance" on page 85
- "Monitoring Memory Overhead for Workflow Processes" on page 86

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

Minimizing Usage of Parameter Search Specification

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, make sure that all 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

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" on page 156.

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 Business Applications for Workflow Performance

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

Assume, for example, 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 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

Overhead and performance and scalability characteristics varies depending on whether you are running workflows locally in the AOM 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 AOM

A workflow instance—that is, one run of a workflow definition—can run within an AOM. 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, the definition would run in that user thread.

In this mode, Workflow adds a fixed overhead (100–200 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 the 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 AOM where the workflow is invoked, or run it on dedicated Siebel Server machines. Compared to running workflows locally, running workflows in WfProcMgr may reduce performance, but improve scalability. Running WfProcMgr on dedicated Siebel Servers typically provides the best scalability, while colocating WfProcMgr and AOM may 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 following pros and cons:

PROS

- 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.

CONS

- On error, you must look at the log files because there is no automatic notification.
- The SRBroker 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 section provides general approaches to tune and optimize performance of Workflow Process Manager.

It is imperative to remember that every implementation of Siebel applications 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 sections:

- "Caching Business Services" on page 87
- "Caching Sessions" on page 88

NOTE: The information provided in this section should be considered general background information. No attempt is made to detail the many variables that affect tuning at specific sites. This section is not a substitute for specific tuning recommendations made by Siebel Technical Services.

Caching Business Services

Business services invoked through Workflow Process Manager should 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 should *not* be reset to TRUE.

Caching Sessions

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 AOM 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; the other connection for read-only operations).

The default value is 10. A value of 0 disables this parameter. The maximum value is 100. In general, you should 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 Configurator for Performance

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

Siebel Configurator provides product configuration and solution-computing capabilities, and can be deployed as a server-based or browser-based module.

NOTE: This chapter covers Siebel Configurator server-based deployments only. For additional information, see *Siebel Product Administration Guide*.

Siebel Configurator is one of the Siebel Order Management modules. These modules work together to support various phases in conducting commerce, including online selling.

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

- *Siebel Product Administration Guide*
- *Siebel System Administration Guide*

Also see documents for related Siebel Order Management modules:

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

Siebel Configurator Infrastructure

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

- **Application Object Manager (AOM).** Siebel Configurator functions may be performed within the AOM, such as Call Center Object Manager (alias SCCObjMgr_enu in a U.S. English environment) for Siebel Call Center.

- **Siebel Product Configuration Object Manager (alias eProdCfgObjMgr_jpn in a Japanese locale).** An optional component, suitable for some Siebel Configurator deployments, that processes configuration requests for user sessions submitted from an AOM component. Typically, this component is run on a separate Siebel Server machine than the one running the AOM. Multiple instances of this component can be run on the separate Siebel Server machine where it is possible to distribute requests across the various instances. For more information, see "Topology Considerations for Siebel Configurator" on page 91.

  NOTE: The three letter extension to the alias of the Siebel Product Configuration Object Manager (jpn in our example of eProdCfgObjMgr_jpn) corresponds to the value for the Locale Code parameter (alias LocaleCode) associated with this Application Object Manager. For more information about the Locale Code parameter, see the *Siebel Global Deployment Guide*.

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

Performance Factors for Siebel Configurator

In planning Siebel Configurator server-based deployments, or in troubleshooting performance for existing deployments, you must consider several key factors that determine or influence performance.

Subsequent sections 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 Configurator returns a response such as an update to the customizable product or a conflict message.

The factors below, 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 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 AOM.

  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 may access multiple models, however, each of which must be separately cached.

Topology Considerations for Siebel Configurator

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

- Running Siebel Configurator in the AOM component.

  For more information, see "Running Siebel Configurator in the AOM Component" on page 91.

- Running Siebel Configurator on one or more dedicated Siebel Servers. (Such servers are sometimes referred to as remote servers, because they are remote to the machine on which AOM is running. In general, this section uses the term dedicated servers.)

  For more information, see "Running Siebel Configurator on Dedicated Servers" on page 92

The optimal deployment approach for Siebel Configurator, and the optimal number of server machines you require for this module, depends on factors such as those described in "Performance Factors for Siebel Configurator" on page 90.

Running Siebel Configurator in the AOM Component

You can run Siebel Configurator in the AOM 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 may yield reasonable performance and make the most effective use of your hardware resources.

With this option, you set all parameters for managing Siebel Configurator caching on each applicable AOM. For details, see "About Siebel Configurator Caching" on page 96.

Running Siebel Configurator on Dedicated Servers

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

Possible variations on this general topology option include:

- Running one eProdCfgObjMgr component with one AOM component
- Running multiple eProdCfgObjMgr components with one AOM component
- Running one eProdCfgObjMgr component with multiple AOM 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) may yield the best performance and make the most effective use of your hardware resources.

With this option, you set some parameters for managing Siebel Configurator caching on each applicable AOM, and some on each applicable dedicated Siebel Configurator server. For details, see "About Siebel Configurator Caching" on page 96.

Configuring AOM for Dedicated Siebel Configurator Deployments

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

The AOM forwards each configuration session request to the dedicated Siebel Configurator server with the fewest concurrent users.

Table 3 lists server parameters for managing dedicated Siebel Configurator deployments. Using Server Manager, set these parameters on each AOM (do not set them on the dedicated Siebel Configurator server machine).

Table 3. Server Parameters for Dedicated Siebel Configurator Server Deployment

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 Configurator deployments using AOM only.

Table 3. Server Parameters for Dedicated Siebel Configurator Server Deployment

Parameter Name	Display Name	Data Type	Default Value	Description
eProdCfgServer	Product Configurator-Remote Server Name	Text		When you have not enabled explicit product mapping for products to a Siebel Configurator server, set this parameter to the names of the dedicated machines 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 milliseconds, that the AOM tries to connect to a dedicated Siebel Server running eProdCfgObjMgr. After the timeout has been reached, an error is returned to the user.

Best Practices for Siebel Configurator Tuning

Using your hardware resources optimally, and configuring your system appropriately, can help you to achieve your performance goals. You should 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 you perform to achieve performance and scalability goals may include:

- Adjusting your system topology. For more information, see "Topology Considerations for Siebel Configurator" on page 91.
- Configuring Siebel Server server components for Siebel Configurator.
- Designing and deploying your customizable product models. For more information, see "Defining Customizable Product Models and Classes" on page 95.

This section applies to deployments using Siebel Web Client. It includes the following topics:

Tuning Siebel Configurator

How you configure your Siebel Server components for Siebel Configurator server deployments, for appropriate tuning, depends in part upon which deployment method you use, as described in "Topology Considerations for Siebel Configurator" on page 91.

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

In particular, note that, for a dedicated Siebel Configurator server, the MaxTasks parameter should generally be set much lower than for an AOM. By default, the ratio of MaxTasks to MaxMTServers is 20:1 for eProdCfgObjMgr.

In addition, depending on request load, MaxTasks should generally be set lower for an AOM running Siebel Configurator than for an AOM that is *not* running Siebel 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 Configurator. For example, for every 100 users, 60 work with Quotes.
- Calculate what percentage of time these users spend using Siebel Configurator. For example, out of the 60 users mentioned previously, only 30 are concurrently using Siebel Configurator.
- Maintain the default ratio of 20:1 for MaxTasks/MaxMTServers.

If you deploy Siebel Configurator using eProdCfgObjMgr on a dedicated Siebel Server machine and database connection (login and log out) is slow, 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 - 1.

  This pools all user connections without sharing and avoids the creation and deletion of a new database connection for each eProdCfgObjMgr session.

- 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 the *Siebel Security Guide*.

For more information about database connection pooling, see "Configuring Database Connection Pooling for AOMs" on page 34.

Specifying the Siebel Configurator File System Location

Siebel Configurator uses a file system directory to cache all configuration related object definitions. The server parameter, Product Configurator - FS location (alias eProdCfgCacheFS), specifies the location. Specify a value for this parameter to reference a server directory path which has write permission. For example, `\\MyServer\SibFS\SiebConfig`.

NOTE: The value for the directory that you specify must be network-accessible.

It is recommended that you do not specify a top-level directory. For example, if the directory `SibFS` is a top-level directory, then specify a sub-directory such as `SiebConfig`.

If you do not specify a value for eProdCfgCacheFS, then Siebel Configurator attempts to use the Siebel File System. If the Siebel File System uses the File System Manager (alias FSMSrvr) component, Siebel Configurator does not cache object definitions to the file system. For more information about the Siebel File System, see *Siebel System Administration Guide*.

Defining Customizable Product Models and Classes

This section 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 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 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 Configurator may 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 on these issues, see *Siebel Product Administration Guide*.

About Siebel Configurator Caching

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

- Caching in memory

  Siebel 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 Configurator" on page 96.

- Caching in the Siebel Configurator File System

  This 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 "Default Caching Behavior for Siebel Configurator" on page 96.

- In addition to the above 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 Configurator" on page 97.

NOTE: The memory resources for your Siebel Configurator server machine must be sufficient to support your caching requirements.

In addition to the topics referred to previously, this section includes the following topics:

- "Parameters for Configuring Siebel Configurator Caching" on page 98
- "Determining Rough Sizing for Caching Parameters" on page 101
- "Administering the Siebel Configurator Cache" on page 101

Default Caching Behavior for Siebel Configurator

The default caching behavior for Siebel Configurator is as follows:

- When a user starts a configuration session, Siebel 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, Siebel Configurator updates or removes these objects.
- Siebel Configurator looks to see if the requested customizable product is cached in memory.
- If the customizable product is not already cached in memory, Siebel Configurator looks in the Configurator File System.

  NOTE: The location of the Configurator File System is specified by the value of the Product Configurator - FS location parameter (alias eProdCfgCacheFS). If no value is specified for eProdCfgCacheFS, Siebel Configurator looks in the Siebel File System. For more information about the Configurator File System, see "Specifying the Siebel Configurator File System Location" on page 95.

- If the customizable product is not in the Configurator File System, it is loaded from the Siebel database. The product is added to the memory cache and to the Configurator File System.
- Thereafter, when a configuration session starts, the customizable product is loaded from the memory cache or the Configurator File System.
- Before loading the customizable product from the 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, 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 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 Configurator

When a user starts a configuration session, Siebel 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. This requires that you select the setting *Explicit Product Mappings Only* on the specified cache. 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:

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

For more information on these parameters, see Table 4 on page 99.

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.

To configure product caching

1. Navigate to Administration - Product > Cache Administration.

   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.

  NOTE: If you select Component as the cache type, you must set the same value for the component parameter Enable internal load balancing (alias EnableVirtualHosts) on both the AOM and on the eProdCfgObjMgr component. For example, if EnableVirtualHosts is set to TRUE on the AOM 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 Step 2.

The value that you specify depends on the value that you selected for Cache Type in Step 2. For example, if you set Cache Type equal to Server, you enter the name of a Siebel Server. If you set Cache Type equal to Component, you enter the name the name of a component.

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 product(s) that you want to associate with the component that you selected in Step 4.

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.

8 If the configuration that you created validates correctly, click Release to enable the cache that you selected cache instances of the products that are mapped to it.

Parameters for Configuring Siebel Configurator Caching

Siebel 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" on page 101.

Table 4 lists the server parameters for configuring Siebel Configurator caching. Set these parameters on the AOM component for an AOM deployment of Siebel Configurator. For a dedicated Siebel Configurator server deployment, set these parameters on the AOM *and* on the eProdCfgObjMgr component.

For information on how to configure server parameters, see the *Siebel System Administration Guide*.

Table 4. Server Parameters for Configuring Siebel Configurator Cache Behavior

Parameter Alias	Parameter Name	Data Type	Default Value	Description
eProdCfgCacheFS	Product Configurator - FS location	String		Specifies the location of the Configurator File System. If no value is specified for eProdCfgCacheFS, Configurator looks in the Siebel File System. For more information about the Configurator File System, see "Specifying the Siebel Configurator File System Location" on page 95.
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.

Table 4. Server Parameters for Configuring Siebel Configurator Cache Behavior

Parameter Alias	Parameter Name	Data Type	Default Value	Description
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 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. **NOTE:** This parameter provides the functionality provided by the parameter eProdCfgNumbOfCachedFactories in previous releases.
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 pre-caching.

Table 4. Server Parameters for Configuring Siebel Configurator Cache Behavior

Parameter Alias	Parameter Name	Data Type	Default Value	Description
ServerSessionLoopSleepTime	ServerSessionLoopSleepTime	Integer	300	Specify an interval time (in seconds) to refresh cached products that are mapped to a Configurator server cache or component cache using the explicit product mapping setting. **NOTE:** eProdCfgCacheEngineObjects must be set to TRUE.

Determining Rough Sizing for Caching Parameters

To help you determine how to set the Siebel 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 (a Mobile Web Client 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 maximum likely memory usage).

For example, X may be the before-loading memory size, Y may be the after-loading size, and Z may 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 – X).
- The size of a Worker for a customizable product varies during runtime, generally increasing as user selections are made. This size may be approximated by subtracting the Model Manager size from the difference of Z less X.

Administering the Siebel Configurator Cache

Siebel administrators or product administrators can refresh or update the Siebel Configurator cache in a number of ways. The following sections describe procedures to refresh or update the Siebel Configurator cache with changes for customizable products, product classes, and attribute definitions. For more information, see the following sections:

Refreshing the Entire Siebel Configurator Cache

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

Other topics within this section describe how you can update the Siebel Configurator cache with changes you made to customizable products, product classes, and attribute definitions in separate operations. For more information, see "Administering the Siebel Configurator Cache" on page 101.

To refresh the entire Siebel Configurator cache

1. Navigate to Administration - Product > Cache Administration.
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 Configurator Cache with Product Changes

While editing a product record, a product administrator can select Refresh Product Cache to refresh the Siebel 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.

For more information about refresh options for the Siebel Configurator cache, see "Administering the Siebel Configurator Cache" on page 101.

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 Configurator Cache with Product Class Changes

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 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 non-versioned cached property such as, for example, a product name. If the product administrator does not select Update Cache, an AOM that has a version already cached uses the old version.

For more information about administration options for the Siebel Configurator cache, see "Administering the Siebel Configurator Cache" on page 101.

To update the cache with class changes

1. Navigate to Administration - Product > Product Classes.

   The Product Classes list applet 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 Configurator Cache with Product Class Changes

While editing a product class record, a product administrator can select Refresh Cache to refresh the customizable products in the Siebel 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.

For more information about refresh options for the Siebel Configurator cache, see "Administering the Siebel Configurator Cache" on page 101.

To refresh the cache with class changes

1. Navigate to Administration - Product > Product Classes.

   The Product Classes list applet 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 Configurator Cache with Attribute Definition Changes

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 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 non-versioned cached property such as, for example, an attribute definition name. If the product administrator does not select Update Cache, an AOM that has a version already cached uses the old version.

