SYBASE®

Monitor Client Library 程序员指南

Adaptive Server® Enterprise

15.5

文档 ID：DC33434-01-1550-01

最后修订日期：2009 年 10 月

Sybase, Inc., One Sybase Drive, Dublin, CA 94568.

目录

关于本手册

读者

本指南面向使用 Adaptive Server® Enterprise Monitor Server 或 Adaptive Server Enterprise Monitor Historical Server 的编程人员。

如何使用本手册

编写 Monitor Client Library 应用程序时，请参考此书有关构建 Monitor Client Library 程序的常见信息。

- 第 1 章 “Monitor Client Library 快速入门”解释了如何设计基本 Monitor Client Library 程序的结构，并包含了一个简单、完整的 Monitor Client Library 应用程序。
- 第 2 章 “数据项和统计类型”描述了 Monitor Client Library 应用程序中用于收集性能数据的数据项、统计类型及数据项有效组合。
- 第 3 章 “Monitor Client Library 函数”描述了各个函数，包括语法、参数值、示例、权限及相关函数。
- 第 4 章 “构建 Monitor Client Library 应用程序”描述了编译、链接 Monitor Client Library 程序的方法。
- 第 5 章 “Monitor Client Library 配置说明”阐述了在 UNIX 或 Windows 上配置 Monitor Client Library 的方法。
- 附录 A “视图示例”提供了有效视图的示例。
- 附录 B “数据类型和结构”概述了 Monitor Client Library 使用的数据类型，并描述了在 C 或 Open Client™ Client Library 中没有等效项的数据类型。
- 附录 C “向后兼容性”列出了旧函数和相应的替代函数。
- 附录 D “疑难解答信息和错误消息”解释了怎样响应使用 Monitor Client Library 时可能会碰到的问题，并列出了可能报告的错误消息。

相关文档

Adaptive Server® Enterprise 文档集包括：

- 针对所用平台的发行公告 — 包含未能及时写入手册的最新信息。

  最新版本的发行公告可能已经推出。要了解本产品 CD 发行以后增加的重要产品或文档信息，请使用 Sybase Product Manuals 网站。

- 针对所用平台的安装指南 — 介绍所有 Adaptive Server 产品及相关 Sybase 产品的安装、升级和某些配置过程。
- New Feature Summary （《新增功能摘要》）— 介绍 Adaptive Server 中的新增功能，为支持这些功能所做的系统更改，以及可能会影响现有应用程序的更改。
- Active Messaging Users Guide （《Active Messaging 用户指南》）— 介绍如何使用 Active Messaging 功能捕获 Adaptive Server Enterprise 数据库中的事务 （数据更改），并将它们作为事件实时发送给外部应用程序。
- 《组件集成服务用户指南》— 说明如何使用组件集成服务功能来连接远程 Sybase 和非 Sybase 数据库。
- 针对所用平台的 《配置指南》 — 提供执行特定配置任务的操作说明。
- 《词汇表》 — 定义 Adaptive Server 文档中使用的技术术语。
- 《Historical Server 用户指南》 — 介绍如何使用 Historical Server 从 Adaptive Server 获取性能信息。
- 《Adaptive Server Enterprise 中的 Java》 — 介绍在 Adaptive Server 数据库中如何安装 Java 类，如何将它们用作数据类型、函数及存储过程。
- 《Job Scheduler 用户指南》 — 提供有关如何使用命令行或图形用户界面 (GUI) 在本地或远程 Adaptive Server 上进行安装和配置以及创建和调度作业的操作说明。
- 《迁移技术指南》 — 介绍迁移到其它 Adaptive Server 版本所需的策略和工具。
- 《Monitor Client Library 程序员指南》— 介绍如何编写访问 Adaptive Server 性能数据的 Monitor Client Library 应用程序。
- 《Monitor Server 用户指南》 — 介绍如何使用 Monitor Server 从 Adaptive Server 获取性能统计信息。
- Monitoring Tables Diagram （《监控表框图》） — 以张贴画的形式阐明监控表及其实体关系。大图只提供印刷品；采用 PDF 格式时提供缩略图。

- Performance and Tuning Series （《性能和调优系列》） — 是一系列丛书，介绍如何调节 Adaptive Server 以获得最佳性能：
  - Basics （《基础知识》） — 包含通晓和研究 Adaptive Server 中的性能问题需具备的基础知识。
  - Improving Performance with Statistical Analysis （《利用统计分析改进性能》） — 介绍 Adaptive Server 如何存储和显示统计信息，及如何使用 set statistics 命令分析服务器统计信息。
  - Locking and Concurrency Control （《锁定和并发控制》） — 介绍如何使用锁定方案来改进性能，以及如何选择索引以最大限度地减少并发。
  - Monitoring Adaptive Server with sp_sysmon （《使用 sp_sysmon 监控 Adaptive Server》） — 讨论如何使用 sp_sysmon 监控性能。
  - Monitoring Tables （《监控表》） — 介绍如何从 Adaptive Server 的监控表中查询统计信息和诊断信息。
  - Physical Database Tuning （《物理数据库调优》） — 介绍如何管理物理数据放置、为数据分配的空间及临时数据库。
  - Query Processing and Abstract Plans （《查询处理和抽象计划》） — 介绍优化程序如何处理查询以及如何使用抽象计划更改某些优化程序计划。
- 《快速参考指南》 — 这是一本袖珍手册，完整地列出了各种命令、函数、系统过程、扩展系统过程、数据类型和实用程序的名称和语法 （该手册在用 PDF 格式阅读时采用正常大小）。
- 《参考手册》 — 是一系列丛书，包含详细的 Transact-SQL® 信息：
  - 《构件块》 — 讨论数据类型、函数、全局变量、表达式、标识符、通配符和保留字。
  - 《命令》 — 提供了命令的文档资料。
  - 《过程》 — 介绍系统过程、目录存储过程、系统扩展存储过程及 dbcc 存储过程。
  - 《表》 — 讨论系统表、监控表及 dbcc 表。

- 《系统管理指南》 —
  - 《卷 1》 — 介绍系统管理的基础知识，包括对配置参数、资源问题、字符集、排序顺序的说明以及有关诊断系统问题的操作说明。《卷 1》的第二部分深入讨论了安全性管理。
  - 《卷 2》 — 包括管理物理资源、镜像设备、配置内存和数据高速缓存、管理多处理器服务器和用户数据库、装入和卸下数据库、创建和使用段、使用 reorg 命令及检查数据库一致性的操作说明和指导。《卷 2》的后半部分介绍了如何备份和恢复系统数据库和用户数据库。
- System Tables Diagram （《系统表框图》） — 以张贴画的形式阐明系统表及其实体关系。大图只提供印刷品；采用 PDF 格式时提供缩略图。
- 《Transact-SQL 用户指南》— 提供有关 Transact-SQL 这一 Sybase 关系数据库语言增强版的文档资料。此指南可用作数据库管理系统初级用户的教科书，其中还包含有关 pubs2 和 pubs3 示例数据库的详细说明。
- Troubleshooting Series （《故障排除系列》） —
  - Troubleshooting: Error Messages Advanced Resolutions （《故障排除：错误消息高级解析》） — 包含针对您可能遇到的问题的故障排除过程。此处讨论的问题是 Sybase 技术支持部门的员工最常听到的问题。
  - Troubleshooting and Error Messages Guide （《故障排除和错误消息指南》） — 包含有关如何解除最常出现的 Adaptive Server 错误消息的详细说明。
- 《加密列用户指南》 — 介绍了如何利用 Adaptive Server 配置和使用加密列。
- 《内存数据库用户指南》 — 介绍如何配置和使用内存数据库。
- Using Adaptive Server Distributed Transaction Management Features （《使用 Adaptive Server 分布式事务管理功能》） — 介绍如何在分布式事务处理环境中配置、使用 Adaptive Server DTM 功能以及如何排除其中的故障。
- 《将 Backup Server 与 IBM® Tivoli® Storage Manager 配合使用》 — 说明如何设置和使用 IBM Tivoli Storage Manager 以创建 Adaptive Server 备份。

- 《在高可用性系统中使用 Sybase 故障切换》 — 提供有关使用 Sybase 的故障切换功能将 Adaptive Server 配置为高可用性系统中的协同服务器的操作说明。
- Unified Agent and Agent Management Console （《Unified Agent 和 Agent Management Console》） — 介绍用于提供管理、监控和控制分布式 Sybase 资源的运行时服务的 Unified Agent。
- 《实用程序指南》 — 提供有关在操作系统级别执行的 Adaptive Server 实用程序 （如 isql 和 bcp）的文档资料。
- 《Web 服务用户指南》 — 介绍如何配置、使用 Adaptive Server Web 服务以及如何排除其中的故障。
- 《适用于 CICS、 Encina 和 TUXEDO 的 XA 接口集成指南》 — 提供有关将 Sybase DTM XA 接口与 X/Open XA 事务管理器配合使用的说明。
- 《Adaptive Server Enterprise 中的 XML 服务》— 介绍了 Sybase 本机 XML 处理器、 Sybase 基于 Java 的 XML 支持及数据库中的 XML，并提供了 XML 服务中可用的查询和映射函数的文档资料。

其它信息来源

使用 Sybase Getting Started CD、 SyBooks™ CD 和 Sybase® Product Manuals 网站可以了解有关产品的更多信息：

- Getting Started CD 包含 PDF 格式的发行公告和安装指南，还可能包含 SyBooks CD 中未收纳的其它文档或更新信息。它随软件一起提供。若要阅读或打印 Getting Started CD 上的文档，需要使用 Adobe Acrobat Reader，该软件可以使用 CD 上提供的链接从 Adobe 网站免费下载。
- SyBooks CD 含有产品手册，它随软件一起提供。基于 Eclipse 的 SyBooks 浏览器使您能够以易于使用的、基于 HTML 的格式阅读手册。

  有些文档可能是以 PDF 格式提供的，您可以通过 SyBooks CD 上的 PDF 目录访问这些文档。若要阅读或打印 PDF 文件，您需要使用 Adobe Acrobat Reader。

  有关安装和启动 SyBooks 的说明，请参见 Getting Started CD 上的《SyBooks 安装指南》或 SyBooks CD 上的 *README.txt* 文件。
- Sybase Product Manuals 网站是 SyBooks CD 的联机版本，您可以使用一种标准 Web 浏览器来访问它。除了产品手册之外，还可以找到有关 EBFs/Maintenance （EBF/ 维护）、 Technical Documents （技术文档）、 Case Management （案例管理）、 Solved Cases （解决的案例）、 Newsgroups（新闻组）和 Sybase Developer Network（Sybase 开发员网络）的链接。

若要访问 Sybase Product Manuals 网站，请转到位于 http://www.sybase.com/support/manuals/ 的 “产品手册” (Product Manuals)。

Web 上的 Sybase 认证

Sybase 网站上的技术文档不断在更新。

❖ **查找有关产品认证的最新信息**

1 将 Web 浏览器定位到位于 http://www.sybase.com/support/techdocs/ 的 “技术文档” (Technical Documents)。

2 单击 “认证报告” (Certification Report)。

3 在 “认证报告” (Certification Report) 过滤器中选择相应的产品、平台和时间范围，然后单击 “查找” (Go)。

4 单击 “认证报告” (Certification Report) 标题显示此报告。

❖ **查找有关组件认证的最新信息**

1 将 Web 浏览器定位到位于 http://certification.sybase.com/ 的 “可用性和认证报告” (Availability and Certification Reports)。

2 在 “按基本产品搜索” (Search by Base Product) 下选择产品系列和产品，或在 “按平台搜索” (Search by Platform) 下选择平台和产品。

3 选择 “搜索” (Search) 以显示所选项目的可用性和认证报告。

❖ **创建个性化的 Sybase 网站 （包括支持页）视图**

建立 MySybase 配置文件。MySybase 是一项免费服务，用于创建个性化的 Sybase 网页视图。

1 将 Web 浏览器定位到位于 http://www.sybase.com/support/techdocs/ 的 “技术文档” (Technical Documents)。

2 单击 “我的 Sybase” (MySybase) 并创建 MySybase 配置文件。

Sybase EBF 和软件维护

❖ **查找有关 EBF 和软件维护的最新信息**

1. 将 Web 浏览器定位到位于 http://www.sybase.com/support 的“Sybase 支持页”(Sybase Support Page)。
2. 选择“EBF/ 维护”(EBFs/Maintenance)。如果出现相应提示，请输入您的 MySybase 用户名和口令。
3. 选择一个产品。
4. 指定时间范围并单击“查找”(Go)。即会显示 EBF/ 维护版本的列表。

   锁形图标表示因为您没有注册为“技术支持联系人”(Technical Support Contact)，因此您没有某些 EBF/ 维护版本的下载授权。如果您尚未注册，但拥有 Sybase 代表提供的或通过支持合同获得的有效信息，请单击“编辑角色”(Edit Roles) 将“技术支持联系人”(Technical Support Contact) 角色添加到 MySybase 配置文件中。
5. 单击信息图标可显示 EBF/ 维护报告，单击产品说明可下载软件。

约定

以下各部分将说明在本手册中使用的约定。

SQL 是一种形式自由的语言。没有规定每一行中的单词数量或者必须折行的地方。然而，为便于阅读，本手册中所有示例和大多数语法语句都经过了格式设置，以便语句的每个子句都在一个新行上开始。有多个成分的子句会扩展到其它行，这些行会有缩进。复杂命令使用已修改的 Backus Naur Form (BNF) 表示法进行了格式处理。

表 1 说明本手册中出现的语法语句的约定：

表 1：本手册的字体和语法约定

元素	示例
命令名、过程名、实用程序名和其它关键字用 sans serif 字体显示。	select sp_configure
数据库名和数据库类型用 sans serif 字体显示。	master 数据库
书名采用正常字体并加书名号；文件名、变量和路径名用斜体显示。	《系统管理指南》 *sql.ini* 文件 *column_name* *$SYBASE/ASE* 目录

变量（即代表您要填充的值的词语）作为查询或语句的一部分出现时用斜体的 Courier 字体显示。	`select column_name`
	`from table_name`
	`where search_conditions`

将小括号作为命令的一部分键入。	compute *row_aggregate* (*column_name*)

元素	示例
双冒号加等号表示语法是用 BNF 表示法编写的。请勿输入此符号。表示 “被定义为”。	`::=`
大括号表示必须至少选择括号中的一个选项。不要输入大括号。	`{cash, check, credit}`
中括号表示可以选择括号中的一个或多个选项，也可不选。不要输入中括号。	`[cash | check | credit]`
逗号表示可以选择任意多个显示的选项。可用逗号作为命令的一部分来分隔选项。	`cash, check, credit`
竖线 (|) 表示只可选择所显示的选项中的一个。	`cash | check | credit`
省略号 (...) 表示可以将最后一个单元*重复*任意次。	`buy thing = price [cash | check | credit] [, thing = price [cash | check | credit]]...` 您必须至少购买一件产品，并给出其价格。可以选择一种付款方式：中括号中的选项之一。还可选择购买其它物品：可根据需要购买任意数量的物品。对于要买的每种产品，给出其名称、价格和付款方式 （可选）。

- 语法语句 （显示命令的语法和所有选项）显示如下：

  sp_dropdevice [*device_name*]

  对于具有多个选项的命令：

  select *column_name*
  from *table_name*
  where *search_conditions*

  在语法语句中，关键字 （命令）采用常规字体，而标识符为小写。斜体表示用户提供的内容。

- 说明 Transact-SQL 命令用法的示例显示如下：

  ```
  select * from publishers
  ```

- 计算机输出的示例显示如下：

```
pub_id    pub_name                city          state
------    --------------------    -----------   -----
0736      New Age Books           Boston        MA
0877      Binnet & Hardley        Washington    DC
1389      Algodata Infosystems    Berkeley      CA

(3 rows affected)
```

本手册中的大多数示例都用小写显示。不过，输入 Transact-SQL 关键字时可以忽略大小写。例如， SELECT、 Select 和 select 之间没有区别。

Adaptive Server 是否区分数据库对象 （如表名）的大小写，取决于安装在 Adaptive Server 上的排序顺序。通过重新配置 Adaptive Server 的排序顺序，可改变单字节字符集的区分大小写设置。有关详细信息，请参见《系统管理指南》。

辅助功能特性

本文档具有为提供辅助功能而进行了专门设计的 HTML 版本。可以利用适应性技术 （如屏幕阅读器）浏览 HTML 文档，也可以用屏幕放大器进行查看。

Adaptive Server HTML 文档已经过测试，符合美国政府 “第 508 节辅助功能”的要求。符合 “第 508 节”的文档一般也符合美国之外的辅助功能原则，如 World Wide Web 协会 (W3C) 针对网站的原则。

注释 可能需要配置辅助功能工具，以便获得最佳使用效果。某些屏幕阅读器按照大小写来辨别文本，例如将 “ALL UPPERCASE TEXT”看作首字母缩写，而将 “MixedCase Text”看作单词。对工具进行配置，规定语法约定，您可能会感觉更方便。有关工具的信息，请查阅文档。

有关 Sybase 如何支持辅助功能的信息，请参见位于 http://www.sybase.com/accessibility 的 “Sybase 辅助功能” (Sybase Accessibility)。“Sybase 辅助功能” (Sybase Accessibility) 站点包括指向 “第 508 节”和 W3C 标准相关信息的链接。

如果需要帮助

对于购买了支持合同的客户安装的每一个 Sybase 产品，都会有一位或多位指定人员获得与 Sybase 技术支持部门联系的授权。如果使用手册或联机帮助不能解决问题，可让指定人员与 Sybase 技术支持部门联系或与所在区域的 Sybase 子公司联系。

第 1 章

Monitor Client Library 快速入门

本章包含有关 Monitor Client Library 快速入门的信息。

概述

Monitor Client Library 是 Adaptive Server Enterprise Monitor 的一部分。它是一种应用程序编程接口 (API)，可用于编写连接到 Adaptive Server、 Adaptive Server Enterprise Monitor Server （简称 Monitor Server）和 Adaptive Server Enterprise Historical Server（简称 Historical Server）的客户端应用程序以收集性能数据。本章描述了 Adaptive Server Enterprise Monitor，解释了 Monitor Client Library 应用程序组件，并列出该应用程序的一个例子。

什么是 Adaptive Server Enterprise Monitor?

Adaptive Server Enterprise Monitor 提供了一种方法，可以在实时模式或历史数据收集模式下监控 Adaptive Server 性能。系统管理员可使用此信息找出潜在的资源瓶颈、研究当前问题以及调优以获得更好的性能。 Adaptive Server Enterprise Monitor 提供反馈用于几个层次的性能调优：

- Adaptive Server 配置
- 表和索引设计
- 应用程序和存储过程中的 SQL 语句

Adaptive Server Enterprise Monitor 的组件

Adaptive Server Enterprise Monitor 由以下四个收集或显示 Adaptive Server 性能数据的组件组成：

- Monitor Server — 是实时收集 Adaptive Server 性能数据并向其它 Adaptive Server Enterprise Monitor 组件提供这些数据的服务器。 Monitor Server 是一种 Sybase Open Server™ 应用程序。
- Historical Server — 从 Monitor Server 获得 Adaptive Server 性能数据并将这些数据保存在文件中供以后分析的一种服务器。 Historical Server 是一种 Sybase Open Server 应用程序。
- 用于 Sybase Central 的 Adaptive Server 插件中的监控器 (Monitor Viewer) — 它为 Monitor Server 提供了图形用户界面。这些监控器从 Monitor Server 获得 Adaptive Server 性能数据，并以表和图形的形式实时显示数据。
- Monitor Client Library — Monitor Server 的一种应用程序编程接口，可供用户用来开发监控应用程序。 Monitor Viewer 和 Historical Server 是 Monitor Client Library 应用程序。

Adaptive Server Enterprise Monitor 体系结构

Adaptive Server 将性能数据保存在 Monitor Server 可以读取的共享*内存区*中。因为 Monitor Server 采用了这种共享内存技术，所以必须在安装受监控的 Adaptive Server 的同一台计算机上安装和运行它。 Adaptive Server 与 Monitor Server 之间存在一一对应的关系。有关 Monitor Server 的详细信息，请参见 《Sybase Adaptive Server Enterprise Monitor Server 用户指南》。

Monitor Client Library 应用程序从 Monitor Server 获取 Adaptive Server 性能统计信息。这些应用程序是 Monitor Server 的客户端。出于性能方面的考虑， Sybase 建议在运行 Adaptive Server 及其对应的 Monitor Server 以外的计算机上运行 Monitor Client Library 应用程序。

Sybase Central 中的 Monitor Viewer 包括一系列监控器，以不同的详尽程度显示 Adaptive Server 资源使用情况的各个方面。每个打开的监控器都是一个独立的应用程序，与 Monitor Server 都有唯一的客户端连接。 Sybase Central 中，每个 Adaptive Server 安装都有它自己的 Monitors 文件夹，里面包含一系列监控器对象。

Historical Server 从 Monitor Server 收集性能信息，并将数据保存到文件以进行随后的分析。Historical Server 接口让用户指定要收集的数据和期望的时间段。它们也包括历史数据回放功能。这些接口是：

- isql 中的命令接口。请参见《Sybase Adaptive Server Enterprise Monitor Historical Server 用户指南》。
- 使用 Monitor Client Library 的编程接口。请参见第 3 章 “Monitor Client Library 函数”和 《Sybase Adaptive Server Enterprise Monitor Historical Server 用户指南》。

编写基础的 Monitor Client Library 程序

基础的 Monitor Client Library 应用程序：

1　定义错误处理。

2　使用下列步骤连接到服务器：

- 分配连接。
- 设置连接属性。
- 连接到服务器。

3　创建一个或多个视图以定义要监控的性能数据。

4　用过滤器将特定的性能数据值定为目标 （可选）。

5　设置性能数据值的报警 （可选）。

6　请求性能数据值。

7　处理结果。

8　关闭与服务器的连接。

9　释放连接或通过重新连接再次使用此连接。

注释　必须拥有 Adaptive Server 的系统管理员角色，或拥有对存储过程 mon_rpc_connect 的执行权限，才能执行监控。

应用程序逻辑流

大多数 Monitor Client Library 应用程序的逻辑流都与下面的相似：

```
分配一个连接
    设置该连接的属性
    连接
    执行循环以针对该连接创建视图
    执行循环以创建过滤器 （可选）
    执行循环以创建报警 （可选）
    执行循环以刷新连接
        针对每个视图
            获取行计数
            针对每一行
                针对每一列
                    获取数据
                    显示数据
    执行循环以删除报警 （可选）
    执行循环以删除过滤器 （可选）
    执行循环以删除视图 （可选）
    关闭监控器连接
    释放或重用连接
```