For more information about administration options for the Siebel Configurator cache, see "Administering the Siebel Configurator Cache" on page 101.

To refresh the cache with attribute definition changes

1. Navigate to Administration - Product > Attribute Definitions.

   The Attribute Definitions list applet appears.

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

Refreshing the Siebel Configurator Cache with Attribute Definition Changes

While editing an attribute definition record, a product administrator can select Refresh Cache to refresh customizable products in the Siebel Configurator cache with the changes made to the attribute definition record.

The next time a user requests the customizable product, the user receives a freshly instantiated version reflecting the attribute definition change and the cache is refreshed with this version. This new instance reflects the changes you made to the attribute definition.

For more information about refresh options for the Siebel Configurator cache, see "Administering the Siebel Configurator Cache" on page 101.

For more information, see "Administering the Siebel Configurator Cache" on page 101.

To refresh the cache with attribute definition changes

1. Navigate to Administration - Product > Attribute Definitions.

   The Attribute Definitions list applet appears.

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

9 Tuning Siebel EAI for Performance

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

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*

About Siebel Enterprise Application Integration

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

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

Best Practices for Siebel EAI Tuning

This section describes best practices 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 Business Applications:

- Try to 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.

- Try to keep processing in a single session wherever possible, to avoid having to make calls between server components.
- Within a session, try to 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, use it minimally and economically and apply documented guidelines. For more information, see "Best Practices for Siebel Scripting" on page 160.
- Configure business components, business services, caching, and other application functionality that supports integration processing to obtain optimal performance. For more information, see other sections in this chapter and see Chapter 12, "Tuning Customer Configurations for Performance."
- Perform capacity planning for all servers that support integration processing. Siebel Expert Services may be consulted for sizing reviews.
- Try to 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. For more information, see:

Improving IBM WebSphere MQ Transport Performance

The performance of an IBM WebSphere MQ queue is highly dependent on the disk performance of the queue manager machine and the layout of the queue's files on the disk. You should 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 machine or across several machines. 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 you are using may 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, you may 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, turn it off.

  Persistent queuing is significantly slower than normal queuing for WebSphere MQ. If you do not use this feature, however, 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 you have running.

In addition, there are specific actions you can take to improve WebSphere MQ Transport performance for outbound and inbound transports, as detailed below.

Inbound Messages

For inbound WebSphere MQ messages, run multiple MQ Receivers in parallel to increase throughput. See additional comments earlier in this topic for details.

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 recompile the SRF file.
- If you need to call the WebSphere MQ Transport in Workflow Process Manager and in a client session, 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, 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

The HTTP Inbound Transport supports two modes, session mode and sessionless mode:

- In session mode, the session stays live until a logoff call is made
- In sessionless mode, login and logoff occur automatically for each request

You should use session mode whenever possible, because the time required to log into 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 SISNAPI connection between the Web server 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, 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
```

For more information about configuring SessPerSisnConn, see "Configuring SISNAPI Connection Pooling for AOM" on page 45.

EAI Siebel Adapter Performance

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 "Best Practices for Siebel Scripting" on page 160.

Disabling Logging

You should 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 server 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, 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, 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" on page 170.

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" on page 106.

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, 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, one will succeed and one will produce an error. Therefore, when running the EAI Siebel Adapters in parallel, you need to 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

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 section discusses some performance issues for the Workflow Process Manager component.

For more information about Siebel Workflow performance, see Chapter 7, "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 AOMs) that may invoke workflow processes are cached and reused for subsequent requests. When sizing a system, you need to look at the maximum number of workflow tasks you expect to have active at a given time. This determines the maximum number of Object Manager sessions Siebel applications create. In general, it is recommended that, where possible, you create small workflow processes. If you cannot avoid creating a large workflow process, then divide the workflow process into sub-processes.

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, you may want to run Workflow Process Manager on multiple Siebel Server machines. 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), you may also want to run multiple multithreaded processes.

If you are going to run several different types of workflows, you should 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 machine. 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 Best Practices for Siebel EAI

Review the following issues for applicability to your deployment, for optimizing Siebel EAI performance:

- **Check disks on the machine.** Do a preliminary test on the queue manager 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 you can use to drive these and determine how much the queue itself can scale. The speed of the disks on the machine is important.
- **Optimize messages.** In the messages, reference only the columns you require.
- **Create smaller business components.** Messages might use only a small portion of the actual business components.

  Create copies of the business components you are using. In the copies, keep active all fields used by the optimized integration object or otherwise used for correctly 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 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 may 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, 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

This chapter covers recommended best practices for improving the performance of EIM and is organized into the following sections:

About Siebel EIM

Siebel Enterprise Integration Manager (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.

EIM is your primary method of loading mass quantities of data into the Siebel database. You should use EIM to perform bulk imports, updates, merges, and deletes of data.

In the Siebel application database, there are application tables (known as base tables), which Siebel applications use. 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 parts:

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 part of this data-exchange process involves the intermediary tables that are called EIM tables, only the second part of the process involves the functionality of Siebel EIM.

When data is entered through the Siebel user interface, the application references properties set at the business component object type. However, when data is entered into Siebel base tables through EIM, EIM references properties set at the table object type.

NOTE: You must use EIM to perform bulk imports, exports, merges, and deletes, because Oracle does *not* support using native SQL to load data directly into Siebel base tables (the tables targeted to receive the data). You should also be aware that EIM translates empty strings into NULL.

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 may not be your highest priority when you initially 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 Expert 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.

For more information, see the following topics:

- "Database Sizing Guidelines" on page 114
- "Database Layout Guidelines (Logical and Physical)" on page 115

Database Sizing Guidelines

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, performance will be affected and, in extreme cases, the system itself may be halted. If you allocate too much space, it may cause inefficiency.

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 on 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 Business Applications (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*, 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)

The overall performance of Siebel Business Applications is largely dependent 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 should be separated across devices. In general, the indexes listed below should be on different physical devices from the tables on which they are created.

  - S_ACCNT_POSTN
  - S_OPTY
  - S_ADDR_ORG
  - S_OPTY_POSTN
  - S_CONTACT
  - S_POSTN_CON
  - S_DOCK_TXN_LOG
  - S_PARTY_REL
  - S_PARTY
  - S_SRV_REQ
  - S_EVT_ACT
  - S_OPTY
  - S_ORG_EXT

  NOTE: If you plan on making extensive use of EIM, 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 EIM operations.

EIM Usage Planning

This topic provides a number of general guidelines for effective and efficient implementations of EIM, regardless of the size of the overall Siebel implementation. It cannot be emphasized enough that taking a strategic perspective to implementing EIM is crucial not only to being able to use EIM, but to the overall success of the Siebel implementation.

For more information, see the following subtopics:

- "Team Definition" on page 116
- "Mapping Data into Siebel Applications" on page 117
- "Testing EIM Processes" on page 118

Team Definition

Based on customer experience, it is recommended that a team of individuals is assigned to manage and maintain the EIM processes required for your organization. You should consider using individuals with the following skill sets:

- For small to medium-sized Siebel application 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 on performing this task, see the *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 EIM in order to process the data into the Siebel base tables.

NOTE: Your organization may 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, 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 to make 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 on performing this task, see the *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 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 EIM team.

Mapping Data into Siebel Applications

EIM uses EIM table mappings to map columns from EIM tables to Siebel base tables. Siebel predefined EIM mappings are fixed and cannot be remapped.

NOTE: EIM uses only EIM table mappings to determine table relationships. EIM does not use configuration logic in the Siebel repository to determine table relationships.

Using Siebel Tools, you can view:

- EIM table mappings to Siebel base tables
- Column mappings to Siebel base table columns
- Siebel base table mappings to EIM tables

Some base tables may 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 Siebel Technical Services regarding the missing mapping. For information on 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 EIM support mappings between columns in extension tables and 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 on Database Extensibility, see *Configuring Siebel Business Applications*.

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.

   To facilitate this, you can request an EIM Data Mapping and Design review from Siebel Expert Services. This review can be used to make sure that the EIM mappings are correct and will accomplish intended goals.

Testing EIM Processes

This issue, fully and completely testing the EIM processes, tends to be overlooked. Testing is more than simply mapping the data and then running an EIM process using the default EIM configuration file. Complete testing requires you to run a large number of identical EIM jobs with similar data. This allows you to not only find any areas that you may have overlooked, but it also provides some insight into the optimal sizing of the EIM batches and exposure to scenarios that may occur in a production environment.

Before using EIM, a database administrator must populate the EIM tables with data to be processed by EIM. Then, you can invoke EIM to process this data, with EIM making multiple passes through the tables to complete the specified process.

EIM reads a special configuration file that specifies the EIM process to perform (import, merge, delete, or export) and the appropriate parameters. The 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 an EIM process, you must edit the contents of the EIM configuration file to define the processes that EIM will perform.

The 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 on 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. This setting will show errors and unused foreign keys. The setting Trace Flags=1 will provide a summary (after each batch) of the elapsed time after EIM updates primary child relationships in the Siebel database tables as necessary and runs optional miscellaneous SQL statements.

- Set Error flag = 1, SQL flag = 8, and 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 Error flag = 0, SQL flag = 0, and Trace flag = 1. These settings will produce a log file showing how long each EIM step took, which is useful when figuring out the optimal batch size as well as monitoring for deterioration of performance in a particular step.

General Guidelines for Optimizing EIM

The following guidelines are recommended for improving EIM performance:

- Verify that all indexes exist for the tables involved. Keep in mind, however, that for large loads you should 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.
- Limit tables and columns to be processed using ONLY BASE TABLES/COLUMNS configuration parameters to minimize EIM processing.
- Consider disabling the Docking: Transaction Logging system preference during the EIM run. Switching 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 EIM process you are running.

  NOTE: Although the limit of rows you can process is directly related to the capabilities of your database server, executing batches greater than 100,000 rows is strongly discouraged.
- For 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 an 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 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, use the SQLPROFILE parameter to identify slow-running steps and queries. For more information, see "Using the SQLPROFILE Parameter" on page 126.

Recommended Sequence for Implementing EIM Processes

The following sequence is recommended for implementing EIM processes:

1. Customize and test the .IFB file to meet the business requirements.
2. Tune the .IFB parameters.
3. Separate the EIM processes.
4. Set the database parameters, making sure the basic requirements are met, including the hardware, the settings, and no or minimal fragmentation.

Before you start optimizing EIM processes, make sure there are no network problems or server performance problems that can affect the results. Siebel Expert Services recommends using at least 100 MB network segments and network-interface cards (NICs) to connect the Siebel server and Siebel database server. In addition, Siebel Expert Services recommends using a network switch or similar technology, rather than a hub, to maximize throughput.

Optimizing the .IFB File

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 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 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 others. The default configuration is to process all base tables for each EIM table.

  NOTE: Siebel Expert 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 EIM task.

- **ONLY BASE COLUMNS** or **IGNORE BASE COLUMNS**. These parameters specify and restrict the selected base columns for the 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 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; this is because EIM does not need to perform the interim processing (using SQL statements) to resolve the values for these FKs. Set the EIM Task parameter Error Flags = 1 to see which FKs are failing to be resolved by EIM (you may have missed excluding that FK with this parameter).

  NOTE: Do not use the IGNORE BASE COLUMNS parameter for merge processes or export processes. This parameter should only be used for import processes and delete processes.

Checking .IFB File Optimization

One method to find out if 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 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 only took 2 minutes to import (excluding the Updating Primaries step).

Separating EIM Processes by Operation

Wherever possible, divide the 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, EIM can determine which records are to be added and which records are to be updated in the base tables, however, 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, you could avoid this additional processing.

In addition, the columns being processed as part of the update activity might be less than those for the insert activity (resulting in an additional performance increase). To illustrate this, the .IFBs in the preceding 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 EIM Performance

Before troubleshooting EIM performance, verify that there are no performance bottlenecks on the Siebel Server machine or network.

Optimizing SQL for EIM

During this process, you need to be able to run several similar batches. If you do not have enough data with which to experiment, you may need to back up and restore the database between runs, so that you can continue processing the same batch.

First, you should run an 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 you should consult an experienced database administrator at this point. The process of optimizing the SQL for 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 may be cases where an index is not used at all or the most efficient index is not being chosen. This may require a thorough analysis.
- Based on this analysis, use a systematic approach to tuning these long-running statements. You should perform one change at a time and then measure the results of the change by comparing them to the initial benchmarks. For example, you may find that dropping a particular index to improve the performance of one long-running statement might negatively impact the performance of other SQL statements.

  The decision on whether to drop the index should be based 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 EIM tasks.

Using the USE INDEX HINTS and USE ESSENTIAL INDEX HINTS Parameters

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, you must specify USE INDEX HINTS within each one.

If these parameters are set to FALSE, EIM does not generate hints during processing. By setting the value to FALSE, you may realize performance gains if the TRUE setting means that hints are being generated that direct the database optimizer to use less than optimal indexes. EIM processing should be tested with both the TRUE and FALSE settings to determine which one provides better performance for each of the respective EIM jobs.

NOTE: The USE INDEX HINTS parameter is only applicable for Oracle database platforms. The USE ESSENTIAL INDEX HINTS parameter is only applicable for Microsoft SQL Server and Oracle database platforms.

These two parameters work for different queries, so you need to enable both to get all of the index hints on Oracle database platforms.

Further information is provided as follows:

- "Example: Using the USE INDEX HINTS and USE ESSENTIAL INDEX HINTS Parameters" on page 124
- "USE INDEX HINTS and USE ESSENTIAL INDEX HINTS: EIM Criteria for Passing Indexes to the Database" on page 125

Example: Using the USE INDEX HINTS and USE ESSENTIAL INDEX HINTS Parameters

The following example illustrates the results achieved for an SQL statement with index hints and without index hints. This example was performed on the Microsoft SQL Server platform.

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.
```

USE INDEX HINTS and USE ESSENTIAL INDEX HINTS: EIM Criteria for Passing Indexes to the Database

This topic explains how 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. When determining which indexes to pass on to the database as index hints, EIM takes the following steps:

1. Before generating a query, EIM makes a list of columns for which it has determined that an index is needed.
2. EIM then checks all of the indexes in the repository to find the index with the most matching columns.

EIM uses the following selection criteria in choosing indexes:

- Unique indexes have priority over non-unique indexes.
- Required columns have priority over non-required columns.

If a new index is created and it is declared in the repository, then there is a chance that EIM will choose it and pass it to the database on a hint.

NOTE: On Oracle databases, EIM uses the Oracle rule-based optimizer (RBO) mode. When you specify an INDEX hint, the RBO mode knows that it must use the index specified in the hint. In this case, the RBO mode does not perform a full table scan, nor does it use any other index. For more information about optimizer modes for Oracle databases, see the *Siebel Database Upgrade Guide*.

Using the SQLPROFILE Parameter

The inclusion of this 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 v6.0.1.2 [2943] ENU SQL profile dump (pid 430).