其中：

- 一个应用程序可有任意多个连接。
- 一个连接可有一个或多个视图。
- 一个视图必须有一个或多个数据项。
- 视图中每个数据项可有一个过滤器。
- 一个视图可有任意多个报警，视图中的每个数据项可有多个报警。

下面几节描述了创建基础 Monitor Client Library 程序的步骤。这些步骤交叉引用了它们后面的示例程序。

第 1 步：定义错误处理

应用程序使用一个或多个回调例程，以处理 Monitor Client Library 和 Server 错误及信息性消息。

第 2 步：连接到服务器

Monitor Client Library 函数需要建立 Adaptive Server Enterprise Monitor 连接。 Adaptive Server Enterprise Monitor 连接使用一个或多个 Open Client 连接，具体取决于连接类型。

Monitor 的两种连接类型分别是*活动*模式和*历史*模式：

- 活动模式连接到 Monitor Server 和 Adaptive Server。它可以直接访问性能数据。
- 历史模式连接到 Historical Server，记录性能数据用于以后使用，或回放记录的数据。

连接到服务器是一个分为三步的过程。应用程序：

- 分配连接结构
- 如有必要，设置连接属性
- 登录到服务器

分配连接结构

应用程序调用 smc_connect_alloc 以分配连接结构。

设置连接结构属性

应用程序调用 smc_connect_props 来设置、检索或清除连接结构属性。

连接属性定义了连接行为的各个方面。例如：

- SMC_PROP_USERNAME 定义了在登录到服务器时连接将使用的 *username*。
- SMC_PROP_PASSWORD 指定该 *username* 的 *password*。
- SMC_PROP_SERVERNAME 定义了此次连接的服务器。

- SMC_PROP_IFILE 定义了用于此连接的 *interfaces* 文件名。如果在 UNIX 系统中不指定这个属性，将使用 SYBASE 环境变量目录下的缺省 *interfaces* 文件。在 Windows NT 中，缺省的 *interfaces* 文件是 *sql.ini*。
- SMC_PROP_SERVERMODE 定义了连接类型：活动连接或历史连接。

必需的连接属性

应用程序必须至少设置指定连接的 *username* 的连接属性 (SMC_PROP_USERNAME)，以使服务器可以通过要求提供有效口令来鉴定用户的身份。如果服务器要求提供口令，则应用程序必须将 SMC_PROP_PASSWORD 属性的值设置为用户的服务器口令。

连接到服务器

应用程序调用 smc_connect_ex 以连接到服务器。建立连接时，smc_connect_ex 与网络建立通信，登录到服务器，并将所有连接特定的属性信息传递给服务器。在与 Adaptive Server 建立连接后，就会将 dbcc traceon 消息写入 Adaptive Server 的错误日志。可以忽略这些错误消息。

例如，如果服务器支持基于网络的用户鉴定，且客户端应用程序请求服务器进行这种鉴定，则 Client Library 和该服务器会查询网络的安全系统，以查看相应用户 （用户名由 SMC_PROP_USERNAME 指定）是否已登录到网络。

第 3 步：创建视图

视图 由数据项组定义。指定的数据项确定数据的汇总方式。由于可指定多个视图，所以应用程序在收集数据上具有极大的灵活性。例如，一个由两个数据项 （设备名、采样值和设备 I/O、采样率）组成的视图，返回每个数据库设备的设备 I/O 率。

有关数据项有效组合的详细信息以及如何汇总数据项的信息，请参见第 2 章 “数据项和统计类型”。

有关视图的示例，请参见附录 A “视图示例”。

数据项

数据项是可以从 Monitor Client Library 上获得的特定数据段，例如，页 I/O、登录名、设备读取数等等。必须指定视图中每个数据项的统计类型。

统计类型

统计类型 定义了数据项的持续时间 （采样或会话）以及服务器是否对数据项执行计算。

六种统计类型为：

- SMC_STAT_VALUE_SAMPLE — 此统计类型返回适用于最近一次采样间隔的活动计数或某种信息。不执行计算。
  - 活动计数 — 对于表示活动计数的数据项，SMC_STAT_VALUE_SAMPLE 返回在最近一次采样间隔期间某项活动发生的次数。例如， SMC_NAME_PAGE_IO 的 SMC_STAT_VALUE_SAMPLE 表示在最近一次采样间隔期间发生的页 I/O 数。
  - 其它信息 — 对于表示字符串的数据项，它是唯一有效的统计类型。例如， SMC_NAME_OBJECT_NAME 的 SMC_STAT_VALUE_SAMPLE 返回数据库对象的名称。如果数据项表示诸如 ID 和配置参数值等绝不会作为计算对象的值，则此统计类型也是唯一对此类数据项有效的数据类型。
- SMC_STAT_VALUE_SESSION — 此统计类型返回自开始收集数据以来 （自打开连接以来）活动的累计计数。不执行计算。例如，SMC_NAME_PAGE_IO 的 SMC_STAT_VALUE_SESSION 表示自会话开始以来发生的页 I/O 的数量。
- SMC_STAT_RATE_SAMPLE — 此统计类型计算每秒的速率。它返回最近一次采样间隔期间平均每秒钟发生某项活动的次数。例如，SMC_NAME_PAGE_IO 的 SMC_STAT_RATE_SAMPLE 表示在最近一次采样间隔期间平均每秒钟发生的页 I/O 的次数。

  计算方法是*最近一次采样间隔期间的计数* 除以*采样间隔持续的秒数*。
- SMC_STAT_RATE_SESSION — 此统计类型计算每秒的速率。它返回当前会话期间某项活动平均每秒发生的次数。例如，SMC_NAME_PAGE_IO 的 SMC_STAT_RATE_SESSION 是自会话开始以来平均每秒发生的页 I/O 次数。

  计算方法是*会话期间的计数* 除以*会话持续的秒数*。

- SMC_STAT_AVG_SAMPLE — 此统计类型计算某项活动在最近一次采样间隔期间每次发生时的平均值。只有几个数据项可使用此统计类型。返回值的意义取决于数据项名称。例如，SMC_NAME_STP_ELAPSED_TIME 的 SMC_STAT_AVG_SAMPLE 表示在最近一次采样间隔期间每次执行某一存储过程时的平均执行时间。
- SMC_STAT_AVG_SESSION — 此统计类型计算某项活动在会话期间每次发生时的平均值。只有几个数据项可使用此统计类型。返回值的意义取决于数据项名称。例如，SMC_NAME_STP_ELAPSED_TIME 的 SMC_STAT_AVG_SESSION 表示在记录会话期间某一存储过程每次执行时的平均执行时间。

注释 并非所有的统计类型对所有的数据项都有效。有关数据项及其使用规则的详细信息，请参见第 2 章 “数据项和统计类型”。

为连接创建视图

smc_create_view 在特定 Monitor 连接上创建视图。一个连接至少要有一个视图。

有关数据项有效组合的详细信息以及如何汇总数据项的信息，请参见第 2 章 “数据项和统计类型”。

可将视图视为表。视图中的数据项用表中的列来表示。特定视图返回的行数，取决于视图中的特定数据项。例如，具有全服务器范围数据的视图仅返回一行，而具有每个设备数据的视图为每个设备返回一行。

例如：

下面是一个由两个数据项组成的视图，它返回采样间隔期间每种锁类型的锁请求速率：

```
SMC_NAME_LOCK_TYPE, SMC_STAT_VALUE_SAMPLE
SMC_NAME_LOCK_COUNT, SMC_STAT_RATE_SAMPLE
```

下面是一个由一个数据项组成的视图，它返回所有锁类型在采样间隔期间的锁请求速率汇总值：

```
SMC_NAME_LOCK_COUNT, SMC_STAT_RATE_SAMPLE
```

如需有关数据项有效组合的完整详细信息以及了解如何汇总数据项，请参见第 2 章 “数据项和统计类型”。

第 4 步：创建过滤器

smc_create_filter 创建数据项的过滤器。过滤器限制视图返回的性能数据的行数。过滤器可应用到视图中指定的任意数据项。视图中每个数据项可包含一个过滤器。如果在视图中包含多个过滤器， Monitor Client Library 将使用 AND 来包含这些过滤器。

可用的过滤器类型如下：

- 等于 — 只返回与指定值之一相等的值 （对每项 “等于”比较执行逻辑 OR 运算）。
- 不等于 — 只返回与指定值中的任何值均不相等的值 （对每项 “不等于”比较执行逻辑 AND 运算）。
- 大于或等于 — 返回大于或等于指定值的值。
- 小于或等于 — 返回小于或等于指定值的值。
- 范围 — 值不小于下边界，且不大于上边界；返回上下边界值之间的数值，包括上下边界值。
- 前 N 个 — 返回最高的 N 个值。

一个视图可包含多个过滤器，但任何特定的数据项只能有一个过滤器与之绑定。当视图包含多个过滤器时，用 AND 来组合过滤器。

可随时添加或删除过滤器。下一次刷新时，过滤器的更改才会生效。

第 5 步：设置报警

smc_create_alarm_ex 可以在视图中的任意数值数据项 （ID 除外）上设置报警。在活动连接的特定数据项上指定报警时，应用程序会提供触发报警时要调用的回调函数。

Historical Server 不能调用回调函数，但每次触发报警时，它可写入日志文件或执行一个过程。

当报警触发时，应用程序可执行的操作类型的一个例子是将消息记录到日志中，这是 Historical Server 提供的功能之一。

可随时添加或删除报警。对报警的具体指定进行更改后，所做的更改从下次刷新时开始生效。

注释　Monitor Client Library 应用过滤器后，再应用报警。

第 6 步：请求性能数据和处理结果

所有连接、视图、报警和过滤器创建完成后，应用程序会请求性能数据的具体值。检索性能数据是一个分为三步的过程：

1 刷新数据。

2 检查行数。

3 查看视图中的每个数据项。

当 Monitor Client Library 应用程序需要检索数据时，它会启动一次刷新，该操作导致 Monitor Client Library 获取刷新数据。每次刷新后，应用程序在视图中逐项 （即表中的每一列）检索数据。

在给定连接上调用 smc_refresh_ex 后，应用程序将检索数据。

收集到的事件的数目决定刷新的频率。含有多个键的视图，刷新频率就需要比键少的视图高。下列症状可能预示应用程序的刷新频率不够大：

- 在 Monitor Server 错误日志中报告大量事件丢失。《Sybase Adaptive Server Enterprise Monitor Server 用户指南》中讨论了也可能有助于减少事件丢失的配置更改。
- 应用程序在调用 smc_refresh_ex 时发生挂起。视图中存在大量的键，造成 Monitor Server 来不及处理收集到的大量事件，因此不会返回控制权。因此， Monitor Server 开始消耗大量的 CPU 时间。

smc_get_row_count 决定视图中提供多少行的结果。视图返回的结果的形式实质上是一个表，其中可能包含很多 “行” 结果数据，但在某些情况下，可能包含零行数据。

smc_get_dataitem_value 检索视图单行单列的性能数据值。

刷新数据时，应用过滤器和报警。

对新性能数据的轮询是客户端驱动的，只受到数据提供系统和数据收集系统的速度限制。

第 7 步：关闭并释放连接

退出前， Monitor Client Library 应用程序必须：

- 关闭所有打开的连接。
- 释放每个连接。

关闭并释放连接

应用程序调用 smc_close 关闭连接，调用 smc_connect_drop 释放连接结构。释放未关闭的连接会出现错误。调用 smc_close 会导致下列 Monitor Client Library 隐含调用：