********************************************************************************

Top 34 SQL statements (of 170) by total time:

Batch Step Pass Total Rows Per Row What

-------- -------- -------- -------- -------- -------- -----------------

106 10 401 1334.48 5000 0.27 update implicit primaries to child

106 9 114 242.56 5000 0.05 copy
```

…list of queries continues

Statistics by step and by pass

```
********************************************************************************

Statements per step by total time:

Step Stmts Total Min Max Avg %

-------- -------- -------- -------- -------- -------- --------

10 15 2627.27 0.00 1334.48 175.15 83.73

9 11 329.52 0.00 242.56 29.96 10.50
```

...list of statistics continues...

SQL statements:

```
    ********************************************************************************
    batch 106, step 10, pass 401: "update implicit primaries to child":
    (total time 22:14m (1334s), 5000 rows affected, time/row 0.27s)
    UPDATE siebel.S_CONTACT BT
    SET PR_BL_PER_ADDR_ID =
    (SELECT VALUE(MIN(ROW_ID), 'No Match Row Id')
    FROM siebel.S_ADDR_PER CT
    WHERE (CT.PER_ID = BT.ROW_ID)),
    LAST_UPD = ?,
    LAST_UPD_BY = ?,
    MODIFICATION_NUM = MODIFICATION_NUM + 1
    WHERE (ROW_ID IN (
    SELECT T_ADDR_PER_PER_ID C1
    FROM siebel.EIM_CONTACT
    WHERE(
    T_ADDR_PER_PER_ID IS NOT NULL AND
    IF_ROW_BATCH_NUM = 106 AND
    T_ADDR_PER__STA = 0 AND
    T_ADDR_PER__EXS = 'N' AND
    T_ADDR_PER__UNQ = 'Y' AND
    T_ADDR_PER__RID IS NOT NULL)
    GROUP BY T_ADDR_PER_PER_ID)
    AND
    (PR_BL_PER_ADDR_ID IS NULL OR PR_BL_PER_ADDR_ID = 'No Match Row Id'))
    ************************************************************
```

...list of SQL statements continues...

Additional Indexes on EIM Tables

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.

NOTE: Having the ROW_ID column as the leading index column would also be a good strategy. Since the ROW_ID is unique, the index is likely to be more selective.

Adding Indexes to Improve Performance of S_ORG_EXT

Table S_ORG_EXT has indexes on many columns, but not all columns. If you have a large number of records (several million accounts) in S_ORG_EXT, you may get a performance improvement in deleting and merging by adding an index to one or more of the following:

- PR_BL_OU_ID
- PR_PAY_OU_ID
- PR_PRTNR_TYPE_ID
- PR_SHIP_OU_ID

Before implementing any additional indexes, first discuss this with qualified support personnel.

Creating Proper Statistics on EIM Tables

Use of the .IFB file parameter UPDATE STATISTICS is only applicable to the DB2 database platform. This parameter can control whether 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 EIM runs.

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, you should execute `db2look utility` in mimic mode to save the statistics from the database system catalogs. For example, if you are testing EIM runs using EIM_CONTACT1 in database SIEBELDB, 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 EIM batches after inserting the RUNSTATS commands provided in "DB2 Version 8 Options." After you find the set of statistics that works best, you can apply that particular mim.sql file to the database.

NOTE: Do not forget to save statistics with `db2look` between runs.

DB2 Version 8 Options

The syntax for DB2 V8 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

Typically, the EIM initial load is a very database-intensive process. Each row that is inserted into the base table requires modifications on the data page and the index pages of all the affected indexes. However, most of these indexes are never used during an EIM run. Index maintenance is a very time-consuming process for most database managers and should be avoided as much as possible.

Therefore, the goal is to determine any indexes that are unnecessary for EIM and that can be dropped for the durations of the 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 EIM import. Past experience has determined that the Fx and Mx indexes can be dropped after an extensive SQL analysis of sample 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 EIM runs.

NOTE: Testing should always be performed when dropping indexes (or adding indexes) to make sure that expected results are achieved.

Controlling the Size of Batches

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. You should run several tests with different batch sizes to determine the size that provides the best rate of EIM transactions per second. Using the setting Trace Flag = 1 while running 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 EIM process.

NOTE: You should also monitor this throughput rate when determining degradation in parallel runs of 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. You should not exceed 100,000 rows in a large batch.

Furthermore, for Microsoft SQL Server and Oracle environments, you should 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 EIM processes, then you should have 152,000 rows in the EIM table. Under no circumstances should you have more than 250,000 rows in any single EIM table because this reduces performance.

The restrictions mentioned in the example above do not apply to 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.

NOTE: The number of rows you can load in a single batch may vary depending on your physical machine setup and on which table is being loaded. To reduce demands on resources and improve performance, you should 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 may find that loads perform better at higher batch sizes than for more complex tables such as S_CONTACT.

Controlling the Number of Records in EIM Tables

You should determine the number of records that can reside at one time in an EIM table while still maintaining an acceptable throughput rate during EIM processing. One observed effect of increasing the number of records in an EIM table is reduced performance of EIM jobs. This is often caused by object fragmentation or full table scans and large index range scans.

NOTE: In a DB2 environment, EIM table size is not an important factor that impacts performance, because it is easy to correct table scans and non-matching 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 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 should start with approximately 50,000 records. After tuning efforts are complete, you should 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

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. You should only set this parameter to FALSE when account synonyms are not needed.

Using the NUM_IFTABLE_LOAD_CUTOFF Extended Parameter

Setting this extended parameter to a positive value will reduce the amount of time taken by 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 on this parameter, see the *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 Docking: Transaction Logging

Typically, a disabled Docking: Transaction Logging setting is only used during initial data loads. The value of the system preference, Docking: Transaction Logging, is set from the System Preferences view within the Siebel application. This setting indicates whether or not the Siebel application logs transactions for the purpose of routing data to Siebel Mobile Web Clients.

The default for this parameter is TRUE. If there are no Siebel Mobile Web Clients, you can set this system preference to FALSE. If you have Siebel Mobile Web Clients, then this parameter must be set to TRUE in order to route transactions to the Siebel Mobile Web Clients. However, during initial data loads, you can set this parameter to FALSE to reduce transaction activity to the Siebel docking tables. After the initial loads are complete, set the parameter back to TRUE.

NOTE: For incremental data loads, Docking: Transaction Logging should remain set to TRUE 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 Triggers

Disabling database triggers, by removing them through the Server Administration 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 EIM load.

Running EIM Tasks in Parallel

Running EIM tasks in parallel is the last strategy you should adopt in order to increase the 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 the *Siebel System Administration Guide*.

NOTE: UPDATE STATISTICS must be set to FALSE in the .IFB file when running parallel EIM tasks on the IBM DB2 platform. Failure to do so can cause EIM tasks and executing RUNSTATS to take a longer time to complete. Also, when running parallel EIM tasks, deadlock and timeout will occur if UPDATE STATISTICS is set to TRUE in the .IFB file.

Database Guidelines for Optimizing EIM

The following topics describes EIM tuning tips for the database platforms supported by Siebel applications (Microsoft SQL Server, Oracle, and IBM DB2 UDB). For more information, see the following topics:

Microsoft SQL Server

The following sections describe EIM tuning tips for the Microsoft SQL Server database platform.

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 may 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 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 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 should be at or near 100%. If it is significantly below 100%, 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 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. You should use the bcp utility or BULK INSERT statement when the following conditions are true:

- The 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 you should immediately back up so 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 it is empty when the bulk copy starts.
4. Make sure you are not replicating the target table.
5. Make sure 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 should 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 should 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 should be 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, may 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. This option should be set 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. This option should be enabled.
- **AUTO UPDATE STATISTICS.** This allows Microsoft SQL Server to automatically manage database statistics and update them as necessary to achieve proper query optimization. This option should be enabled.

Oracle Databases

This section provides EIM tuning tips for the Oracle database platform.

Avoiding Excessive Table Fragmentation

Before running EIM, you should consult with 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, 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 disallocation activities (such as truncate or drop commands) can be demanding on CPU resources.

To check if segment extension is occurring in an 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 an 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 EIM processes can be executed against an EIM table provided they all use different batches or batch ranges. The concern is that you may experience contention for locks on common objects. To run multiple jobs in parallel against the same EIM table, you should make sure that the FREELIST parameter is set appropriately for the tables and indexes used in the EIM processing.

This includes EIM tables and indexes, as well as base tables and indexes. The value of this parameter specifies the number of block IDs that will be stored in memory which are available for record insertion. Generally, you should set this 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 should permit 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 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 updates the wrong data due to hash keys overflow. This is an Oracle-specific issue. To avoid this problem, use less than 255 NVL functions in the update statement.

IBM DB2 UDB

This topic describes EIM tuning tips for the IBM DB2 UDB database platform.

In addition to the information available here and elsewhere on *Siebel Bookshelf*, IBM have published a Redbook, *Siebel 7.8 with IBM DB2 UDB V8.2 Handbook*, which provides additional information about Siebel Business Applications on the IBM DB2 UDB database platform. See http://www.redbooks.ibm.com for more information.

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 EIM loads or where many EIM tasks execute in parallel, place individual EIM tables in separate tablespaces.
- Use large page sizes for EIM and 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 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 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 and the base table bufferpools.
- After you load new data, reorganize the tables if the data on a disk is out of cluster. Use the REORGCHK command if the results of executing the RUNSTATS command indicate that clustering has deteriorated (clustering index < 80% clustered) and that a reorganization of tables is required. For more information about using the REORGCHK command, see FAQ 2072 on Siebel SupportWeb.

  NOTE: Allocate time to conversion schedules to allow for the reorganization of tables and the gathering of statistics prior to allowing end users access a system containing new data.
- Use IBM DB2 snapshot monitors to make sure performance is optimal and to detect and resolve any performance bottlenecks.
- You can turn off logretain during the initial load. However, you should 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 EIM processes in parallel will cause deadlock and timeout on IBM DB2 UDB databases if multiple EIM processes attempt to update the same catalog table simultaneously. To avoid this, set the UPDATE STATISTICS parameter to FALSE in the EIM configuration file (.IFB file).
- Executing UPDATE STATISTICS in each EIM process consumes significant database server resources. It is recommended that the database administrator updates statistics outside of the EIM process using the RUNSTATS command.
- Consider the following settings for 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

- Consider the following settings for the DB2 database manager configuration parameters:

Registry Value	Setting
INTRA_PARALLEL =	NO (may be used during large index creation)
MAX_QUERYDEGREE =	1 (may be increased during large index creation)
SHEAPTHRES =	100,000 (depends upon available memory, SORTHEAP setting, and other factors)

- Consider the following settings for the 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
LOGRETAIN =	NO (only during initial EIM loads)
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 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 9 EIM batches if you set SHEAPTHRES = 100000. If executing concurrent EIM batches, make sure to allocate sufficient physical memory so that memory swapping or memory paging do not occur.
STAT_HEAP_SZ =	8000

IBM DB2 UDB for z/OS

For DB2 configuration settings, you can find a listing (from the JCL) of the Database Manager Configuration Parameters (DSNZPARM) in *Implementing Siebel Business Applications on DB2 UDB for z/OS and OS/390*.

Further IBM DB2 information is provided in the following sections:

- "IBM DB2 Loading Process for EIM" on page 141
- "General Recommendations for the IBM DB2 Loading Process" on page 141

IBM DB2 Loading Process for EIM

Figure 3 illustrates the load process for IBM DB2.

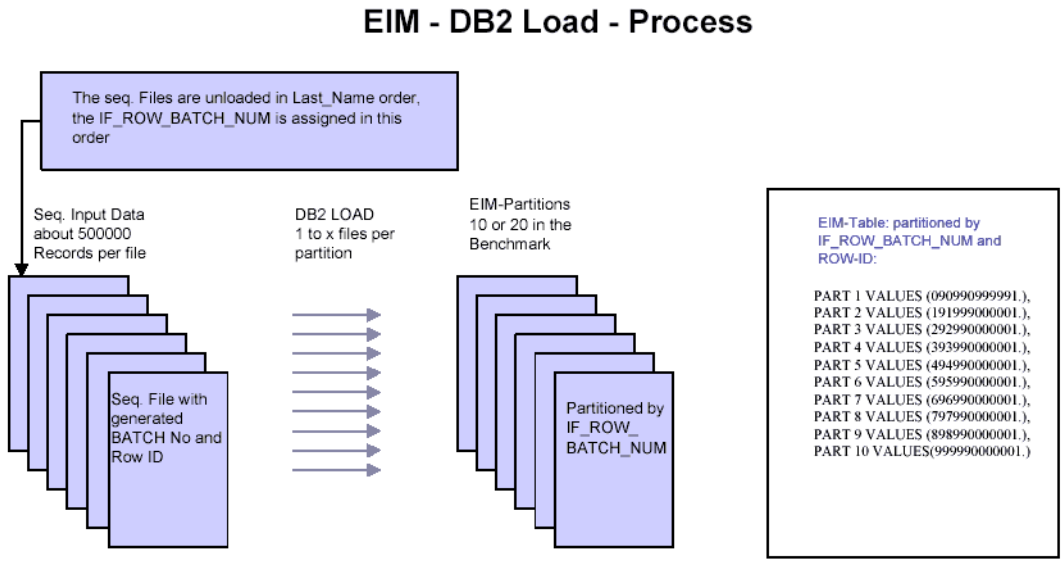

Figure 3. IBM DB2 Loading Process for EIM

For more information, see the *Siebel Enterprise Integration Manager Administration Guide*.

General Recommendations for the IBM DB2 Loading Process

The following general recommendations apply when performing the IBM DB2 loading process for EIM:

- Use the ONLY/IGNORE BASE TABLES parameters or ONLY/IGNORE BASE COLUMNS parameters in the .IFB files to reduce the amount of processing performed by 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 2,000 records, 5000 for DB2/390). EIM can be configured through the use of an extended parameter to use a range of batches, you should remember to put the variable name into the .IFB file.
- Multiple EIM processes can be executed against an EIM table, provided they all use different batches or batch ranges. However, the main limit to EIM performance is not the application server but the database. Contention for locks on common objects may occur if multiple EIM streams are executed simultaneously for the same base table. Multiple EIM job streams can run concurrently for different base tables, for example, S_ORG_EXT and S_ASSET.
- Run 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 EIM processes.
- Set the system preference (in Administration - Application > System Preferences) for Docking: Transaction Logging to FALSE during the initial database load. This 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 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.
- Remember to 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 the EIM table is empty before loading data into it for 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, truncating the EIM table between loads from the data source helps to preserve performance.
- When running Siebel applications on an IBM DB2 database, 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 may 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, you should 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 may need to run other SQL statements beforehand.

Data Management Guidelines for Optimizing EIM

The following recommendations apply when performing the EIM loading process:

- The 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.
- 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 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 EIM

The following recommendations are for setting run parameters when performing the EIM loading process:

- 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 applications use 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 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 tables 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 EIM attempts to resolve.
- Set the USING SYNONYMS parameter to FALSE in the .IFB file. This logical operator indicates to 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, 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 an EIM Task

When monitoring the Siebel server, the assumption is that you have allocated sufficient processor and memory resources for running the EIM task on the Siebel application servers and Siebel database servers.

If you are using Microsoft Windows 2000 as the operating system for the Siebel Server, the Microsoft Windows Performance Monitor can be used to verify the amount of processor and memory being used by the hardware.

If you are using Sun Solaris or IBM AIX as operating systems for the Siebel Server, you can use *vmstat* and *iostat* to verify the amount of processor and memory being used by the hardware.

11 Tuning Siebel Remote for Performance

This chapter discusses tuning for Siebel Remote that may enhance performance. It contains the following topics:

For more information about Siebel Remote, see the *Siebel Remote and Replication Manager Administration Guide* on the *Siebel Bookshelf*. Siebel SupportWeb also contains documents such as Troubleshooting Steps and Technical Notes 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 the following components to manage the exchange of data and files:

- Database Extract (alias DbXtract)
- Generate New Database (alias GenNewDb)
- Parallel Database Extract (alias PDbXtract)
- Synchronization Manager (alias SynchMgr)
- Transaction Merger (alias TxnMerge)
- Transaction Processor (alias TxnProc)
- Transaction Router (alias TxnRoute)

For more information about each of these components, see the *Siebel Remote and Replication Manager Administration Guide*.

The section, "Tuning Siebel Remote Server Components" on page 146, discusses how you can configure some of these components to optimize the performance of your Siebel Remote deployment.

Tuning Siebel Remote Server Components

This section describes how you can improve the performance of certain components on the Siebel Remote Server. It includes the following sub-sections:

Increasing Throughput for the Database Extract and Parallel Database Extract Components

The following list describes tips to improve the throughput for the Database Extract and Parallel Database Extract components:

- Run multiple instances

  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, you create them using Siebel Tools.

  The recommended number of temporary tables to use depends on the database platform in use. For example:

  - Microsoft SQL Server and IBM DB2 Universal Database

    Use one temporary table for each instance of the Database Extract component that you execute. For example, if you execute 11 instances of the Database Extract component, then use 11 temporary tables.

  - Oracle Enterprise Server

    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, 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 may increase throughput.

- Transaction Router component

  Stop the Transaction Router component if running multiple instances of Database Extract or Parallel Database component in order to avoid table locking.

- Extract lists of users simultaneously

  Separate users to be extracted into groups of 50 - 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 as deleting from S_DOCK_INITM_*N* is usually logged and can take a long time.

- Defragment tables

  Defragment tables to which you intend to extract data and S_DOCK_INITM_*N* tables.

For more information about the components and parameters described previously, see *Siebel Remote and Replication Manager Administration Guide*. For more information about performance issues for the Database Extract component, see Troubleshooting Steps 15 on Siebel SupportWeb.

Tuning the Transaction Router Component

This section describes how to resolve or avoid performance issues for the Transaction Router component that arise from the following sources:

- Visibility-Related Transactions
- Docking Rules and Data Distribution
- Slow-Running Queries
- Increasing Transaction Router Throughput

For further information on Transaction Router performance issues, consult the following documents on Siebel SupportWeb:

- Troubleshooting Steps 8

  This document describes how to diagnose and resolve Transaction Router performance issues.

- Troubleshooting Steps 38

  This document 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, consider the following two possible solutions:

- Reextract all mobile users and regional nodes

  For more information, see the *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 should 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, log a service request with Siebel Technical Support and provide the following pieces of information:

- RDBMS trace of the Transaction Router task
- Transaction Router log files
- .dx files 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 Troubleshooting Steps 8 on Siebel SupportWeb.

Slow-Running Queries

If you diagnose the root cause of the Transaction Router performance issue to be slow-running queries, 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 the *Siebel Data Model Reference*.

- Check if the tables and indexes involved in the poor-performing queries require de-fragmentation.

Increasing Transaction Router Throughput

The following factors can impact throughput for the Transaction Router component:

- Large batch sizes for Siebel Enterprise 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 the *Siebel Enterprise Integration Manager Administration Guide*.

- Large batch size for Siebel Assignment Manager

  Where possible, reduce the batch file size for Siebel Assignment Manager as large batch files can impact the performance of the Transaction Router component. For information, see the *Siebel Assignment Manager Administration Guide*.

In both cases described above, 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.

Tuning the Mobile Web Client in a Siebel Remote Deployment

This section discusses how you can optimize the performance of a Mobile Web Client in a Siebel Remote deployment. It includes the following topics:

For additional information about performance tuning for Mobile Web Clients, see Chapter 5, "Tuning Siebel Web Client for Performance."

Optimizing Parameters in the Application Configuration File

This section discusses how you can modify values for the parameters specified in your Siebel application configuration file to optimize the performance of a 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, 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.

AutoStopDB

Make sure that AutoStopDB is set to FALSE so that the SQL Anywhere database engine continues to execute after the user exits the Siebel application. This reduces the time required to restart the Siebel application at a later time. If AutoStopDB is set to TRUE, the SQL Anywhere database engine automatically closes down when you exit a Siebel application.

You set the AutoStopDBIn parameter in the `[Local]` section of your application configuration file. The default value is FALSE.

Allocating Memory to the SQL Anywhere Database Engine Cache

The amount of memory (especially cache) made available to the SQL Anywhere database engine is one of the major factors that can influence performance. The SQL Anywhere database engine uses memory for many purposes, but one of the major uses is to hold data that is accessed repeatedly, so that is does not have to retrieve data from the database each time it is needed.

You can configure how much memory is available to the cache by setting a value for the -c command line option in the ConnectString parameter of the siebel.cfg file. For example, the following entry:

```
-c15m -ch25m
```

allocates a minimum of 15 MB of memory to the cache. The value for the parameter (ch) indicates that the amount of memory allocated to the cache can be increased upwards to a maximum of 25 MB.

By default, the values for these parameters are expressed as a percentage of the total memory available. For example, the following entry:

```
-c5p -ch7p
```

specifies that a minimum of 5% of available memory be allocated to the cache memory and a maximum of 7%.

Allocating more memory to the memory cache of the SQL Anywhere database engine reduces the amount of memory that is available to other applications on the local machine.

As a guideline, use the difference between 80% of total machine memory and the amount of memory used by all applications on the machine during regular use. For example, if a local machine that has 512 MB of memory available uses 328 MB during regular use (after all applications including Siebel applications are loaded), then you can allocate 82 MB of memory to the cache of the SQL Anywhere database engine.

You should also conduct tests to determine an upper limit for the amount of memory that you can allocate to the SQL Anywhere database engine cache.

CAUTION: Do not increase the amount of memory allocated to the cache to a level that it results in paging. Paging is where memory utilization exceeds the total available memory and can cause reduced performance.

Sort Collation

The parameter `SortCollation` should be set to binary to optimize the retrieval of data from the local database. The `SortCollation` parameter is not a default part of the application configuration file. You have to manually add it to the configuration file of your Siebel application. You set the value of `SortCollation` in the `[Local]` section of your application configuration file.

For more information about this parameter, refer to the *Siebel System Administration Guide*. To determine the current status of SortCollation, see Alert 801 on Siebel SupportWeb.

Best Practice for Synchronization

This section lists some points that may help you optimize data synchronization between your Siebel Mobile Web Client and Siebel Remote Server. Note the following points:

- Synchronize frequently

  Synchronizing frequently reduces the number of transactions to transmit and commit for each synchronization session. The longer a user waits between synchronization sessions, the more data there is to send.

- Enable TrickleSync on the Siebel Mobile Web Client

  Each time 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.

- Disable Docking objects

Choosing an Appropriate Routing Model

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. If none of the supplied routing models are appropriate, contact Oracle's Siebel Expert Services, who will help you to develop a routing model that is appropriate to your environment.

12 Tuning Customer Configurations for Performance

This chapter discusses how you can avoid common performance-related problems in Siebel applications that stem from customer configuration done using Siebel Tools or Siebel scripting languages. It contains the following topics:

Application development information is also available in the following books on the *Siebel Bookshelf* and in *Siebel Tools Online Help*:

- *Configuring Siebel Business Applications*
- *Using Siebel Tools*
- *Siebel Developer's Reference*
- *Siebel Object Types Reference*
- *Siebel Object Interfaces Reference*
- *Siebel eScript Language Reference*
- *Siebel VB Language Reference*

General Best Practices for Customer Configurations

This section provides some general best practices for customer configuration using Siebel Tools.

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

The Siebel application 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 may have various potential performance pitfalls, the impact of which may be amplified in environments with large databases and wide data distribution across servers. Follow guidelines presented here and in other documentation to avoid such problems.

In addition to the topics in this section, see also:

- "Best Practices for Siebel Scripting" on page 160
- "Best Practices for Data Objects Layer" on page 164
- "Best Practices for Business Objects Layer" on page 169
- "Best Practices for User Interface Objects Layer" on page 173

Review information presented in *Configuring Siebel Business Applications* and other documentation on the *Siebel Bookshelf*, and other sources.

Miscellaneous Configuration Guidelines

The following are some miscellaneous configuration guidelines for maintaining optimal performance:

- **Avoid using sort specifications on non-indexed columns or joined columns.** For more information, see "Managing Database Indexes in Sorting and Searching" on page 165 and other relevant topics.
- **Avoid the use of case insensitivity.** Use of case-insensitive queries can significantly increase the possibility of performance issues due to the additional complexity required at the database level to support case-insensitive database operations.

  Prior to enabling case insensitivity, a thorough review of business requirements and performance criteria is highly recommended. In addition, if the feature is enabled, a performance test should be conducted with a full copy of the production database. The severity of the performance impact increases with the complexity of the configuration and the size of the production database.

  It is also recommended that Siebel Expert Services be engaged to optimize the configuration and review requirements. Case insensitivity is a database platform constraint and should also be reviewed with the database platform vendor.

  For more information about configuring case insensitivity for an application or for specified fields, see the *Siebel Applications Administration Guide*.
- **Limit the use of case insensitivity for queries.** Case-sensitive searches perform better than case-insensitive queries. Siebel applications are case-sensitive by default. You can enable case insensitivity either for the entire application or for specified fields. In general, the larger the database, and the larger the number of records returned by a case-insensitive query, the more that overall database performance is affected. Overall performance is also affected by the number of users who perform case-insensitive queries. End users can also force case-sensitive or case-insensitive queries.

  For more information about configuring case sensitivity for an application or for specified fields, see the *Siebel Applications Administration Guide*.
- **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.
- **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 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 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" on page 170.

  In particular, reduce the number of fields displayed in the master applet on related views. The information is static and may 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 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 (fields for which the Required user property is TRUE) in your business components can improve performance. See also "Limiting the Number of Active Fields" on page 170.
- **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 may also slow performance. If TRUE, 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 may be used for joined tables, with a resulting savings in overhead, provided 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 may also provide guidance to end users on constructing appropriate queries.

- **Cache business services.** Cache business services that should be accessible at all times in a user session. To do this, set the Cache parameter to TRUE for each applicable Business Service object definition. Caching of business services has an impact on memory, as the services are cached per session. Make sure that only frequently accessed business services in a session are marked 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.

Then you test the application to determine bottlenecks, using realistic data volumes and distribution in your test environment. Focus your testing effort on the slowest, most important, and most highly configured views.

If a performance problem is detected in testing or production, your next step is to analyze the SQL statements being generated by Siebel applications. This is one of the most useful diagnostic tools available to you for performance analysis.

Specifying SQL Spooling in Siebel Developer Web Client

After making configuration changes in Siebel Tools, 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 the *Siebel Installation Guide* for the operating system you are using.

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 may 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:\Program Files\siebel\7.8\web client\bin\siebel.exe
/c D:\Program Files\siebel\7.8\web client\bin\enu\uagent.cfg /s siebel_sql.txt"
```

If you do not specify a path, the SQL trace file is created in the Siebel client root bin directory, such as "`D:\Program Files\siebel\7.8\web client\bin`".

- For the purpose of spooling SQL, you can create shortcuts for Siebel Developer Web Client to run customer applications such as Siebel eService. For example, the shortcut would point to the application configuration file eservice.cfg.
- You can enable SQL spooling for an Application Object Manager (AOM) component by setting the Object Manager SQL Log (ObjMgrSqlLog) parameter to 4 at the component level. For more information, see *Siebel System Administration Guide*.
- You can also 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

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, you should 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 should be easily distinguishable based on the tables they access.

Troubleshooting Performance Using SQL Query Plans

If you identify a problematic query in the SQL trace file, you can obtain more information about it using the database query tool provided with the RDBMS, such as SQL*Plus for Oracle.

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 an SQL Anywhere database, see "Example of Obtaining Query Plan" on page 160.

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 you make to improve query performance through the user interface. For more information, see "Managing Database Indexes in Sorting and Searching" on page 165.

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. 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 may 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, or 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" on page 172.

Example of Obtaining Query Plan

The following procedure shows an example of obtaining a query plan when running against a local SQL Anywhere database using the Siebel Mobile Web Client.

To obtain a query plan for an SQL statement in your trace file

1. Execute the Interactive SQL (dbisqlc.exe) program, located in the Siebel client installation directory (Siebel Mobile).
2. In order to analyze an SQL statement from the SQL trace file, copy the SQL statement and paste it into the Interactive SQL program's Command pane.
3. Replace bind variable references with the corresponding bind variable values.
4. Click the Execute button.

   The query runs against the local SQL Anywhere database. The Statistics pane provides analysis information.

SQL Queries Against Database Data

The database that underlies Siebel applications can be queried to obtain information on a read-only basis.

CAUTION: Update queries should never be directly performed on the Siebel Database. All data manipulation and restructuring should be performed through Siebel Tools or through the Siebel application.

Best Practices for Siebel Scripting

This section provides guidelines for Siebel scripting using Siebel eScript or Siebel VB, or for using declarative alternatives in place of scripts.

Using Declarative Alternatives to Siebel Scripting

Often, customers use scripts for data validation, responses to data changes, or other purposes that may best be addressed through declarative means: by defining properties or specifying business service method invocation using Siebel Tools.

Scripting is often unnecessary and should be minimized or avoided because it may introduce performance problems, add risk and complexity, require greater maintenance, and duplicate functionality already available in Siebel applications.

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 may 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" on page 172 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 may often be handled through scripting, may best be addressed through declarative means. Such mechanisms, many of which may 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*. For more information on many of the these topics, consult the *Siebel Bookshelf*.

Siebel Scripting Guidelines for Optimal Performance

This section 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*

The following are some guidelines for appropriate use of Siebel scripting:

- **Use declarative alternatives.** Generally, you should try all other possibilities before using scripting to accomplish a functional requirement. See also "Using Declarative Alternatives to Siebel Scripting" on page 160.
- **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, may reduce performance.

  For example, using Set/Get Profile attribute calls, or invoking multiple business service methods, may require more server round trips and lead to performance problems. Adding extra functionality to scripts that display messages may 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 should return simple values or a single record, and should 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—should 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 data does not change between invocations), you can cache a limited set of data within your scripts. (In some cases, you may 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, may 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, complex code involving many layers of object instantiation may 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 should 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. The best practice is to destroy objects in reverse order of creation—that is, destroy child objects before you destroy parent objects.

- **Verify your script is defined on the appropriate method.** A script that is not defined on the right method may have a performance impact. For example, if special code needs to be run at the record level when an insert or update is done, 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 your script is implemented in the right view.** A script that is not implemented in the right view may 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 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 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 if 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 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. Additional guidelines for using error handling in scripts are provided in Technical Note 514, located on Siebel SupportWeb.
- **Avoid nested query loops.** Nested query loops may involve a large number of subqueries and may 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 may 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." You should use it in place of references to active business objects and components.

  For example, in a business component event handler, you should use *this* in place of ActiveBusComp(), usage of which may 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)," +
  " 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 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 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 your scripts are fully tested and optimized, and are no more complex than required to meet your business needs.

Best Practices for Data Objects Layer

This section describes best practices for configuring selected elements in the data objects layer for optimal performance.

Multilingual LOVs Query and Cache Performance

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 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 may also manifest when client-side export is performed to automate this behavior, or anytime 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 on configuring MLOVs, see *Configuring Siebel Business Applications* and *Siebel Global Deployment Guide*.

Managing Database Indexes in Sorting and Searching

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 Server 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 may 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 may 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 repeated instances of values (homogeneity)—the less benefit the index offers relative to its costs.

For Boolean fields, indexes generally offer little value. Some performance benefit may 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 may significantly reduce performance for batch UPDATE and INSERT operations, such as are performed by Siebel EIM.

You should discuss any custom index requirements with Siebel Expert 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 wanted a sort specification that ordered contacts in first-name order, 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 may scan through all rows in the table rather than only those to be returned by the query.

The index needs to contain all 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), 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" on page 167.

NOTE: If a query or sort includes columns that are not denormalized to the intersection table, performance is likely to degrade, because indexes are not used.

Reusing Standard Columns

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 Siebel Expert 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, the data in the denormalized column can become out of sync with its source. This can result in a number of problems ranging from inconsistent sorting to corrupt data.

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), 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_INVL
OC_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_EX
T_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_INVL
OC_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_EX
T_SS_U1),T9(se)
```

Best Practices for Business Objects Layer

This section describes best practices for configuring selected elements in the business objects layer for optimal performance.

Using Cache Data Property to Improve Business Component Performance

To cache on the AOM the content of a business component for subsequent use in the same user session, the property Cache Data property should be set to TRUE for the business component.

Setting Cache Data to TRUE is appropriate for semi-static data that may 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 PickList Generic and Internal Product business components. (See "Using Properties to Improve Picklist Performance" on page 171.)

Cache Data should be FALSE for business components representing transactional data that may change within a user session.

Limiting the Number of Active Fields

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

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, may 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 may 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. This 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, 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 may be performed, impacting performance.
  - If the function is *not* supported in the RDBMS, the Siebel application may 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 the first case, the calculations can take place before the results are returned, while, in the second case, they have to be performed in memory (on the Application Object Manager or client).

  NOTE: Even if the calculated field is supported at the RDBMS level, there may be other reasons why a search specification on a calculated field may 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" on page 165.

Using Properties to Improve Picklist Performance

To cache the content of certain picklists for subsequent use in the same user session, the property Cache Data property should be set to TRUE for the PickList Generic business component. By default, this property is TRUE.

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

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 a 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 may 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

In most cases, the Check No Match property of a Multi Value Link object definition (used to implement Primary ID fields) should 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 should 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, a secondary query is performed to determine if there are child records in the MVG. If there are no child records, the Primary ID field is set to the value *NoMatchRowId*.
  - If a parent record's Primary ID field has the value *NoMatchRowId*, 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*.

Best Practices for User Interface Objects Layer

This section describes best practices for configuring selected elements in the user interface objects layer for optimal performance.

Addressing Performance Issues Related to Grid Layout

The grid layout feature allows developers to create effective and usable form applets for Siebel views. However, performance may 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 may reduce the number of adjacent spacer cells you require.
- Aligning stacked sets of labels consistently.

- Making labels the same height as the adjacent fields.
- Eliminating horizontal or vertical spacer cells 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 feature, see *Configuring Siebel Business Applications*.

Maintaining Performance When Using Applet Toggles

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 may 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 may 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 view layout 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 for Performance

This chapter describes tuning steps designed to improve the performance and scalability of your Siebel Enterprise installation. This chapter contains the following topics:

Before doing any of the procedures in this chapter, you must have completed the minimum necessary configuration steps described in the chapters about installing the Siebel Gateway Name Server and the Siebel Server contained in the *Siebel Installation Guide* for the operating system you are using.

For additional information about tuning and monitoring, see *Siebel System Monitoring and Diagnostics Guide* and *Siebel System Administration Guide*.

NOTE: Settings provided in this appendix are based on a controlled lab environment using a standard Siebel application, such as Siebel Call Center for Siebel Industry Applications. 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 platform.

Tuning Microsoft Windows for Enhanced Siebel Server Performance

This section describes how you can configure settings for your Microsoft Windows operating system to optimize the performance of Siebel applications.

Maximizing Data Throughput

Changing the setting for data throughput from *Maximize data throughput for file sharing* (default) to *Maximize data throughput for network applications* may result in the following benefits:

- Better Symmetrical Multi-Processing (SMP) scalability
- Improved networking performance
- Allocation of more physical memory for your Siebel applications

For more information on these settings, refer to 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 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 on how to configure this setting, refer to Microsoft's documentation.

NOTE: Each Siebel process (Application Object Manager) cannot use more than 2 GB of RAM.

Tuning the Siebel Server for All UNIX Platforms

For all Siebel Server machines running on supported UNIX platforms, setting the environment variables described in this section can help you manage your server resources appropriately and stay within appropriate CPU-usage limits.

Environment Variable for Siebel Assert Creation

For Siebel Server machines or Web server machines, 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.

This variable should be set to a non-zero value only if you are performing system diagnostics, and it should only be set in consultation with Siebel Technical Services.

For more information about this variable, see *Siebel System Monitoring and Diagnostics Guide*.

Environment Variable for Operating System Resource Limits

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 may include coredumpsize, cputime, filesize, descriptors, maxmemory, and others.

Environment Variable for Operating System Latches

If the total number of tasks on the Siebel Server is greater than 500, you should 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 or the total number of tasks on the Siebel Server is less than 500, you do not need to set these variables.

NOTE: Before changing these variables, stop the Siebel Server using the stop_server command, then run the cleansync utility. For more information about this utility, see Siebel SupportWeb.

Set SIEBEL_OSD_NLATCH and SIEBEL_OSD_LATCH on the Siebel Server machine based on the following formulas (define the variables in the applicable profile for the Siebel Server):

- SIEBEL_OSD_NLATCH = 7 * (cumulative MaxTasks for all components) + 1000
- SIEBEL_OSD_LATCH = 1.2 * (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 = 500 and, for WfProcMgr, MaxTasks = 100. In this example, parameter values should be as follows:

- SIEBEL_OSD_NLATCH = 5200 = 7 * [500 + 100] + 1000
- SIEBEL_OSD_LATCH = 720 = 1.2 * [500 + 100]

Tuning the Siebel Web Server Extension for All UNIX Platforms

You must tune the Siebel Web Server Extension (SWSE) to run Siebel applications on UNIX platforms.

To tune the SWSE for UNIX platforms

1. In the SWSE installation directory, navigate to the bin subdirectory.
2. Using a text editor such as vi, open the eapps.cfg file for editing.
3. Set the appropriate AnonUserName user names and passwords. This will depend on your user authentication strategy. For more information, see *Siebel Security Guide*.
4. Set GuestSessionTimeout to 60.

   NOTE: This configuration is appropriate for application scenarios where users browse without logging in.
5. Restart the Web server for these changes to take effect.

Tuning Siebel Business Applications for AIX

This section provides instructions for configuring and tuning Web servers, OS settings, and Siebel Enterprise components so you can run Siebel applications on AIX. It includes the following topics:

■ "Tuning Kernel Settings for AIX" on page 181

Tuning the IBM HTTP Server for AIX

This section describes recommended values for environment variables that are optimized for scalability and performance on IBM HTTP Server (IHS) Web server. You can further adjust these settings at your discretion to optimize the performance of your Web server.

The following environment variables are set in *Webserver_Root*/bin/startapa, where *Webserver_Root* is the root directory in which your Web server is installed:

```
export AIXTHREAD_SCOPE=S
export AIXTHREAD_MNRATIO=1:1
export AIXTHREAD_MUTEX_DEBUG=OFF
export AIXTHREAD_RWLOCK_DEBUG=OFF
export AIXTHREAD_COND_DEBUG=OFF
export CORE_NAMING=true
export YIELDLOOPTIME=number_of_CPUs_on_Web_server_machine
export SPINLOOPTIME=1000
export MALLOCMULTIHEAP=heaps:number_of_CPUs_on_Web_server_machine,considersize
export MALLOCTYPE=buckets
export LDR_CNTRL=IGNOREUNLOAD@LOADPUBLIC@PREREAD_SHLIB@MAXDATA=0x60000000
```

For the MALLOCMULTIHEAP and YIELDLOOPTIME parameters, the values should include the number of CPUs on the Web server machine. For example, if there are two CPUs, these parameters should be defined as follows:

```
export MALLOCMULTIHEAP=heaps:2,considersize
export YIELDLOOPTIME=2
```

To set the number of threads for IBM HTTP Server

1 Using a text editor, set values for parameters in the `workers.c` section of the file *Web_server_install*`/conf/httpd.conf`, where *Web_server_install* is the root directory in which your Web server is installed. Set the parameter values as follows:

`ThreadLimit`	*N*
`StartServers`	1
`ServerLimit`	1
`MaxClients`	*N*
`MinSpareThreads`	1
`MaxSpareThreads`	*N*
`ThreadsPerChild`	*N*
`MaxRequestsPerChild`	0

where:

N = A value 1 or 1.2 * maximum number of concurrent users (threads). The value for the applicable parameters can be set equal to or less than 1.2 * the number of concurrent users the Web server must support.

High values for the MaxClients or ThreadLimit parameters can increase memory usage. If you want to reduce memory usage, it is recommended that you reduce the values that you set for these parameters.

By default, the Web server provides persistent connections. Consider setting the KeepAlive parameter equal to Off to stop the Web server providing persistent connections. This frees connections for reuse. However, doing this can incur a cost as memory will be used to close and set up new TCP/IP connections.

Alternatively, you can reduce the value for the KeepAliveTimeout parameter so that a thread is reused more quickly.

2 In the file `httpd.conf`, also set the following values:

- The value for `ServerName` *must* match the Primary Internet Address you used in installing SWSE.
- Change the values for `User` and `Group` to a valid machine user and group:
  - Ideally, the user ID should have no privileges that allow access to files other than those used by the Siebel application. This user should, however, have full access rights (read, write, execute) to the SWSE installation directory and its subdirectories.
  - It is recommended that the group should be created specifically for running this server.

  CAUTION: For security reasons, it is recommended that you do not use root for `User` or `Group`.
- The value for `UseCanonicalName` is recommended to be set to `Off`. It *must* be set to `Off` if you are load-balancing your Web servers.

- If you are not using the CGI functionality of IHS, you may want to comment out the line that loads the CGI module. Doing so will make tracking IHS processes simpler, because there will be always one child process. The line is as follows:

  ```
  LoadModule cgid_module modules/mod_cgid.so
  ```

Tuning the Siebel Server 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. Table 5 on page 180 and Table 6 on page 180 describe each of these environment variables and their recommended settings.

Table 5. Environment Variables Used for Optimization in $*SIEBEL_ROOT*/siebenv

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 should 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.

Table 6. Environment Variables Used for Optimization in $*SIEBEL_ROOT*/bin/siebmtshw

Environment Variable	Value	Description
SPINLOOPTIME	1000	Controls the number of times to retry a busy lock before yielding to another processor.
YIELDLOOPTIME	4	Controls the number of times to yield the processor before blocking on a busy lock (only for libpthreads). Set this variable, at the minimum, to the number of CPUs.

Table 6. Environment Variables Used for Optimization in $*SIEBEL_ROOT*/bin/siebmtshw

Environment Variable	Value	Description
MALLOCTYPE	buckets	Malloc buckets provide an optional buckets-based extension of the default allocator. This feature improves malloc performance for applications that issue large numbers of small allocation requests. When malloc buckets are enabled, allocation requests that fall within a predefined range of block sizes are processed by malloc buckets. All other requests are processed in the usual manner by the default allocator.
MALLOCMULTIHEAP	heaps:*n*	Controls the number of heaps within the process private segment. *n* should be equal to the number of processors on the server.
LDR_CNTRL	IGNOREUNLOAD@LOADPUBLIC @PREREAD_SHLIB@MAXDATA= 0x60000000	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 needing the extra memory. The MAXDATA value reserves six 256-MB segments for all executables launched from this environment, and overrides the default executable setting. The default depends on the executable. With the default value, a Siebel component can support a maximum value of 5000 for the MaxTasks parameter. With this value, MaxTasks can be set as high as 9000. Depending on the environment, you may reserve up to a maximum of seven segments. If it is not possible to use that many segments, the Siebel Server will terminate very early.

Tuning Kernel Settings for AIX

There are a number of AIX kernel settings you can tune for optimal Siebel Server or Web server performance under AIX. These include the Virtual Memory Management and TCP settings. You must have root privileges to modify these settings. (On AIX 5.2, kernel settings can alternatively be tuned using vmo rather than vmtune.)

For more information about AIX kernel settings, refer to your operating system vendor's documentation.

To change the kernel settings using vmtune

1. Using a text editor such as vi, open the /etc/rc.net file for editing.
2. Modify the vmtune settings, as follows:

```
if [ -f /usr/samples/kernel/vmtune ] ; then

   /usr/samples/kernel/vmtune -p 5 -P 8 -f 720 -F 768 -b 200 -s 1
```

   In providing values for minfree (-f for vmtune) and maxfree (-F for vmtune), use the following formulas:

   - minfree = *number_of_CPUs* * 120 = 6 * 120 = 720
   - maxfree = *number_of_CPUs* * (120 + *maxpgahead*) = 6 * (120 + 8) = 768

   where:

   number_of_CPUs = the number of CPUs on the AIX server you are tuning (for example, 6)

   maxpgahead = the value of the maxpgahead (-R for vmtune) parameter: for example, 8)

3. Modify the network options, as follows:

```
if [ -f /usr/sbin/no ] ; then

   /usr/sbin/no -a rfc1323=1

   /usr/sbin/no -a tcp_sendspace=24576

   /usr/sbin/no -a tcp_recvspace=24576

   /usr/sbin/no -a rfc2414=1

   /usr/sbin/no -a tcp_init_window=3

   /usr/sbin/no -a use_isno=0

   /usr/sbin/no -a tcp_nagle_limit=0
```

4. Check the settings for all User Limits (ulimit) and make sure that they are set to -1 (unlimited), as follows:

```
ulimit –a
```

   NOTE: To change the set limits, update the /etc/security/limits file by changing all ulimit parameter values to –1 (unlimited).

5. Save your changes and exit the editor.
6. Restart the server machine to have the new settings take effect.

Tuning Siebel Business Applications for Solaris

This section provides instructions for configuring and tuning Web servers, OS settings, and Siebel Enterprise components so you can run Siebel applications on Solaris. It includes the following topics:

Tuning the Sun Java System Web Server for Solaris

If you have a busy Web server, some of your users might experience difficulty connecting to your Web server. To address this issue, change the `tcp_conn_req_max_q` and `tcp_conn_req_max_q0` default values, using the `ndd` command. For details on how to use the `ndd` command, see "Tuning AOM Instances for Solaris" on page 186.

You also should tune the Sun Java System Web Server for optimal performance using the following procedure.

To tune the Sun Java System Web Server

1. Using a text editor such as `vi`, open the file *`Webserver_Root`*`/config/magnus.conf`, where *`Webserver_Root`* is the root path of the Sun Java System Web Server.
2. Set the parameter `RqThrottle` to 1200.

   The `RqThrottle` parameter specifies the maximum number of simultaneous transactions the Web server can handle. The default value is 512. By changing this value to 2048, you can minimize latencies for the transactions that are performed.
3. Add or modify the `MaxKeepAliveConnections` parameter, setting its value to 1000. The default value is 200.
4. Save your modifications to the `magnus.conf` file.
5. Restart the Web server.

   After making the changes above to the Sun Java System Web Server parameters, change the following parameters on the workstation hosting the Sun Java System Web Server.
6. Open the `/etc/system` file for editing.

7 Set the following Solaris system parameters:

Parameter	Scope	Default Value	Tuned Value	Comments
rlim_fd_max	/etc/system	1024	8192	Process open file descriptors limit; should account for the expected load (for the associated sockets, files, and pipes, if any).
rlim_fd_cur	/etc/system	64	8192	

8 Restart the workstation hosting the Sun Java System Web Server.

Tuning Kernel Settings for Solaris

To run Siebel Servers or Web servers in a Solaris environment, you need to set Solaris kernel parameters to specific recommended values for particular releases of Solaris servers. To learn the specific parameter recommendations for Siebel Servers or Web servers running on Solaris, contact Oracle's Siebel Expert Services.

There are a number of Solaris kernel parameter settings that significantly affect performance of Siebel applications in general, and the Siebel Server in particular. These include parameters that govern elements such as file descriptors, stack size, memory, and semaphores.

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 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 Solaris kernel parameters, or lower, to run a Siebel Server in a 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 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 = 8192
set rlim_fd_max = 8192
```

3 Add or modify the following lines, which are shared memory settings. In the first line, select either Solaris 32-bit or 64-bit, respectively:

```
set shmsys:shminfo_shmmax = 0xffffffff [or] 0xffffffffffffffff
set shmsys:shminfo_shmmni = 1024
set shmsys:shminfo_shmseg = 1024
```

4 Add or modify the following lines, which are message queue settings:

```
set msgsys:msginfo_msgmax = 4096
```

5 Add or modify the following lines, which are semaphore settings:

```
set semsys:seminfo_semaem = 16384
set semsys:seminfo_semopm = 100
set semsys:seminfo_semmni = 4096
set semsys:seminfo_semmns = 16384
set semsys:seminfo_semmnu = 4096
set semsys:seminfo_semume = 2500
set semsys:seminfo_semmsl = 500
```

6 Save your changes and exit the editor.

7 Restart the server machine to have the new settings take effect.

Maximizing Siebel Server Performance for Solaris

To gain the maximum CPU performance for your Siebel Server when running on Solaris, use the Multiple Page Size Support (MPSS) with optimal configuration of 4 MB heap size and 64 KB stack size, as outlined in the following procedure.

To set up MPSS with optimal configuration of 4 MB heap size and 64 KB stack size

1 Using an editor such as `vi`, open the `/etc/system` file for editing.

2 Add the following line to the file:

```
set kernel_cage_enable=1
```

3 Reboot the server.

4 Create a configuration file (mpss.cfg) for MPSS configuration with the following line in the file:

```
sieb*:4M:64K
```

where 4M is the heap size (4 MB) and 64K is the stack size (64 KB).

5 Add the following lines to the `$`*`Siebel_Root`*`/siebsrvr/bin/siebmtshw` file:

```
LD_PRELOAD=/usr/lib/mpss.so.1
```

`MPSSCFGFILE`=Full path, including the file name, to the MPSS configuration file created in Step 4.

`MPSSERRFILE`=Full path, including the file name, to the MPSS error log you want to be generated in case of any errors.

```
export LD_PRELOAD MPSSCFGFILE MPSSERRFILE
```

Tuning AOM Instances for Solaris

Solaris machines running more than 50 Application Object Manager instances (multithreaded processes for AOM) may 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, 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 Business Applications for HP-UX

This section provides instructions for configuring and tuning Web servers, OS settings, and Siebel Enterprise components so you can run Siebel applications on HP-UX. It includes the following topics:

Tuning the HP Apache Web Server for HP-UX

This section provides recommended initial settings for HP Apache Web Server environment variables. You can further modify these settings at your discretion to optimize the performance of your Web server.

The default `ThreadLimit` for the HP Apache Web Server is 64, but it can be set it to a much higher number. The highest setting depends on the kernel settings. `ThreadsPerChild` and `MaxClients` are related directives.

- `ThreadLimit = 20000` is the maximum value supported by the HP Apache Web Server. You can reset this to the number your system supports.

  NOTE: The `ThreadLimit` directive must be executed before other directives.

- `ThreadsPerChild` = Number of threads per child. Cannot exceed `ThreadLimit`.
- `MaxClients` = Maximum connection. Cannot exceed `ThreadsPerChild`.

To set the number of threads for HP Apache Web Server

1. Using a text editor, set values for parameters in the `workers.c` section of the file *`web_server_install`*`/conf/httpd.conf`, where *`web_server_install`* is the root directory in which your Web server is installed. Set the parameter values as follows:

   ```
   ThreadLimit          N
   StartServers         1
   ServerLimit          1
   MaxClients           N
   MinSpareThreads      1
   MaxSpareThreads      N
   ThreadsPerChild      N
   MaxRequestsPerChild  0
   ```

   where:

   N = A value similar to 1.2 or 1.5 * maximum number of concurrent users (threads). The value for the applicable parameters *must* be greater than the number of concurrent users the Web server must support. However, setting parameter values higher than what is described here will consume additional memory unnecessarily.

   NOTE: If you are not using multiplex sessions, make sure the kernel parameter `max_thread_proc` is set to a number greater than 2*N*.

2. Change the values for `User` and `Group` to a valid machine user and group:

   - Ideally, the user ID should have no privileges that allow access to files other than those used by the Siebel application. This user should, however, have full access rights (read, write, execute) to the SWSE installation directory and its subdirectories.
   - It is recommended that the group should be created specifically for running this server.

   CAUTION: For security reasons, it is recommended not to use root for `User` or `Group`.

3 Set `MaxKeepAliveRequests` to 0.

Tuning Kernel Settings 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
maxusers                  128 - 128
msgmap                   4098 - (NPROC+2)
msgmni                   4096 - (NPROC)
msgtql                   4096 - (NPROC)
ncallout                 8000 - 8000
nclist                   2148 - (100+16*MAXUSERS)
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)
semmap                   1026 - 1026
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

Siebel Business Applications 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, the system will respond:

```
global privileges: RTSCHED
```

14 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:

For information on how to analyze the data that the Siebel ARM feature collects, see "Analyzing Siebel ARM Data" on page 199.

NOTE: Additional diagnostic and monitoring tools are available using the Siebel Management Framework. The deployment of the Siebel Management Framework in your Siebel environment is optional. For more detailed information about the Siebel Management Framework, see the *Siebel System Monitoring and Diagnostics Guide*.

About Siebel Application Response Measurement

Siebel ARM is a framework for capturing critical performance data in Siebel Business Applications. This data is saved in binary file format. 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:

- **Web Server Time.** Time duration a request has spent on the Web server.
- **Infra-Network Time.** Time duration between a request from the Web server and the Siebel Server (including the network time).
- **Siebel Server Time.** Time duration for the request to be processed by the Siebel Server and 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 following areas are listed as they appear in Siebel ARM output; the name in parenthesis after the area name represents the area symbol, which also appears in Siebel ARM output.

- SARM Framework (SARM)
- Siebel Repository (SRF)

- Web Engine (SWE)
- Build Web Page (SWEPAGE)
- Web Server Plugin (SWSE)
- Database Connector (DBC)
- Application Server (INFRA)
- Workflow (WORKFLOW)
- eScripts (SCRIPT)
- Request Manager (SRM)
- Request Broker (SRB)
- File System Manager (FSM)
- Business Service (BUSSRVC)
- Email Response (EMR)
- Security / Authentication (SEC)
- Object Manager (OBJMGR)
- Assignment Manager (AM)
- Fulfillment Engine (FSFULFILL)
- Preventative Maintenance Engine (FSPREVMNT)
- Siebel Loyalty (LOY)
- Handheld Sync (HHSYNC)
- SmartScript (SMARTSCRIPT)
- Siebel Anywhere (SIEBANYWHERE)
- Communications Channel Manager (CSMM)
- Communications Server Service (CSS)
- Customer/Order Management - Configurator (COMCFG)
- EAI Transports (EAITRANSP)
- MWC Profiler (MWC)[1]
- Communications Outbound Manager (COM)
- Universal Inbox (UINBOX)

1. MWC = Mobile Web Client

Each of the previous areas contain one or more subareas, which further define the timing and performance of their respective area. The amount of areas and subareas present in Siebel ARM files is dependent on the granularity level. This level is configured by the parameter SARM Granularity Level. For more information on this parameter, see "About Siebel ARM Parameters and Variables" on page 192.

For more information about the format of Siebel ARM files, see "About Siebel ARM Files" on page 199.

About Siebel ARM Parameters and Variables

The following parameters on the Siebel Server and environment variables on the Web 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 Table 7 on page 192 for a listing of each Siebel ARM parameter and its equivalent environment variable. Descriptions of each parameter and environment variable follow the table.

Table 7. Siebel ARM Parameters and Environment Variables

Parameter Display Name	Parameter Alias	Environment Variable Name
SARM Granularity Level	SARMLevel	SIEBEL_SARMLevel
SARM Buffer Size	SARMBufferSize	SIEBEL_SARMBufferSize

Table 7. Siebel ARM Parameters and Environment Variables

Parameter Display Name	Parameter Alias	Environment Variable Name
SARM Period	SARMPeriod	SIEBEL_SARMPeriod
SARM Max Number of files	SARMMaxFiles	SIEBEL_SARMMaxFiles
SARM Data File Size	SARMFileSize	SIEBEL_SARMFileSize

For details on enabling Siebel ARM using these parameters and variables, see "Enabling and Configuring Siebel ARM" on page 195.

SARM Granularity Level

Specifies the amount of response measurement detail logged to Siebel ARM files and effectively enables or disables the Siebel ARM feature. This parameter or environment variable 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 SARM Buffer Size Siebel ARM outputs the stored data to file on a physical disk. The SARM Buffer Size parameter or environment variable is specified in bytes. The default value is 5,000,000 bytes (approximately 5 MB). The valid settings range 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 parameter or environment variable SARM Period. The setting of this parameter may determine the size of the data saved to file rather than the threshold value defined by SARM Buffer Size.

For example, if SARMBufferSize 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 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 parameter is specified in minutes, and has a default value of 3 minutes. The valid settings for this parameter range from 1 minute to 60 minutes.

NOTE: Only use SARM Period to output Siebel Server performance data based on elapsed time. Siebel ARM outputs Web server performance data based only on the SARM Buffer Size value.

See the description for SARM Buffer Size for information on outputting data from memory based on size of data in memory.

SARM Max Number of Files

Specifies the maximum number of Siebel ARM files created per component instance. The default value is four, and there is no Siebel-specified upper limit to the number of files Siebel ARM creates. (The parameter or environment variable 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 1 plus the value of SARM Max Number of Files. That is, Siebel ARM removes the oldest file for that process only after the SARM Max Number of Files-plus-1 file reaches SARM Data File Size.

See the description for SARM Data Size for an example on how to calculate memory usage using these parameters or environment variables.

SARM Data File Size

Specifies how large a file becomes before Siebel ARM stores data in a new file on the physical disk. The parameter is specified in bytes. The default value is 15000000 bytes (15 MB), and there is no Siebel-specified 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. (The parameter or environment variable 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, there are SARM Max Number of Files of size SARM Data File Size), Siebel ARM removes the first (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 + 1 times SARM Data File Size bytes. 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, 15MB per file, times 5 files per process, times 5 processes (instances of component).

Enabling and Configuring Siebel ARM

Enabling and configuring Siebel Application Response Measurement (Siebel ARM) involves two tasks:

- Setting Siebel ARM parameters on the Siebel Server.
- Setting Siebel ARM environment variables on the Web server.

By default, Siebel ARM is disabled.

Setting Siebel ARM Parameters on the Siebel Server

Perform the following procedure to enable and configure Siebel ARM on the Siebel Server.

NOTE: If the Siebel ARM parameters are not visible, make sure the parameter Show Advanced Objects (alias ShowAdvancedObjects) is set to TRUE for the server component Server Manager (alias ServerMgr).

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. For further information on this parameter and its settings, see "About Siebel ARM Parameters and Variables" on page 192.

   You can enable Siebel ARM at either the enterprise, Siebel Server, or server component level.

2. Set the other Siebel ARM-related parameters to configure the Siebel ARM file characteristics on the Siebel Server. For further information on these parameters, see "About Siebel ARM Parameters and Variables" on page 192.

   You can configure Siebel ARM at the Siebel Server or server component level.

For further information on setting Siebel Server parameters using the Server Manager GUI or command-line interface, and for background information on parameter administration, see *Siebel System Administration Guide*.

Setting Siebel ARM Environment Variables on the Web Server

Perform the following procedure to enable and configure Siebel ARM on the machine hosting the Web server.