- 如有必要，可一次或多次调用 smc_drop_alarm 删除报警。
- 如有必要，可一次或多次调用 smc_drop_filter 删除过滤器。
- 一次或多次调用 smc_drop_view 以删除视图。

重新打开连接

应用程序关闭连接后，在释放连接结构前，可调用 smc_connect_ex 重新打开连接。

回放记录的数据

要从 Historical Server 检索记录的数据，过程与上述步骤类似，但有不同之处：

- 应用程序必须连接到 Historical Server。在建立连接前，将 smc_prop_servermode 设置为 SMC_SERVER_M_HISTORICAL。
- 连接以后，创建视图之前，应用程序必须调用 smc_create_playback_session。
- 创建所有视图后，应用程序必须调用 smc_initiate_playback。
- 回放记录的历史数据不允许报警。
- 不能删除视图和过滤器。
- 最后一次刷新后，应用程序必须调用 smc_terminate_playback。

Monitor Client Library 示例程序

此节包括一个 Monitor Client Library 示例程序清单，其过程为连接到服务器，发送查询，处理结果，然后退出。

示例程序

下面的示例程序 *monitor.c* 演示了前面部分概述的步骤。紧随示例之后提供了对每一步的注释。

```
/*monitor.c
 ** Example program showing logic flow of Monitor Client Library
 ** application. This example assumes the use of an ANSI C
 ** compliant compiler. This program creates two connections
 ** to the Monitor Server. Data is extracted from one connection
 ** at the beginning and end of the monitoring session.
 ** Data is extracted from the other connection every
 ** SAMPLE_INTERVAL seconds NUM_OF_SAMPLES times.
 */
 #include <stdio.h>
 #include <stdlib.h>
 #include <ctype.h>
/* The mcpublic.h header file contains function prototypes, etc.
 ** for monitor client library functions. It also includes a
 ** header file called mctypes.h, which defines the datatypes
 ** used for monitor client library applications.
 */
#include "mcpublic.h"
#define NUM_OF_SAMPLES 10
 #define SAMPLE_INTERVAL 5
 #define NUM_SERVER_DATA_ITEMS 3
 #define NUM_DB_INFO_ITEMS 14
 #define NUM_NW_INFO_ITEMS 6
 #define OPTIONAL_CALLS -1

/*Error signals*/
 #define VIEW_NONEXISTENT -1
 #define CONNECT_NONEXISTENT -1

SMC_RETURN_CODE main (SMC_INT argc, SMC_CHARP argv[])
 {
  SMC_VALUE_UNION serverNameUnion;
   SMC_VALUE_UNION userNameUnion;
   SMC_VALUE_UNION passwordUnion;
   SMC_VALUE_UNION interfacesFileUnion;
```

```
   SMC_VALUE_UNION workUnion;
   SMC_VALUE_UNION returnedDataUnion;
   SMC_CONNECT_ID connect1_id;
   SMC_CONNECT_ID connect2_id;
   SMC_VIEW_ID server_view_id;
   SMC_VIEW_ID db_info_view_id;
   SMC_VIEW_ID nw_info_view_id;

  SMC_RETURN_CODE ret;
   SMC_DATAITEM_TYPE dataitem_type;  /*Holds data item type
                                     returned by get_dataitem_type
                                  function call*/
/*Needed if alarms and filters are used */
 #ifdef OPTIONAL_CALLS
   SMC_ALARM_ID   alarm_id;
   SMC_FILTER_ID  filter_id;
   SMC_CHARP      filter_strings[2]; /*datatype is pointer to
                                     string. This is an array
                                     of pointers.*/
 #endif
   SMC_SIZET  row,num_of_rows,item;  /*This is an integer data
                                     type*/
   SMC_SIZET  outputLength;          /*Length of output returned
                                     by smc_connect_props
                                  function call*/
/*
 ** Definition of SMC_DATAITEM_STRUCT datatype
 */
   SMC_DATAITEM_STRUCT  server_info_view[NUM_SERVER_DATA_ITEMS];
   SMC_DATAITEM_STRUCT  db_info_view[NUM_DB_INFO_ITEMS];
   SMC_DATAITEM_STRUCT  nw_bytes_view[NUM_NW_INFO_ITEMS];

   SMC_VALUE_UNION   server_data[NUM_SERVER_DATA_ITEMS];
   SMC_VALUE_UNION   db_data[NUM_DB_INFO_ITEMS];
   SMC_VALUE_UNION   nw_data[NUM_NW_INFO_ITEMS];

/*Callback function prototypes. Actual functions are defined
 ** below.
 */
  SMC_VOID errorCallback(SMC_CONNECT_ID,SMC_COMMAND_ID,SMC_VOIDP);
  SMC_VOID alarmCallback(SMC_CONNECT_ID,SMC_COMMAND_ID,SMC_VOIDP);

  SMC_BOOL explicitInterfacesFile = FALSE;

  int index,iterations;
```

```
/*
 ** These are labels used when printing out data returned by the
 ** database info view.
 */
   SMC_CHARP db_info_labels[NUM_DB_INFO_ITEMS] = {
     "Database ID:  ",
     "Object ID:  ",
     "Database name:  ",
     "Object name:  ",
     "Page hit percent:  ",
     "Page I/O:  ",
     "Page logical reads this sample:  ",
     "Page logical reads this session:  ",
     "Page logical read rate this sample:  ",
     "Page logical read rate this session:  ",
     "Page physical reads this sample:  ",
     "Page physical reads this session:  ",
     "Page physical read rate this sample:  ",
     "Page physical read rate this session:  "
   };
/*
 ** These are labels used when printing out data returned by
 ** network info view.
 */
   SMC_CHARP nw_info_labels[NUM_NW_INFO_ITEMS] = {
     "Network bytes received this sample:  ",
     "Network bytes received this session:  ",
     "Network bytes sent this sample:  ",
     "Network bytes sent this session:  ",
     "Network byte I/O rate this sample:  ",
     "Network byte I/O rate this session:  "
   };
   if (argc <5){
     printf("Usage <%s> -U <user_name> [-P <password>]\
           -S <monserver name> [-I <interfaces_file>]\n",argv[0]);
     exit(1);
   }
/*
** Connect to a server.
*/
```

用于连接到服务器的代码

有关注释，请参见第 5 页的 “第 2 步：连接到服务器”。

```
/*
** Allocate first connection
*/
   ret=smc_connect_alloc(errorCallback,
                         &connect1_id   /*Pointer to connect_id!*/
                         );
   if (ret != SMC_RET_SUCCESS) {
     printf("Attempt to allocate first connection failed \
             with error %d.\n",ret);
     exit(1);
   }
/*
 ** Allocate second connection
 */
   ret=smc_connect_alloc(errorCallback,
                         &connect2_id   /*Pointer to connect_id!*/
                         );
   if (ret != SMC_RET_SUCCESS) {
     printf("Attempt to allocate second connection failed \
             with error %d.\n",ret);
     exit(1);
   }
/*
 ** Set mandatory and some optional connection properties.
 ** Mandatory connection properties are user name, server name,
 ** and password if user password is not NULL. If interfaces
 ** file name is not set, default is "interfaces" in directory
 ** pointed to by $SYBASE environment variable.
```

用于必需的连接属性的代码

有关注释，请参见第 6 页的 “必需的连接属性”。

```
*/

  for (index=1;index<argc;index++) {
/*User name*/
    if (strncmp(argv[index],"-U",2) == 0) {
       userNameUnion.stringValue = argv[index+1];
       ret=smc_connect_props(connect1_id,
                             SMC_PROP_ACT_SET, /*Property action*/
                             SMC_PROP_USERNAME,/*Property*/
                             &userNameUnion, /*Note that union,
                                          not member of union,
                                          is used for
                                          property value*/
```

```
                                SMC_NULLTERM,   /*Indicates null-
                                                terminated string
                                                for buffer length*/
                                NULL            /*Use NULL when
                                                setting a property*/
                                );
      }                              /*End if argument is user name*/
    if (ret != SMC_RET_SUCCESS) {
      printf("Could not set user name.\n");
      exit(SMC_RET_FAILURE);
    }
 /*Password. Default password is a null string*/    if (strncmp(argv[index
],"-P",2) == 0) {
       passwordUnion.stringValue = argv[index+1];
       ret=smc_connect_props(connect1_id,
                                SMC_PROP_ACT_SET, /*Property action*/
                                SMC_PROP_PASSWORD,/*Property*/
                                &passwordUnion,  /*Note that union,
                                                 not member of union,
                                                 is used for
                                                 property value*/
                                SMC_NULLTERM,    /*Indicates null-
                                                 terminated string
                                                 for buffer length*/
                                NULL             /*Use NULL when
                                                 setting a property*/
                                );
      }                               /*End if argument is password*/
    if (ret != SMC_RET_SUCCESS) {
      printf("Could not set password.\n");
      exit(SMC_RET_FAILURE);
    }
/*Server name*/
     if (strncmp(argv[index],"-S",2) == 0) {
       serverNameUnion.stringValue = argv[index+1];
       ret=smc_connect_props(connect1_id,
                              SMC_PROP_ACT_SET,  /*Property action*/
                              SMC_PROP_SERVERNAME,/*Property*/
                              &serverNameUnion, /*Note that union,
                                                not member of union,
                                                is used for
                                                property value*/
                               SMC_NULLTERM,    /*Indicates null-
                                                terminated string
                                                for buffer length*/
                               NULL             /*Use NULL when
```

```
                                                    setting a property*/
                                );
      }                          /*End if argument is server name*/
    if (ret != SMC_RET_SUCCESS) {
      printf("Could not set server name.\n");
      exit(SMC_RET_FAILURE);
    }
  /*Interfaces file. If unspecified, $SYBASE/interfaces is used*/
      if (strncmp(argv[index],"-I",2) == 0) {
        interfacesFileUnion.stringValue = argv[index+1];
        ret=smc_connect_props(connect1_id,
                              SMC_PROP_ACT_SET,  /*Property action*/
                              SMC_PROP_IFILE,    /*Property*/
                              &interfacesFileUnion, /*Note that
                                                    pointer to union,
                                                    not member of
                                                    union,is used for
                                                    property value*/
                               SMC_NULLTERM,     /*Indicates null-
                                                 terminated string
                                                 for buffer length*/
                               NULL              /*Use NULL when
                                                 setting a property*/
                               );
        explicitInterfacesFile = TRUE;
      }             /*End if argument is interfaces file pathname*/
    if (ret != SMC_RET_SUCCESS) {
      printf("Could not set interfaces file name.\n");
      printf("Using default interfaces file.\n");
    }
    }              /*End for loop getting connection properties
                 from command-line arguments*/
 /*
  ** Optional smc_get_connect_props call that sets a pointer to be
  ** passed to error callback. In this case, the pointer is to a
  ** string that tells which connection encountered the error.
  */
    workUnion.voidpValue = "first connection"; /*Call to set user
                                               data handle looks
                                               for value to set in
                                               void pointer member
                                               of union.*/
    ret=smc_connect_props(connect1_id,SMC_PROP_ACT_SET,\
                  SMC_PROP_USERDATA,&workUnion,SMC_NULLTERM,NULL);
    if (ret != SMC_RET_SUCCESS){
    printf("smc_connect_props call failed to \
```

```
            set userDataHandle.\n");
    }
  /*
   ** Demonstration of "get" mode for smc_get_connect_props
   */
   /*Check if user name has been set*/
    ret=smc_connect_props(connect1_id,
                        SMC_PROP_ACT_GET,/*Property action is "get"*/
                        SMC_PROP_USERNAME,
                        &workUnion,
                        SMC_UNUSED,     /*Length parameter ignored
                                          on "get" operations*/
                        &outputLength   /*Note this is a pointer!*/
                            );
    if (ret != SMC_RET_SUCCESS) {
      printf ("Could not get user name. Execution continuing.\n");
    }
    else {
      if (outputLength == 0) {
        printf("User name not set.  Quitting execution.\n");
        exit(SMC_RET_FAILURE);
      }
     else {
  /*
   ** Application is responsible for freeing
   ** memory allocated to string member of SMC_VALUE_UNION by
   ** library.
   */
      free(workUnion.stringValue);
     }
     }
   /*Check if server name has been set*/
    ret=smc_connect_props(connect1_id,
                        SMC_PROP_ACT_GET,/*Property action is "get"*/
                        SMC_PROP_SERVERNAME,
                        &workUnion,
                        SMC_UNUSED,      /*Length parameter ignored
                                         on "get" operations*/
                        &outputLength    /*Note this is a pointer!*/
                            );
     if (ret != SMC_RET_SUCCESS) {
      printf ("Could not get server name. Execution continuing.\n");
     }
     else {
       if (outputLength == 0) {
         printf("Server name not set. Quitting execution.\n");
```

```
      exit(SMC_RET_FAILURE);
     }
     else {
       free(workUnion.stringValue);
     }
   }
/*
 ** Allocate properties for second connection. No need to
 ** repeat error checking.
 */
  ret=smc_connect_props(connect2_id,SMC_PROP_ACT_SET, \
             SMC_PROP_USERNAME,&userNameUnion,SMC_NULLTERM, NULL);
   if (ret != SMC_RET_SUCCESS) {
     printf("Could not set user name for second connection.\n");
     exit(SMC_RET_FAILURE);
   }
   ret=smc_connect_props(connect2_id,SMC_PROP_ACT_SET, \
             SMC_PROP_PASSWORD,&passwordUnion,SMC_NULLTERM,NULL);
   if (ret != SMC_RET_SUCCESS) {
     printf("Could not set password for second connection.\n");
     exit(SMC_RET_FAILURE);
   }
   ret=smc_connect_props(connect2_id,SMC_PROP_ACT_SET, \
          SMC_PROP_SERVERNAME,&serverNameUnion,SMC_NULLTERM,NULL);
   if (ret != SMC_RET_SUCCESS) {
     printf("Could not set server name for second connection.\n");
     exit(SMC_RET_FAILURE);
   }
   if (explicitInterfacesFile) {
     ret=smc_connect_props(connect2_id,SMC_PROP_ACT_SET, \
           SMC_PROP_IFILE,&interfacesFileUnion,SMC_NULLTERM,NULL);
    if (ret != SMC_RET_SUCCESS) {
     printf("Could not set server name for second connection.\n");
     exit(SMC_RET_FAILURE);
     }
   }

/*
 ** Optional smc_connect_props call to set user-defined pointer to
 ** be passed to error callback. This pointer points to a
 ** string that tells where the error callback was triggered.
 */
 workUnion.voidpValue = "second connection"; /*Call to set user
                                               data handle looks for
                                               value to set in void
                                               pointer member
```

```
                                                    of union.*/
    ret=smc_connect_props(connect2_id,SMC_PROP_ACT_SET, \
                    SMC_PROP_USERDATA,&workUnion,SMC_NULLTERM,NULL);
    if (ret != SMC_RET_SUCCESS){
    printf("smc_connect_props call failed to set userDataHandle.\n");
      }
   /*
   ** Connect to monitor server
```

用于连接到服务器的代码

有关注释，请参见第 6 页的 “连接到服务器”。

```
   */
    /*
    ** First connection
    */
      ret=smc_connect_ex(connect1_id);
      if (ret != SMC_RET_SUCCESS) {
       printf("First connection failed to connect to \
             monitor server.\n");
       exit(SMC_RET_FAILURE);
      }


   /*
    ** Second connection
    */
      ret=smc_connect_ex(connect2_id);
      if (ret != SMC_RET_SUCCESS) {
        printf("Second connection failed to connect to \
             monitor server.\n");
        exit(SMC_RET_FAILURE);
      }



   /*
    ** Create views on connections.
    */
```

用于创建视图的代码　　有关注释，请参见第 6 页的 “第 3 步：创建视图”。

```
** Define views.
/*
** Each data item must be paired with a
 ** statistic type . View definitions are used in create_view
 ** calls after connecting to monitor server.
 */
 /*This is a server-
wide view that returns one row of data*/ server_info_view[0].dataItemName
=SMC_NAME_SQL_SERVER_NAME;
  server_info_view[0].dataItemStatType = SMC_STAT_VALUE_SAMPLE;
  server_info_view[1].dataItemName = SMC_NAME_SQL_SERVER_VERSION;
  server_info_view[1].dataItemStatType = SMC_STAT_VALUE_SAMPLE;
  server_info_view[2].dataItemName = SMC_NAME_TIMESTAMP;
  server_info_view[2].dataItemStatType = SMC_STAT_VALUE_SAMPLE;
/*
 ** This is a view with key and result data items that returns
 ** multiple rows of data.
 */
db_info_view[0].dataItemName = SMC_NAME_DB_ID; /*Key data items*/
 db_info_view[0].dataItemStatType = SMC_STAT_VALUE_SAMPLE;
 db_info_view[1].dataItemName = SMC_NAME_OBJ_ID;
 db_info_view[1].dataItemStatType = SMC_STAT_VALUE_SAMPLE;
 db_info_view[2].dataItemName = SMC_NAME_DB_NAME;    /*Result data
                                                     items*/
 db_info_view[2].dataItemStatType = SMC_STAT_VALUE_SAMPLE;
 db_info_view[3].dataItemName = SMC_NAME_OBJ_NAME;
 db_info_view[3].dataItemStatType = SMC_STAT_VALUE_SAMPLE;
 db_info_view[4].dataItemName = SMC_NAME_PAGE_HIT_PCT;
 db_info_view[4].dataItemStatType = SMC_STAT_VALUE_SAMPLE;
 db_info_view[5].dataItemName =SMC_NAME_PAGE_IO;
 db_info_view[5].dataItemStatType = SMC_STAT_VALUE_SAMPLE;
 db_info_view[6].dataItemName = SMC_NAME_PAGE_LOGICAL_READ;
 db_info_view[6].dataItemStatType = SMC_STAT_VALUE_SAMPLE;
 db_info_view[7].dataItemName = SMC_NAME_PAGE_LOGICAL_READ;
 db_info_view[7].dataItemStatType = SMC_STAT_VALUE_SESSION;
 db_info_view[8].dataItemName = SMC_NAME_PAGE_LOGICAL_READ;
 db_info_view[8].dataItemStatType = SMC_STAT_RATE_SAMPLE;
 db_info_view[9].dataItemName = SMC_NAME_PAGE_LOGICAL_READ;
 db_info_view[9].dataItemStatType = SMC_STAT_RATE_SESSION;
 db_info_view[10].dataItemName = SMC_NAME_PAGE_PHYSICAL_READ;
 db_info_view[10].dataItemStatType = SMC_STAT_VALUE_SAMPLE;
 db_info_view[11].dataItemName = SMC_NAME_PAGE_PHYSICAL_READ;
 db_info_view[11].dataItemStatType = SMC_STAT_VALUE_SESSION;
 db_info_view[12].dataItemName = SMC_NAME_PAGE_PHYSICAL_READ;
 db_info_view[12].dataItemStatType = SMC_STAT_RATE_SAMPLE;
```

```
 db_info_view[13].dataItemName = SMC_NAME_PAGE_PHYSICAL_READ;
 db_info_view[13].dataItemStatType = SMC_STAT_RATE_SESSION;
/*
 ** Another server-wide view
 */
   nw_bytes_view[0].dataItemName = SMC_NAME_NET_BYTES_RCVD;
   nw_bytes_view[0].dataItemStatType = SMC_STAT_VALUE_SAMPLE;
   nw_bytes_view[1].dataItemName = SMC_NAME_NET_BYTES_RCVD;
   nw_bytes_view[1].dataItemStatType = SMC_STAT_VALUE_SESSION;
   nw_bytes_view[2].dataItemName = SMC_NAME_NET_BYTES_SENT;
   nw_bytes_view[2].dataItemStatType = SMC_STAT_VALUE_SAMPLE;
   nw_bytes_view[3].dataItemName = SMC_NAME_NET_BYTES_SENT;
   nw_bytes_view[3].dataItemStatType = SMC_STAT_VALUE_SESSION;
   nw_bytes_view[4].dataItemName = SMC_NAME_NET_BYTE_IO;
   nw_bytes_view[4].dataItemStatType = SMC_STAT_RATE_SAMPLE;
   nw_bytes_view[5].dataItemName = SMC_NAME_NET_BYTE_IO;
   nw_bytes_view[5].dataItemStatType = SMC_STAT_RATE_SESSION;

ret=smc_create_view (connect1_id,      /*Connect ID assigned when
                                        connect allocated*/
                     server_info_view, /*This is a pointer to
                                  array of SMC_DATAITEM_STRUCTS
                                  which defines the view*/
                     NUM_SERVER_DATA_ITEMS, /*No. of items in
                                            the view*/
                     "server info view",    /*Ignored on a live
                                            connection*/
                     &server_view_id        /*Value is assigned
                                            by this call*/
                     );
 if (ret != SMC_RET_SUCCESS) {              /*Cleanup from failed
                                            create_view call*/
   ret=smc_connect_drop(connect1_id);       /*Create view failed
                                            so no further use for
                                            this connection*/
   connect1_id = CONNECT_NONEXISTENT;
 }
 /*
 ** The second connection will have two views
 */
   ret=smc_create_view(connect2_id,db_info_view,NUM_DB_INFO_ITEMS,
                       "db info view",&db_info_view_id);
   if (ret != SMC_RET_SUCCESS) {
     db_info_view_id = VIEW_NONEXISTENT;
   }
   ret=smc_create_view(connect2_id,nw_bytes_view,NUM_NW_INFO_ITEMS,
```

```
                        "nw bytes view",&nw_info_view_id);
   if (ret != SMC_RET_SUCCESS) {
     nw_info_view_id = VIEW_NONEXISTENT;
   }
/*
** Create a filter.
*/
```