To enable and configure Siebel ARM on the Web server

1. Set the environment variable SIEBEL_SARMLevel to a value of 1 or 2 to enable Siebel ARM on the machine hosting the Web Server. For further information on this parameter and its settings, see the description for SARM Granularity Level in "About Siebel ARM Parameters and Variables" on page 192.
2. Set the other Siebel ARM-related environment variables to configure the Siebel ARM file characteristics on the machine hosting the Web server. For further information on these parameters, see "About Siebel ARM Parameters and Variables" on page 192.

For further information on setting environment variables on both Windows and UNIX, see *Siebel System Administration Guide*.

Best Practices for Siebel ARM

Review the following information as recommendations of best practice when converting Siebel ARM files.

- Set the Siebel ARM granularity level to level 1 for monitoring production deployments; set the Siebel ARM 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 may be useful when enabling Siebel ARM for use with other third-party ARM tools.
- Make sure 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 details on this process, see the descriptions for SARM Data File Size and SARM Period in "About Siebel ARM Parameters and Variables" on page 192.
- Change the value of the SARM Memory Size Limit (alias SARMMaxMemory) or SARM Period (alias SARMPeriod) to a lower setting if the Siebel ARM files remain empty on a consistent basis. For details on this process, see the descriptions for SARM Data File Size and SARM Period in "About Siebel ARM Parameters and Variables" on page 192.
- Make sure the Siebel ARM file name and path name, as necessary, are correct when referencing the Siebel ARM files in the commands.
- If the Siebel ARM Analyzer Tool cannot convert large Siebel ARM files or the output file is too large, split the Siebel ARM file by using the –p flag with the Siebel ARM Analyzer Tool. For further information on the –p flag, see Table 8 on page 201.
- 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 details on the number of files saved, see the description for SARM Max Number of Files in "About Siebel ARM Parameters and Variables" on page 192.)

  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 trace data identifies each request 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 details on running this analysis, see "Running Siebel ARM Data CSV Conversion" on page 204.
- Compile performance aggregation data over a period of time to determine a trend analysis.

15 Analyzing Siebel ARM Data

This chapter describes how to analyze Siebel ARM data. It includes the following topics:

For more information about the features of Siebel ARM, including how to collect Siebel ARM data, see Chapter 14, "Monitoring Siebel Application Performance with Siebel ARM."

NOTE: Additional diagnostic and monitoring tools are available using the Siebel Management Framework. The deployment of the Siebel Management Framework in your Siebel environment is optional. For more detailed information about the Siebel Management Framework, see the *Siebel System Monitoring and Diagnostics Guide*.

About Siebel ARM Files

When enabled, the Siebel ARM feature saves binary Siebel ARM files in the:

- Siebel Server log subdirectory on Windows: *SIEBSRVR_ROOT*\log
- Siebel Server log subdirectory on UNIX: *SIEBSRVR_ROOT*/enterprises/*EnterpriseServerName*/*SiebelServerName*/log
- Siebel Web Server Extension log subdirectory: *SWSE_ROOT*\log.

For information on the Siebel ARM feature, see "About Siebel Application Response Measurement" on page 191.

The Siebel ARM feature names the binary data files as in the following example:

```
T200401081744_P001768_N0006.sarm
```

where:

- T = Constant value, indicating timing convention information follows.
- 200401081744 = Indicates date and time of Siebel ARM file. This example indicates this file was saved on January 8th, 2004 at 17:44.
- P = Constant value, indicating process ID information follows.
- 001768 = Indicates the process ID on which Siebel ARM collects data.
- N = Constant value, indicating Siebel ARM ID information follows.
- 0006 = Indicates 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` = Siebel ARM file extension.

NOTE: The Siebel ARM feature creates an empty Siebel ARM file on the Web server before populating it with data. It begins storing data to these files after the feature reaches the value of the SARM Data File Size parameter. For details on this process, see parameter descriptions in "About Siebel ARM Parameters and Variables" on page 192.

To analyze the data contained in the binary Siebel ARM files, you must use one of the following tools which present the collected data in a readable format:

- Siebel ARM Analyzer Tool

  The Siebel ARM Analyzer Tool is a command-line tool that allows you analyze the performance data collected by Siebel ARM. For more information on the Siebel ARM Analyzer Tool, see "Analyzing Siebel ARM Files using the Siebel ARM Analyzer Tool" on page 200.

- Siebel ARM Query Tool

  The Siebel ARM Query Tool is a command-line tool that allows you analyze the performance data collected by Siebel ARM. For more information on the Siebel ARM Query Tool, see "Analyzing Siebel ARM Files using the Siebel ARM Query Tool" on page 218.

- Siebel Diagnostic Tool

  This tool allows you analyze the performance data collected by Siebel ARM using a graphical user interface. For more information about this tool, see the *Siebel System Monitoring and Diagnostics Guide*

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. For more information about this tool, see "About Siebel ARM Analyzer Tool" on page 201.

The following sections describe how to generate different types of analysis output using the Siebel ARM Analyzer Tool:

- "Running Performance Aggregation Analysis" on page 202
- "Running Call Graph Generation" on page 202
- "Running User Session Trace" on page 203
- "Running Siebel ARM Data CSV Conversion" on page 204

The following sections 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 formats based on the type of conversion analysis. For more information, see:

- "About Call Graph Generation Analysis and Data" on page 213
- "About User Session Trace Analysis and Data" on page 215
- "About Siebel ARM to CSV Conversion Data" on page 217

About 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 different types of output that this tool can generate, see "Analyzing Siebel ARM Files using the Siebel ARM Analyzer Tool" on page 200.

This command-line utility resides in the bin subdirectory (BIN) of the Siebel Server root directory as the executable program `sarmanalyzer.exe` on Microsoft Windows or sarmanalyzer on UNIX. Table 8 on page 201 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. For information on this flag, see Table 8 on page 201.

Table 8. Siebel ARM Analyzer Tool Flags

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 details, see "Running Performance Aggregation Analysis" on page 202.
`-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 and the arguments 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 arguments AREA or DETAILS when running a performance aggregation analysis. For further information on this analysis, see "Running Performance Aggregation Analysis" on page 202.
`-i`	Use this flag with a directory argument when running a user session trace analysis. For further information on this analysis, see "Running User Session Trace" on page 203.
`-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 MB 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

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" on page 205.

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 = The name and path of the XML output file.

   aggregate_argument = Either AREA or DETAILS depending on which area you want the Siebel ARM post-process tools to aggregate data from. For further information, see "About Performance Aggregation Analysis and Data" on page 205.

   sarm_file_name.sarm = 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 further information on analyzing the performance aggregation analysis XML output, see "About Performance Aggregation Analysis and Data" on page 205.

For further information on running the Siebel ARM Analyzer Tool, see "Analyzing Siebel ARM Files using the Siebel ARM Analyzer Tool" on page 200 and "About Siebel ARM Analyzer Tool" on page 201.

Running Call Graph Generation

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" on page 213.

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
```

where:

output_file_name.xml = The name and path of the XML output file.

-d xml = Identifies the call graph generation analysis.

sarm_file_name.sarm = The name and path of the binary Siebel ARM file.

3 Review the XML output in the file named *output_file_name*.xml. For further information on analyzing the call graph analysis XML output, see "About Call Graph Generation Analysis and Data" on page 213.

For further information on running the Siebel ARM Analyzer Tool, see "Analyzing Siebel ARM Files using the Siebel ARM Analyzer Tool" on page 200.

Running User Session Trace

Use the following procedure to obtain user session trace analysis output. Before running this analysis, manually collect Siebel Server and Web server Siebel ARM files and store 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" on page 215.

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 = The name and path of the XML output file.
- *user_name* = The User ID of the session you want to trace.
- *SARM_File_Directory* = The directory containing the Siebel ARM files of the Web Server and the Siebel Server.
- *start_time* = Optionally set this 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* = Optionally set this 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 further information on analyzing user session trace XML output, see "About User Session Trace Analysis and Data" on page 215.

For further information on running the Siebel ARM Analyzer Tool, see "Analyzing Siebel ARM Files using the Siebel ARM Analyzer Tool" on page 200.

Running Siebel ARM Data CSV Conversion

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" on page 217.

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 = The name and path of the CSV output file.

-d csv = Identifies the Siebel ARM data CSV conversion analysis.

sarm_file_name.sarm = 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 further information on analyzing CSV data, see "About Siebel ARM to CSV Conversion Data" on page 217.

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 on this flag, see Table 8 on page 201.

For further information on running the Siebel ARM Analyzer Tool, see "Analyzing Siebel ARM Files using the Siebel ARM Analyzer Tool" on page 200.

About Siebel ARM Analyzer Output Files

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 details on converting Siebel ARM files and running the Siebel ARM Analyzer Tool, see "Analyzing Siebel ARM Files using the Siebel ARM Analyzer Tool" on page 200.

Use an XML editor or Web browser to view the XML output files, which result from a number of 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

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" on page 191.

For details on creating this format of Siebel ARM output, see "Running Performance Aggregation Analysis" on page 202.

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 is dependent on the argument used for the –a flag when performing the analysis (either AREA or DETAILS) and the setting for the SARM Granularity Level parameter. For information on this parameter, see "About Siebel ARM Parameters and Variables" on page 192.

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 running the analysis, 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 on each of the tags, see Table 9.

Table 9. Performance Aggregation Analysis Tags

Tag	Description
Area	Specifies performance data captured for a specific area of the Siebel ARM architecture. There may be one or more areas captured with performance data. For further information on Siebel ARM areas, see "About Siebel Application Response Measurement" on page 191.
Name	Name of the area containing performance data. For a listing of area names, see "About Siebel Application Response Measurement" on page 191.
Symbol	Symbol of the area containing performance data. For a listing of symbol names, see "About Siebel Application Response Measurement" on page 191.
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 an Application Object Manager (AOM) 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 AOM 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.

Table 9. Performance Aggregation Analysis Tags

Tag	Description
ResponseTime	Specifies the time spent for a request to enter and exit an instrumentation area (layer) including calls to other child areas. 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 further information on Siebel ARM node tags, see "About Call Graph Generation Analysis and Data" on page 213. ■ +<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 further information on Siebel ARM node tags, see "About Call Graph Generation Analysis and Data" on page 213.
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 non-recursive 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.

Table 9. Performance Aggregation Analysis Tags

Tag	Description
`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. ■ `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 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. **NOTE:** For further information on Siebel ARM node tags, see "About Call Graph Generation Analysis and Data" on page 213.

Table 9. Performance Aggregation Analysis Tags

Tag	Description
`ExclusiveMemory`	Specifies 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 bya 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. **NOTE:** For further information on Siebel ARM node tags, see "About Call Graph Generation Analysis and Data" on page 213.

Table 9. Performance Aggregation Analysis Tags

Tag	Description
SubArea	Specifies performance data captured for a specific subarea of the given area. There may 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. ■ +<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 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 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.

Table 9. Performance Aggregation Analysis Tags

Tag	Description
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 may 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 futher 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 may 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. ■ +<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 futher 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

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 on instrumented areas, see "About Siebel Application Response Measurement" on page 191.

For details on creating this format of Siebel ARM output, see "Running Call Graph Generation" on page 202.

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 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. For descriptions on each of the tags, see Table 10.

```
<SarmNode>
   <SarmID>
   <TypeLevel>
   <RootID>
   <ParentSARMID>
   <ParentTimeID>
   <ParentProcID>
   <AreaCodeSymbol>
   <AreaDescription>
   <SubAreaCodeSymbol>
   <SubAreaDescription>
   <Count>
   <Duration>
   <PooledMemoryUsage>
   <PooledMemoryCalls>
   <SystemMemoryUsage>
   <SystemMemoryCalls>
   <AppInt1>
   <AppInt2>
   <AppString1>
   <AppString2>
```

```
    +<ChildNode>
</SarmNode>
```

Table 10. Call Graph Generation Analysis Tags

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 further information on granularity level, "About Siebel ARM Parameters and Variables" on page 192
RootID	The SarmID of the root Siebel ARM node.
ParentSARMID	The parent SarmNode 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 on Siebel architecture areas, see "About Siebel Application Response Measurement" on page 191.
AreaDescription	Name of the instrumentation area within the Siebel architecture. For information on Siebel architecture areas, see "About Siebel Application Response Measurement" on page 191.
SubAreaCodeSymbol	Symbol of the subarea within an area of the Siebel architecture. For information on Siebel architecture areas, see "About Siebel Application Response Measurement" on page 191.
SubAreaDescription	Name of the subarea within an area of the Siebel architecture. For information on Siebel architecture areas, see "About Siebel Application Response Measurement" on page 191.
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 timestamp.
Duration	Total time to execute the instrumented area.

Table 10. Call Graph Generation Analysis Tags

Tag	Description
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.
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

Running a user session trace analysis using Siebel ARM files from the Web server and 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 further details on the Siebel ARM architecture, see "About Siebel Application Response Measurement" on page 191.

For details on creating this format of Siebel ARM output, see "Running User Session Trace" on page 203.

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 details on those tags, see "About Performance Aggregation Analysis and Data" on page 205.