用于创建过滤器的代码　有关注释，请参见第 9 页的 “第 4 步：创建过滤器”。

```
/*
 ** Filters and alarms may be applied to data items within a view.
 ** This is optional.
 ** In this case, we only want to see I/O activity for a
 ** particular database and tempdb. If any physical reads occur,
 ** an alarm is triggered that posts a message to the screen.
 */


#ifdef OPTIONAL_CALLS
   filter_strings[0] = "my_db";     /*Change to db of interest*/
   filter_strings[1] = "tempdb";
   workUnion.voidpValue = filter_strings;
   ret=smc_create_filter(connect2_id,        /*Connection id*/
                         db_info_view_id,    /*View id*/
                         &db_info_view[2],   /*Pointer to a data
                                             item within the view
                                             to be filtered*/
                         SMC_FILT_T_EQ,      /*Type of filter*/
                         &workUnion,         /*Filter value*/
                         2,                  /*Number of elements
                                             in array of filter
                                             values*/
                         SMC_DI_TYPE_CHARP, /*datatype of filter
                                             values*/
                         &filter_id          /*Value is assigned by
                                             this function call*/
                         );
   if (ret != SMC_RET_SUCCESS) {
     printf("Filters were not applied. Continuing execution.\n");
   }
/*
** Set alarms.

*/
```

用于设置报警的代码　　有关注释，请参见第 9 页的 “第 5 步：设置报警”。

```
   workUnion.longValue = 1;                    /*Value above which
                                                  alarm is triggered*/
      ret=smc_create_alarm_ex(connect2_id,     /*Connection id*/
                              db_info_view_id, /*View id*/
                              &db_info_view[11], /*Pointer to a data
                                                  item within the view
                                                  to which the alarm
                                                  is applied*/
                              &workUnion,      /*Where value that
                                                  triggers the alarm
                                                  is located*/
                              SMC_DI_TYPE_LONG, /*datatype of item
                                                  to which alarm is
                                                  applied*/
                              SMC_ALARM_A_NOTIFY,/*Trigger alarm
                                                  callback function.
                                                  This is the only
                                                  action possible when
                                                  the server mode is
                                                  LIVE.*/
                              NULL,    /*For server mode HISTORICAL,
                                       this is where log file to be
                                       written to or program to be
                                       run is specified. For server
                                       mode LIVE, this field is
                                       ignored.*/
   /*The following is a string that is passed to the alarm callback function.
   */
                            "Physical read occurred in database.",
                             alarmCallback,     /*Alarm callback
                                                function*/
                             &alarm_id          /*Variable into which
                                                alarm id is placed.*/
                             );
      if (ret != SMC_RET_SUCCESS) {
        printf("Alarm was not applied. Execution continuing.\n");
      }
    #endif
   /*
   ** Request data and process results.
   */
```

用于请求性能数据和处理结果的代码

有关注释，请参见第 10 页的 “第 6 步：请求性能数据和处理结果”。

```
/*
 ** Get data from first connection. As server name and version
 ** do not change during the connection, we only get it once.
 ** Post the time when the refresh was done.
 */
   if (connect1_id != CONNECT_NONEXISTENT) { /*If the connect is
                                              not successful,the
                                              error callback is
                                              triggered. For a
                                              friendlier display,
                                              we check first.*/
     ret=smc_refresh_ex(connect1_id,         /*ID of connect*/
                        0                    /*STEP not used in
                                               live connection*/
                        );
     if (ret != SMC_RET_SUCCESS) {
       printf("refresh call failed on first connect ID.\n");
     }
     else {                 /*Check row count even though only one
                              row is expected in this case. If no
                              rows are returned, get_dataitem_value
                              calls will return errors.*/
     ret=smc_get_row_count(connect1_id,
                           server_view_id,
                           #_of_rows);
     if (ret != SMC_RET_SUCCESS){
      printf("Get row count call failed.\n");
     }
     else {
      if (num_of_rows > 0){
/*
 ** A get_dataitem_value call is made for each item in the view.
 ** The retrieved data is stored in an array of SMC_VALUE_UNIONs.
 */
          for (index=0;index <NUM_SERVER_DATA_ITEMS;index++){
            ret=smc_get_dataitem_value(connect1_id,
                                server_view_id,
                                &server_info_view[index],/*Look at
                                                           each data
                                                           item in
                                                           the view*/
                                0,              /*Only one row of
                                                  data is returned for
                                                  this particular view,
```

```
                                                so the value for row
                                                is hard-coded in this
                                                case.*/
                                    &server_data[index]   /*Retrieved
                                                           data stored
                                                           here*/
                                    );
            }                                       /*End for loop*/
    /*
     ** Display the returned data.
     */
            printf("Adaptive Server Enterprise name is: \
                    %s.\n",server_data[0].stringValue);
             printf("Adaptive Server Enterprise version is: \
                    %s.\n",server_data[1].stringValue);
             printf("Date and time is: \
                  %s.\n",server_data[2].stringValue);
    /*
     ** The application is responsible for freeing memory allocated
     ** by the Monitor Client Library for string members of
     ** SMC_VALUE_UNIONs. This also illustrates the use of the
     ** smc_get_dataitem_type function call.
     */
      for (index=0;index <NUM_SERVER_DATA_ITEMS;index++){
        ret=smc_get_dataitem_type(&server_info_view[index], \
                                  &dataitem_type);
        if (ret != SMC_RET_SUCCESS) {
          printf("Get dataitem type failed for item %d \
                    in server_info_view.\n");
        }
        else {
          if (dataitem_type == SMC_DI_TYPE_CHARP) {
            free(server_data[index].stringValue);
          }
        }
      }                 /*End for loop*/
             }          /*End if number of rows > 0*/
           }            /*End case get_row_count was successful*/
         }              /*End case smc_refresh_ex call was successful*/
       }                /*End case connect still valid*/
     /*
     ** Get the data from the views in the second connection to see
     ** how the data changes over time. To do this, we sample
     ** NUM_OF_SAMPLES times, pausing SAMPLE_INTERVAL times between
     ** each sample. The process of retrieving data is within a loop.
     */
```

```
for (iterations=0;iterations<NUM_OF_SAMPLES;iterations++){
   sleep(SAMPLE_INTERVAL);
   ret=smc_refresh_ex(connect2_id,     /*Note second connection
                                       specified for refresh*/
                      0                /*Step not used in live
                                       connection*/
                      );
   if (ret == SMC_RET_SUCCESS) {
     if (db_info_view_id != VIEW_NONEXISTENT){ /*Attempting
                                               get_row_count for
                                               nonexistent view
                                               will cause errors
                                               so check if view
                                               was actually
                                               created*/
       ret=smc_get_row_count(connect2_id,
                             db_info_view_id,
                             #_of_rows   /*Multiple rows will
                                          be returned. For
                                          each row of data
                                          returned, use
                                          get_dataitem_value
                                          loop. Function call
                                          puts number of rows
                                          returned into
                                          variable.*/
                            );
       for(row=0;row<num_of_rows;row++){
         for (index=0;index <NUM_DB_INFO_ITEMS;index++){
           ret=smc_get_dataitem_value(connect2_id,
                          db_info_view_id, /*View specified for
                                           get_dataitem_value.*/
                          &db_info_view[index],
                          row,             /*Multiple rows in
                                           this case */
                          &db_data[index]
                          );
           if (ret != SMC_RET_SUCCESS) {
             printf("Get dataitem value failed for data item \
                     %s.\n",db_info_labels[index]);
           }
           else {
             printf("%s",db_info_labels[index]);
             ret=smc_get_dataitem_type(&db_info_view[index],\
                                       &dataitem_type);
             if (ret != SMC_RET_SUCCESS){
```

```
                printf("Get data item type failed for data item \
                        %s.\n",db_info_view[index]);
              }
              else {
                switch (dataitem_type) {
                case SMC_DI_TYPE_CHARP:
                  printf("%s.\n",db_data[index].stringValue);
                  free(db_data[index].stringValue);
                  /*Application is responsible for freeing
                  memory allocated for strings by library*/
                  break;
                case SMC_DI_TYPE_LONG:
                  printf("%d.\n",db_data[index].longValue);
                  break;
                case SMC_DI_TYPE_DOUBLE: /*Rates are generally
                                         floating point variables*/
                  printf("%f.\n",db_data[index].doubleValue);
                  break;
                default:
                  printf("Unknown datatype encountered.\n");
                  break;
                }    /*End switch*/
              }      /*End case get_dataitem_type successful*/
            }        /*End case get_dataitem_value successful*/
          }          /*End for loop to get each data item value*/
        }            /*End for loop to get each row of data*/
      }            /*End case view exists*//*
** Retrieve data from second view in refresh.
** Processing is much the same.
*/
      if (nw_info_view_id != VIEW_NONEXISTENT){ /*Attempting
                                                  get_row_count for
                                                  nonexistent view
                                                  causes errors, so
                                                  check to see if
                                                  view was actually
                                                  created*/
        ret=smc_get_row_count(connect2_id,
                              nw_info_view_id,
                              #_of_rows    /*This is a server-
                                              wide view so only
                                              one row should be
                                              returned*/
                             );
        if (num_of_rows > 0 ){
          for (index=0;index <NUM_NW_INFO_ITEMS;index++){
```

```
    ret=smc_get_dataitem_value(connect2_id,
                    nw_info_view_id, /*Note view
                                     specified for
                                  get_dataitem_value*/
                    &nw_bytes_view[index],
                    0,          /*One row in this case*/
                    &nw_data[index]
                    );
    if (ret != SMC_RET_SUCCESS) {
      printf("Get dataitem value failed for data item \
              %s.\n",nw_info_labels[index]);
    }
    else {
      printf("%s",nw_info_labels[index]);
      ret=smc_get_dataitem_type(&nw_bytes_view[index],\
                                &dataitem_type);
      if (ret != SMC_RET_SUCCESS){
        printf("Get data item type failed for data item \
                %s.\n",nw_bytes_view[index]);
      }
      else {
        switch (dataitem_type) {
        case SMC_DI_TYPE_CHARP:
          printf("%s.\n",nw_data[index].stringValue);
          free(nw_data[index].stringValue);
          /*Application is responsible for freeing
          memory allocated for strings by library*/
          break;
        case SMC_DI_TYPE_LONG:
          printf("%d.\n",nw_data[index].longValue);
          break;
        case SMC_DI_TYPE_DOUBLE:    /*Rates are generally
                                    floating point
                                    variables*/
          printf("%f.\n",nw_data[index].doubleValue);
          break;
        default:
          printf("Unknown datatype encountered.\n");
          break;
        }    /*End switch*/
      }      /*End case get_dataitem_type successful*/
    }        /*End case get_dataitem_value successful*/
  }          /*End for loop to get each data item value*/
}            /*End if any rows of data returned*/
else {
  printf("No data returned for network info view.\n");
```

```
        }
      }             /*End case view exists*/
    }               /*End case refresh successful*/
    else {
      printf("Refresh of second connect failed. \
              Return code is %d.\n",ret);
    }
  }                 /*End for loop for number of iterations*//*
 ** This shows how to drop filters and alarms. It is not necessary
 ** to do this prior to closing a connection, as it is done
 ** automatically when the connection is closed. Filters may be
 ** dropped, for example, to see the filtered results of a query
 ** followed by the unfiltered results.
 */
 #ifdef OPTIONAL_CALLS
  ret=smc_drop_filter(connect2_id,db_info_view_id,filter_id);
  if (ret != SMC_RET_SUCCESS) {
    printf("Attempt to drop filter failed.\n");
  }
  ret=smc_drop_alarm(connect2_id,db_info_view_id,alarm_id);
  if (ret != SMC_RET_SUCCESS) {
    printf("Attempt to drop alarm failed.\n");
  }
 #endif
/*
 ** Get another time stamp before disconnecting. To do this,
 ** do a refresh on the first connection again and only display
 ** the time stamp data returned.
 */
  if (connect1_id != CONNECT_NONEXISTENT) {
    ret=smc_refresh_ex(connect1_id,0 );
    if (ret != SMC_RET_SUCCESS) {
      printf("refresh call failed on first connect ID.\n");
    }
    else {                          /*Check row count even though
                                    only one row is expected. If
                                    no rows are returned,
                                    get_dataitem_value calls
                                    will return errors.*/
      ret=smc_get_row_count(connect1_id,
                            server_view_id,
                            #_of_rows);
      if (ret != SMC_RET_SUCCESS){
        printf("Get row count call on first connection \
                failed.\n");
      }
```

```
        else {
          if (num_of_rows > 0){
              ret=smc_get_dataitem_value(connect1_id,
                              server_view_id,
                              &server_info_view[2],  /*In this case
                                                     we are only
                                                     interested in
                                                     the third data
                                                     item*/
                              0,            /*Only one row of data
                                            is returned for this
                                            particular view, so the
                                            value for row is hard-
                                            coded in this case.*/
                              &server_data[2]
                              );
              printf("Date and time on conclusion of monitoring:\
                      %s\n",server_data[2].stringValue);
              free(server_data[2].stringValue);
              /*Application must free string memory returned
              by library*/
            }                 /*End if row of data returned*/
        }                     /*End case get_row_count successful*/
      }                       /*End case refresh successful*/
    }                         /*End case connection exists*/

  /*

  ** Close and deallocate the connection.
  */
```