```
<UserID>
   <Session>
      <SessionID>
      <UserActionID>
         <ID>
         <SWERequest>
            <ReqID>
            <TotalServerTime>
            <WebServerTime>
            <NetworkTime>
            <SiebServerTime>
            <DatabaseTime>
            <DatabaseCalls>
            +<SiebsrvrDetail>
```

For descriptions on each of the tags specific to the user session trace analysis, see Table 11. For descriptions of the tags that are also a part of the performance aggregation analysis, see Table 9 on page 207.

Table 11. User Session Trace Tag Descriptions

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 Web Server Extension (SWSE) plug-in request.

Table 11. User Session Trace Tag Descriptions

Tag	Description
TotalServerTime	Total request time on the servers (includes Web server, Siebel Server, and network time).
WebServerTime	Total time spent on the Web server for a given request.
NetworkTime	Total time spent between the Web server and the Siebel Server. This time may also include some Siebel infrastructure time routing the request to the handling Siebel Server task.
SiebServerTime	Time spent on the Siebel Server.
DatabaseTime	The time spent on the network when communicating to the database.
DatabaseCalls	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 further information on these tags, see "About Performance Aggregation Analysis and Data" on page 205 and "About Call Graph Generation Analysis and Data" on page 213.

About Siebel ARM to CSV Conversion Data

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 details on creating this format of Oracle's Siebel ARM output, see "Running Siebel ARM Data CSV Conversion" on page 204.

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" on page 213. Information can be reviewed and organized by these columns. See Figure 4 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

Figure 4. Example of CSV Data

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 sections provide more information about using the Siebel ARM Query Tool:

About 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. It differs from the Siebel ARM Analyzer Tool in the following ways:

- Usability

  The Siebel ARM Query offers more command line options that allow you to create more complex queries.

- Performance

  The Siebel ARM Query tool processes data faster than the Siebel ARM Analyzer tool.

- Memory management

  You can increase the amount of memory that the Siebel ARM Query Tool uses in order to increase the statistical accuracy of results. By default, the Siebel ARM Query Tool uses approximately 20 MB of memory. This corresponds to a statistical error of approximately 1%. If you increase the amount of memory that the Siebel ARM Query Tool uses to 500 MB, you can reduce the statistical error to 0.2%.

  The amount of memory that you allocate does not affect the performance of the Siebel ARM Query Tool; any additional memory that you allocate is applied to the buffer that the Siebel ARM Query Tool uses for statistical calculation.

  NOTE: It is impossible to completely eliminate statistical error.

  For information on the parameters to modify the amount of memory that the Siebel ARM Query Tool uses for statistical accuracy, see "Configuring the Siebel ARM Query Tool" on page 220.

- 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" on page 221.

- Data filtering capability

  You can specify filters that include or exclude data from analysis by the Siebel ARM Query Tool.

- Aggregating data

  You can specify the order of aggregation. This includes rollup calculations. For more information about aggregation, see "Aggregating Siebel ARM Data with the Siebel ARM Query Tool" on page 235.

- 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 on how to specify the output format, see "Configuring Output from the Siebel ARM Query Tool" on page 222.

- Macro language

  The Siebel ARM Query Tool supports the use of macros. For more information, see "Using Macros with the Siebel ARM Query Tool" on page 238.

General Commands for the Siebel ARM Query Tool

This section describes the general options for use with the Siebel ARM Query Tool. These include commands to display online help and to display progress information to the command window. Table 12 describes these commands.

Table 12. General Flags for the Siebel ARM Query Tool

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.
-planonly	Prints an execution plan for a query without executing the query.
-quiet	Use this flag to print only the results to the output console. No progress information appears when you specify this flag. You can specify that progress information be saved to a file as in the following example: `> sarmquery –output verbose=verbose.txt –input data.sarm -aggregate time=1` where progress information is saved to the file verbose.txt.

Configuring the Siebel ARM Query Tool

This section describes how you can configure the Siebel ARM Query Tool. An examples of how you can configure the Siebel ARM Query Tool is modifying the amount of memory that the Siebel ARM query uses to improve statistical accuracy. Table 13 describes the flags that you can use to configure the Siebel ARM Query Tool. All the options described Table 13 are preceded by the option -config.

Table 13. Configuration Options for the Siebel ARM Query Tool

Flag	Description
file=<macro_file_name>	Specifies the file that contains a macro.
macro=macro_name(str)	Executes the macro and passes the string *str* as the first argument to the macro. Before you can use this flag, you must specify the file that contains the macro using the 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, the Siebel ARM Query Tool uses local time.

Table 13. Configuration Options for the Siebel ARM Query Tool

Flag	Description
`gmt=[+-]HHMM`	Parse and report 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 (00-59). For example to report Pacific Time (PST), use the following flag: `> sarmquery –config gmt=-0800`
`datalimit=<limit>`	Use this flag to specify 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>`	Use this flag to specify the maximum number of seconds that the Siebel ARM Query Tool can execute. When the maximum number of seconds elapses, the Siebel ARM Query 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.
`memlimit=<MB>`	Use this flag to specify the amount of memory that the Siebel ARM Query Tool can allocate to an internal buffer which the Siebel ARM Query Tool uses to compute aggregated statistics. The more memory that the internal buffer can use, the more accurate the statistics returned. The default value for this flag is twenty megabytes (MB). You can allocate a value between five MB and five hundred 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 section 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. Table 14 describes the available input options.

All the options described Table 14 are preceded by the option -input.

Table 14. Input Flags for the Siebel ARM Query Tool

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 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:
	`> 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 section discusses the output options that are available to you when you use the Siebel ARM Query Tool. The Siebel ARM Query Tool can simultaneously write output to files in the .TXT, .XML and .CSV format. You can use the Siebel ARM Query Tool to write different types of information to different files as illustrated in the following examples while Table 15 describes the flags that you can use to configure output.

```
> sarmquery –i d:\sarmdata –sel subarea=infra_entry –sel resp=1000 –out header=hdr.csv
–out sarm=sarm.csv -aggregate area –out agg=agg.xml
```

The above 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
- Writes 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 Comma Separated Values (CSV) format to the file named *sarmdata.csv*. The optional value #20000 writes a maximum of 20000 lines to sarmdata.csv. Once sarmdata.csv contains 20000 lines, the Siebel ARM Query Tool writes to a new file with the name sarmdata_0002.csv and so on until all data is output.

NOTE: All flags in Table 15 are preceded by the command option `-output`.

Table 15. Output Flags for the Siebel ARM Query Tool

Flag	Description
`fdr=<filename>`	Converts all fdr files (specified using `-input`) to CSV format and write them to the specified file.
`fdrhdr=<filename>`	Converts all the fdr headers to CSV and write them to the specified file.
`error=<filename>`	By default, the Siebel ARM Query Tool writes output `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.

Table 15. Output Flags for the Siebel ARM Query Tool

Flag	Description

`header=<filename>`
`appheader=<filename>`
`webheader=<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.
	■ webheader
	Writes header information from Siebel ARM files generated by the Siebel Web Server.
	■ 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 –i dir –o header=all.csv –o appheader=app.csv –o webheader=web.csv –o cliheader=cli.csv`

Table 15. Output Flags for the Siebel ARM Query Tool

Flag	Description
`sarm=<filename> appsarm=<filename> websarm=<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. ■ websarm Writes all Siebel ARM data generated by the Siebel Web Server (SWSE) 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 and the SWSE to different files: `> sarmquery –output appsarm=app.csv –output websarm=web.csv`
`agg=<filename> iagg=<filename> sagg=<filename>`	The available flags are: ■ agg Write the aggregation report to the specified file. ■ sagg Write a subset of the aggregation report to the specified file. This flag specifies exclusive metrics. ■ iagg Write 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`.

Table 15. Output Flags for the Siebel ARM Query Tool

Flag	Description
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 write the percentages of server requests that complete within 100 milliseconds, between 100 milliseconds to 500 milliseconds and so on: `> sarmquery –hist resp=100,500,1000,2000,5000 – out pctcount=stdout.txt` ■ pctselftime Writes the percentage of time spent in internal parts of the Siebel Server. This is also know as SLA fingerprint.

Using Filters with the Siebel ARM Query Tool

This section describes how to use 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. This metadata describes the data that the body section contains. 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.

- Table 16 describes the filters that you can use for the metadata in Siebel ARM files
- Table 17 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:

  `> sarmquery –select user=JohnDoe`

  retrieves Siebel ARM records associated with the users ‘johndoe’, ‘JOHNDOE’, and ‘JoHnDoE’.

- String filters also accept leading and trailing wild cards. For example, each of the following queries retrieve records associated with JohnDoe, 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 the options described Table 16 are preceded by the option `-select`.

Table 16. Header Filters for the Siebel ARM Query Tool

Flag	Description
`component=<name>`	Selects headers from Siebel ARM files that were generated by the named component.
`fillfactor=<percent>` Or: `fillfactor=<minpct>,<maxpct>`	Use this flag to specify a percentage value; the Siebel ARM Query Tool retrieves headers that are <percent> full. 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=<name>`	Selects headers from Siebel ARM files generated on the named host.
`procid=<int>`	Selects headers from Siebel ARM files generated by a process whose id (pid) is <int>.
`segcapacity=<nrecs>` Or: `segcapacity=<min>,<max>`	Select headers whose capacity to hold Siebel ARM records matches the number of records specified by <nrecs>. <nrecs> is specified in number of records. 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>`	Select 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, a minimum number of seconds and a maximum number of seconds 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 to retrieve a range of values.

Table 16. Header Filters for the Siebel ARM Query Tool

Flag	Description
`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. Specify two arguments, <min> and <max> to retrieve a range.
`server=<name>`	Retrieve headers from Siebel ARM files generated on the specified Siebel <name>.
`sourcetype=<val>`	Retrieve headers from Siebel ARM files generated by the specified type of server or process. Specify one of the following parameters in place of <val> to retrieve the headers: ■ app This retrieves headers generated by Siebel Servers. ■ web This retrieves headers generated by the Siebel Web Server Extension (SWSE). ■ cli This 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. Specify two arguments to retrieve a range of values between the minimum threshold value and the maximum threshold value.

Table 16. Header Filters for the Siebel ARM Query Tool

Flag	Description

`starttime=<start>`	Specify a start time to retrieve headers that contain Siebel ARM records that ended after the specified start time. 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="2006-02-13 17:05:00"`
	■ A number interpreted in Universal Time Code (UTC)
	For example:
	`sarmquery -select starttime=1108083900`
	■ A negative number to indicate the number of seconds from current time:
	For example, this command retrieves headers that contain data generated less than 300 seconds ago:
	`> sarmquery -select starttime=-300`
	NOTE: Time filters are compared against the generation end time for Siebel ARM records.

Table 16. Header Filters for the Siebel ARM Query Tool

Flag	Description
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. ■ String in the format *YYYY-MM-DD hh:mm:ss* For example: > sarmquery -select endtime="2006-02-13 17:05:00" ■ A number interpreted in Universal Time Code (UTC) For example: sarmquery -select endtime=1108083900 ■ A negative number to indicate the number of seconds from 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 greater than 600 seconds ago: > sarmquery -select starttime=-600

Table 16 describes the filters that you can use to when you formulate a query to retrieve Siebel ARM data.

All the options described Table 16 are preceded by the option -select.

Table 17. Siebel ARM Query Tool Flags for Record Selection

Flag	Description
clickid=<ID>	The Web server Siebel ARM files (-select sourcetype=web) contain a user click ID (also known as operation ID). This filter retrieves all records where clickid equals <ID>.
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.

Table 17. Siebel ARM Query Tool Flags for Record Selection

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>. 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 Equal to SARMLevel=1. This indicates high level information. ■ <level> = 2 Equal to SARMLevel=2. This indicates. detailed performance information ■ <level> = 3 Equal to SARMLevel=3 This indicates debug or internal performance information.
parentid=<ID>	Select 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/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 blind search using <sessid>. Wild cards are acceptable.
taskid=<taskid>	Select all records that associated with task <taskid>.
threadid=<threadid>	Select all records that were created by the thread whose OS thread id is <threadid> .

Table 17. Siebel ARM Query Tool Flags for Record Selection

Flag	Description
user=<username>	Select all records that were created by user named <username>. <username> is typically the login name. search is case blind and wild cards are acceptable.

area=<area>
area=<areacode>
subarea=<sub>
subarea=<subcode>
pararea=<parea>
pararea=<pareacode>
parsubarea=<psarea>
parsubarea=<pscode>
	Two filters, area and subarea, identify each Siebel ARM record.
	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 enterprise database. Siebel ARM records associated with DBC indicate the performance of database queries. For example, the following command:
	`> sarmquery –select area=DBC`
	retrieves all Siebel ARM records associated with database queries.
	You can retrieve a complete list of areas, subareas and descriptions from the Siebel ARM Query Tool. 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, you can use them directly in this command.
	Otherwise, you can use the string form of the symbol. When using the string form, you don't need to use the entire text – you may use a partial text provided 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:
	`> sarmquery –select pararea=swe`
	retrieves all Siebel ARM records whose parent area is 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.

Table 17. Siebel ARM Query Tool Flags for Record Selection

Flag	Description

`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 integer 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
	This filter 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.
	■ detail
	This filter retrieves Siebel ARM records where the detail metadata equals <string>. Detail metadata typically contains identification information. For example, it may contain a database row ID or the name of a business component method.
	■ int1
	This filter 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 depends 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:
	`> sarmquery –select memusage=1000000`
	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.

Table 17. Siebel ARM Query Tool Flags for Record Selection

Flag	Description

`pctcpu=<pct>`
Or:
`pctcpu=<min>,<max>`
	Specify a single argument to retrieve Siebel ARM records that indicate a CPU consumption of <pct> 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 etc.

`starttime=<start>`
Or:
`endtime=<end>`
	The semantics and syntax are similar to the time filters described for headers, except the filters select Siebel ARM records based on the timestamps.

Table 17. Siebel ARM Query Tool Flags for Record Selection

Flag	Description

`tree=all`
`tree=descendant`
`tree=ancestor`
`tree=parent`
	These are a specialized set of selection filters that don't 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.
	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:
	■ -select tree=all
	Replaces a selected record by all records in its tree.
	■ -select tree=parent
	Replaces a selected record by its parent. The record itself is discarded.
	■ -select 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.
	■ -select tree=children
	Replaces a selected record by its immediate children. The selected record is not included.

Aggregating Siebel ARM Data with the Siebel ARM Query Tool

This section 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 computing statistics on the group.

Multiple `–aggregate` options can be specified, which the Siebel ARM Query Tool interprets 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 retrieved by the selection filters.

All the options described Table 18 are preceded by the option -aggregate.

Table 18. Flag Options for Aggregation

Flag	Description
`area`	Siebel ARM records that have the same value for area are grouped together.

`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/subarea. As a result, aggregate instance should either be preceded by an aggregation of area/subarea or a filter on area/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. **NOTE:** User name is case sensitive.

Table 18. Flag Options for Aggregation

Flag	Description

`sessionid`	Aggregates Siebel ARM records that have the same value for session ID.
	
	TIP: This flag can 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 may have multiple requests. Sometimes a single user action on the client (browser) may result 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>`	Aggregate Siebel ARM records by their timestamps over the interval specified by <interval> where <interval> is a value in minutes. For example, if you specify 5, then Siebel ARM records with timestamps of 12:00 to 12:05 form one aggregation, those with timestamps 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 application server over intervals of fifteen minutes: `> sarmquery –select source=app –select subarea=infra_entry -aggregate time=15`

Generating Histograms with 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 may return too many values if you do not use the histogram.

For example, the following query:

```
> sarmquery -aggregate resptime
```

might retrieve millions of rows as the first row returns the requests that completed in one millisecond, the second row the number of requests that completed in two milliseconds.

A more effective query generates a histogram as in the following example:

```
> sarmquery –histogram resptime=100,200,300,400,500
```

which 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 above example, the values 100, 200, 300 and so on are known as *bin endpoints*.

Table 19 describes the flag options that you can use with the histogram parameter. All options described in Table 19 are preceded by the option `-histogram`.

Table 19. Flag Options for the Histogram Parameter

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>…<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 width of the execution trees.
`depth=<val1>,<val2>…<valN>`	Creates a histogram of the depth of the execution trees.

Using Macros with the Siebel ARM Query Tool

The Siebel ARM Query Tool supports the use of macros. A macro allows you to save and reuse 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 should 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 that 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=web -select subarea=swse_login -select tree=all -select depth=1 -aggregate user -output avginclresp=stdout.txt
```

However, in the .CFG file the above 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 may 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="2004-06-10 10:00:00" –sel endtime="2004-06-11 09:59:59"
```

For a complete description of the macro language syntax and other options, execute the following command from the command line:

```
> sarmquery –macrosyntax
```

Index

D

E

K

L

M

N

O

P

T

U

V

W