用于关闭并释放连接的代码　有关注释，请参见第 11 页的 “第 7 步：关闭并释放连接”。

```
  /*
   ** Cleanup. This consists of closing all connections, then
   ** de-allocating them. Alternatively, connections can be re-used.
   */
     ret=smc_close(connect1_id,
                   SMC_CLOSE_REQUEST        /*Close only if no
                                            outstanding commands
                                            (only close request type
                                            currently supported)*/
                   );
     if (ret != SMC_RET_SUCCESS) {
       printf("Attempt to close first connection failed. \
               Return code is %d.\n",ret);
```

```
    }
    ret=smc_close(connect2_id,SMC_CLOSE_REQUEST);
    if (ret != SMC_RET_SUCCESS) {
      printf("Attempt to close second connection failed. \
              Return code is %d.\n",ret);
    }
  /*
   ** Connections can be re-used at this point, for example, to
   ** connect to different servers. However, we de-allocate them.
   */
    ret=smc_connect_drop(connect1_id);
    if (ret != SMC_RET_SUCCESS){
      printf("Attempt to drop first connection failed. \
              Return code is %d.\n",ret);
    }
    ret=smc_connect_drop(connect2_id);
    if (ret != SMC_RET_SUCCESS){
      printf("Attempt to drop second connection failed. \
              Return code is %d.\n",ret);
    }
    return(SMC_RET_SUCCESS);
  }                                          /*End main*/
  /*
  ** Callback functions
```

用于定义错误处理的代码

有关注释，请参见第 5 页的 “第 1 步：定义错误处理”。

```
  */
   SMC_VOID errorCallback(
     SMC_CONNECT_ID connectID,
     SMC_COMMAND_ID commandID,        /*Value internal to Monitor
                                      Client Library*/
     SMC_VOIDP userDataHandle         /*User-defined pointer. Set by
                                      smc_connect_propscall*/
     )
   {
     SMC_SIZET          ret;
     SMC_VALUE_UNION    errorInfo;    /*Used for getting information
                                      from smc_get_command_info
                                      function call*/
     SMC_SIZET          returned_msg_length;
     printf ("Inside new error callback.\n");
  /*
   ** Use smc_get_command_info function call to get information
   ** from error and alarm callbacks.
   */
```

```
ret=smc_get_command_info(connectID,
                      commandID,
                      SMC_INFO_ERR_MAPSEVERITY, /*Information
                                                requested about
                                                command*/
                      &errorInfo,         /*Where information
                                          returned about
                                          command is placed*/
                      NULL                /*Value is numeric
                                          so length of returned
                                    data not needed*/
                     );
if (ret != SMC_RET_SUCCESS){
  printf("get_command_info call requesting error map \
          severity failed. Error returned is:  %d\n",ret);
  }
else{
  printf("Monitor Client Library error severity level is: \
          %d\n",errorInfo.sizetValue);
}
ret=smc_get_command_info(connectID,
                         commandID,
                         SMC_INFO_ERR_MSG,
                         &errorInfo,
                         &returned_msg_length      /*Find string
                                                   length */

                );
if (ret != SMC_RET_SUCCESS){
  printf("get_command_info call requesting error message \
          failed. Error returned is:  %d\n",ret);
}
else{
  printf("Error message text is:  %s\n",errorInfo.stringValue);
  free(errorInfo.stringValue);
  /*Application is responsible for freeing string buffer
  memory allocated by library*/
}
ret=smc_get_command_info(connectID,
                          commandID,
                          SMC_INFO_ERR_NUM,
                          &errorInfo,
                          NULL
                          );
if (ret != SMC_RET_SUCCESS){
  printf("get_command_info call requesting error number \
```

```
            failed. Error returned is:  %d\n",ret);
    }
    else{
      printf("Error number is: %d\n",errorInfo.sizetValue);
    }
    ret=smc_get_command_info(connectID,
                             commandID,
                             SMC_INFO_ERR_SEVERITY,
                             &errorInfo,
                             NULL
                             );
    if (ret != SMC_RET_SUCCESS){
      printf("get_command_info call requesting error severity \
            failed.  Error returned is:  %d\n",ret);
    }
    else{
      printf("Error severity level is: %d\n",errorInfo.sizetValue);
    }
    ret=smc_get_command_info(connectID,
                             commandID,
                             SMC_INFO_ERR_SOURCE,
                             &errorInfo,
                             NULL
                             );
    if (ret != SMC_RET_SUCCESS){
      printf("get_command_info call requesting error source \
            failed.  Error returned is:  %d\n",ret);
    }
    else{
      printf(" Error source is:  %d\n",errorInfo.sizetValue);
    }
    ret=smc_get_command_info(connectID,
                             commandID,
                             SMC_INFO_ERR_STATE,
                             &errorInfo,
                             NULL
                             );
    if (ret != SMC_RET_SUCCESS){
      printf("get_command_info call requesting state failed. \
            Error returned is: %d\n",ret);
    }
    else{
      printf(" Error state is:  %d\n",errorInfo.sizetValue);
    }
  /*
   ** Demonstrate use of userDataHandle. This value was set as a
```

```
 ** connection property for the connection in the main program and
 ** is passed to this function.
 */
   if (userDataHandle != NULL){
     printf("Connection on which error occurred is \
             %s.\n",userDataHandle);
   }
 }                                          /*End errorCallback */
 /*Alarm callback*/
 SMC_VOID alarmCallback(
   SMC_CONNECT_ID connectID,
   SMC_COMMAND_ID commandID,           /*Value internal to Monitor
                                       Client Library*/
   SMC_VOIDP userDataHandle
   )
 {
 #define MSG_BUFFER_LENGTH 80
   SMC_SIZET          ret;
   SMC_VALUE_UNION    alarmInfo;      /*Union into which requested
                                      data is placed*/
   SMC_SIZET          returned_msg_length;
   printf ("Alarm callback triggered.\n");
/*
 ** Use smc_get_command_info function call to get information
 ** from error and alarm callbacks.
 */
   ret=smc_get_command_info(connectID,
                            commandID,
                            SMC_INFO_ALARM_ALARMID,
                            &alarmInfo,
                            NULL
                            );
   if (ret != SMC_RET_SUCCESS){
     printf("get_command_info call failed.  \
             Error returned is:  %d",ret);
   }
   else{
     printf("Alarm ID is:  %d\n",alarmInfo.sizetValue);
   }
/*
 ** This demonstrates the use of the SMC_INFO_ALARM_VALUE_DATATYPE
 ** information that might be useful in a generic alarm callback
 ** function.
 */
   ret=smc_get_command_info(connectID,
                            commandID,
```

```
                          SMC_INFO_ALARM_VALUE_DATATYPE,
                          &alarmInfo,
                          NULL
                          );
if (ret != SMC_RET_SUCCESS){
  printf("get_command_info call failed. \
          Error returned is:  %d",ret);
}
else{
  switch(alarmInfo.intValue){
  case SMC_DI_TYPE_INT:
    ret=smc_get_command_info(connectID,
                             commandID,
                             SMC_INFO_ALARM_CURRENT_VALUE,
                             &alarmInfo,
                             NULL
                             );
if (ret != SMC_RET_SUCCESS){
  printf("get_command_info call failed.  \
          Error returned is:  %d",ret);
}
else {
  printf("Current value of alarmed data item is:\
          %d.\n",alarmInfo.intValue);
}
break;
  case SMC_DI_TYPE_LONG:
    ret=smc_get_command_info(connectID,
                             commandID,
                             SMC_INFO_ALARM_CURRENT_VALUE,
                             &alarmInfo,
                             NULL
                             );
if (ret != SMC_RET_SUCCESS){
  printf("get_command_info call failed.  \
          Error returned is:  %d",ret);
}
else {
  printf("Current value of alarmed data item is: \
          %d.\n",alarmInfo.longValue);
}
break;
  case SMC_DI_TYPE_DOUBLE:
    ret=smc_get_command_info(connectID,
                             commandID,
                             SMC_INFO_ALARM_CURRENT_VALUE,
```

```
                                &alarmInfo,
                                NULL
                                );
 if (ret != SMC_RET_SUCCESS){
   printf("get_command_info call failed.  Error returned is:  %d",ret);
 }
 else {
   printf("Current value of alarmed data item is: \
           %f.\n",alarmInfo.doubleValue);
 }
 break;
   default:
   printf("Invalid value returned for datatype of \
           current alarm value.\n");
   break;
   }                                                  /*End switch*/
 }
 ret=smc_get_command_info(connectID,
                          commandID,
                          SMC_INFO_ALARM_ROW,
                          &alarmInfo,
                          NULL
                          );
if (ret != SMC_RET_SUCCESS){
   printf("get_command_info call failed.  \
           Error returned is:  %d",ret);
 }
 else{
   printf("Row of data which triggered alarm is: \
           %d\n",alarmInfo.sizetValue);
 }
 ret=smc_get_command_info(connectID,
                          commandID,
                          SMC_INFO_ALARM_VALUE_DATATYPE,
                          &alarmInfo,
                          NULL
                          );
 if (ret != SMC_RET_SUCCESS){
   printf("get_command_info call failed.  \
           Error returned is:  %d",ret);
 }
 else{
   switch(alarmInfo.intValue){
   case SMC_DI_TYPE_INT:
     ret=smc_get_command_info(connectID,
                              commandID,
```

```
                                SMC_INFO_ALARM_THRESHOLD_VALUE,
                                &alarmInfo,
                                NULL
                                );
if (ret != SMC_RET_SUCCESS){
  printf("get_command_info call failed. \
          Error returned is:  %d",ret);
}
else {
  printf("Value of data item exceeded alarm-triggering \
          value of: %d.\n",alarmInfo.intValue);
}
break;
  case SMC_DI_TYPE_LONG:
    ret=smc_get_command_info(connectID,
                                commandID,
                                SMC_INFO_ALARM_THRESHOLD_VALUE,
                                &alarmInfo,
                                NULL
                                );
if (ret != SMC_RET_SUCCESS){
  printf("get_command_info call failed. \
          Error returned is:  %d",ret);
}
else {
  printf("Value of data item exceeded alarm-triggering \
          value of: %d.\n",alarmInfo.longValue);
}
break;
  case SMC_DI_TYPE_DOUBLE:
    ret=smc_get_command_info(connectID,
                                commandID,
                                SMC_INFO_ALARM_THRESHOLD_VALUE,
                                &alarmInfo,
                                NULL
                                );
if (ret != SMC_RET_SUCCESS){
  printf("get_command_info call failed.  \
          Error returned is:  %d",ret);
}
else {
  printf("Value of data item exceeded alarm-triggering\
          value of: %f.\n",alarmInfo.doubleValue);
}
break;
  default:
```

```
      printf("Invalid value returned for datatype of \
              THRESHOLD alarm value.\n");
      break;
      }                                                  /*End switch*/
    }
    ret=smc_get_command_info(connectID,
                             commandID,
                             SMC_INFO_ALARM_TIMESTAMP,
                             &alarmInfo,
                             &returned_msg_length
                             );
    if (ret != SMC_RET_SUCCESS){
      printf("get_command_info call failed.  \
              Error returned is:  %d",ret);
     }
    else{
      printf("Time when alarm was triggered is: \
              %s\n",alarmInfo.stringValue);
      free(alarmInfo.stringValue); /*Application is responsible
                                    for freeing string buffer memory
                                    allocated by library.*/
    }

    ret=smc_get_command_info(connectID,
                             commandID,
                             SMC_INFO_ALARM_VIEWID,
                             &alarmInfo,
                             NULL
                             );
    if (ret != SMC_RET_SUCCESS){
      printf("get_command_info call failed.  \
              Error returned is:  %d",ret);
    }
    else{
      printf("ID of view which triggered alarm is: \
              %d.\n",alarmInfo.sizetValue);
    }
  }                                 /*End newAlarmCallback*/
```

第 2 章

数据项和统计类型

本章包含有关数据项和统计类型的信息。

概述

数据项是特殊的性能数据片段，可以通过使用 Monitor Client Library 获得。统计类型指定数据项执行的计算，以及要报告数据项收集的数据的持续时间。

本章描述数据项类型和统计类型。它还描述每个数据项及其特性。

结果和键数据项

数据项分为结果和键两类：

- *键数据项*可提升视图中的详细程度，并通常导致刷新视图时返回额外的行。包含每个后续键时，可以看作是在视图定义中添加了词语 “每个”。例如，假设最初为 “页 I/O”结果数据项。通过添加 “数据库”键数据项，即 “每个”数据库的页 I/O，可以提升粒度。通过添加 “对象”键数据项，即 “每个”数据库的 “每个”对象的页 I/O，可进一步提升粒度。

- *结果数据项*按视图中键数据项决定的详细程度返回性能数据。如果没有指定键数据项，那么只返回一行数据。

注释 数据项的“结果”或“键”称号是数据项的一个特性，与视图中该数据项的关联统计类型无关。

数据项和视图

视图通常既包括键数据项，又包括结果数据项。键和结果的混合在确定返回数据的细节数量方面具有很大的灵活性。全服务器范围数据是一个例外，如事务或网络活动数据。对于全服务器范围数据，不指定键数据项并且只返回一行数据。

表 2-1 显示视图返回数据的示例。

表 2-1：视图返回的数据示例

视图定义中包含的元素	返回内容
SMC_NAME_PAGE_IO	page I/Os for the whole server Row results: Page I/O --------- 145
SMC_NAME_SPID、 SMC_NAME_LOGIN_NAME、 SMC_NAME_PAGE_IO （其中 SPID 是键数据项）	page I/O per process Row results: SPID Login Name Page I/O ---------- 3 sa 45 5 joe 100
SMC_NAME_SPID、 SMC_NAME_DB_ID、 SMC_NAME_OBJ_ID、 SMC_NAME_DB_NAME、 SMC_NAME_OBJ_NAME 和 SMC_NAME_PAGE_IO（其中 SMC_NAME_SPID、 SMC_NAME_DB_ID 和 SMC_NAME_OBJID 是键数据项）	page I/O per database table per process Row results: SPID DBID ObjID DBName ObjName PageIO ---------- 1 5 208003772 pubs2 titles 10 1 5 336004228 pubs2 blurbs 5 5 5 22003430 pubs2 sales 100

不含数据的行与不含行的视图

当没有要报告的活动时，某些数据项在视图中导致出现一个空行 （也就是结果数据项为零值的行），而其它数据项导致行被省略。控制空行是否在视图中显示的规则是：

- 服务器级数据项总是返回一行，即使没有要报告的活动也如此。
- 包括键数据项 SMC_NAME_SPID 或 SMC_NAME_APPLICATION_NAME 的视图仅报告采样周期末尾活动的进程。
- 包括键数据项 SMC_NAME_OBJ_ID 或 SMC_NAME_ACT_STP_ID 的视图，在采样周期中没有要报告的活动时，会省略行。
- 包括未在前一项中列出的键的视图，即使在没有活动时也都返回行。

服务器级状态

某些数据项只可在服务器级使用。具有服务器级数据项的视图只包含结果数据项并提供在 Adaptive Server 上汇总的性能数据。

组合数据项

数据项不能随意组合。视图中键数据项的存在与否决定是否允许其它数据项出现在视图中。

如果视图中包括一个键数据项，那么视图中的所有结果数据项必须对该键数据项有效。同样，对于视图中每个结果数据项，其需要的所有键必须出现在视图中。

如果视图不包含键数据项，它可以包括任何不需要键的数据项。

结果和键的组合

在某些情况下，如果使用一个可选的键数据项，还必须使用一个或多个其它的数据项。在本章的数据项说明中，将具有此要求的数据项和其它必需的数据项一起放在括号中，并用加号 (+) 分开。

并非所有结果数据项都需要键数据项。如果视图只包括结果数据项，缺省情况下摘要处于服务器级。当视图中没有包括键数据项时，只有可选键的结果数据项可与服务器级的数据项一起使用。

要在视图中组合不同的结果数据项，请匹配公用键数据项。

连接摘要

某些视图消耗 Monitor Server 连接摘要。有关 Monitor Server 连接摘要的信息，请参见 《Adaptive Server Enterprise Monitor Server 用户指南》。

当前语句和应用程序名称数据项

要为当前语句数据项 (SMC_NAME_CUR_STMT_x) 或 SMC_NAME APPLICATION NAME 获取数据， Monitor Client 应用程序必须连接到 Monitor Server，并在启动要监控的应用程序前创建视图。

数据项定义

本节按字母顺序列出带有以下信息的数据项：

- 说明
- 服务器级状态
- 结果或键名称
- 对于结果数据项，必需的键和可选的键
- 对于键数据项，要求键数据项的结果数据项，以及可使用键数据项但并不要求它的结果数据项
- 版本兼容性：Adaptive Server 11.5 和更高版本
- 有效统计类型

有效统计类型如下：

- SMC_STAT_VALUE_SAMPLE
- SMC_STAT_VALUE_SESSION
- SMC_STAT_RATE_SAMPLE
- SMC_STAT_RATE_SESSION
- SMC_STAT_AVG_SAMPLE
- SMC_STAT_AVG_SESSION

数据项可能使用以下数据类型：

- LONG — 长整型
- ENUMS — 整型
- DOUBLE — 双精度
- CHARP — 字符型
- DATIM — 日期 / 时间

有关枚举类型的详细信息，请参见附录“数据类型和结构”。

注释　不是所有的统计类型都可用于每个数据项。

不能在同一视图中使用 SMC_NAME_SPID 和 SMC_NAME_APPLICATION_NAME。

数据项的名称说明

数据项名称的语法是它报告的信息说明的缩写。所有数据项都以 SMC_NAME 开头。名称的其余部分为英文单词和 / 或缩写。缩写及其含义如下：

- ACT — 活动的
- APP — 应用程序
- CNT — 计数 （数量）
- CUR — 当前
- DATIM — 日期和时间
- DB — 数据库
- DEV— 设备
- ID — 标识号
- IMMED — 立即
- IO — 输入 / 输出 （页读取和写入）
- KPID — 持久进程 ID
- MAX — 最大
- MEM — 内存

- NET — 网络
- NUM — 编号
- OBJ — 数据库对象
- PCT — 百分比
- PKT — 包
- PROC — 进程
- RCVD — 接收的
- REF — 引用的
- SPID — 服务器进程 ID
- STMT — 语句
- STP — 存储过程
- XACT — 事务

《Historical Server 用户指南》中描述的数据项等同于这些数据项，不过前者使用的是自然语言命名约定。

SMC_NAME_ACT_STP_DB_ID

说明 报告活动存储过程的数据库标识号。

版本兼容性 11.0 和更高版本

数据项类型 键

服务器级 无

需要此键的结果数据项

SMC_NAME_ACT_ STP_DB_NAME
SMC_NAME_ACT_ STP_NAME
SMC_NAME_ACT_ STP_OWNER_NAME
SMC_NAME_STP_CPU_TIME
SMC_NAME_STP_ELAPSED_TIME
SMC_NAME_STP_EXECUTION_CLASS
SMC_NAME_STP_LINE_TEXT
SMC_NAME_STP_NUM_TIMES_EXECUTED

可选用此键的结果数据项

SMC_NAME_LOCKS_GRANTED_IMMED
SMC_NAME_LOCKS_GRANTED_WAITED
SMC_NAME_LOCKS_NOT_GRANTED
SMC_NAME_PAGE_INDEX_LOGICAL_READ
SMC_NAME_PAGE_INDEX_PHYSICAL_READ
SMC_NAME_PAGE_HIT_PCT
SMC_NAME_PAGE_IO
SMC_NAME_PAGE_LOGICAL_READ
SMC_NAME_PAGE_PHYSICAL_READ
SMC_NAME_PAGE_WRITE

统计类型和数据类型

VALUE_ SAMPLE	VALUE_ SESSION	RATE_ SAMPLE	RATE_ SESSION	AVG_ SAMPLE	AVG_ SESSION
LONG					

SMC_NAME_ACT_STP_DB_NAME

说明　报告活动存储过程的数据库名称。

版本兼容性　11.0 和更高版本

数据项类型　结果

服务器级　无

必需的键　SMC_NAME_ACT_STP_DB_ID

可选键　无

统计类型和数据类型

VALUE_ SAMPLE	VALUE_ SESSION	RATE_ SAMPLE	RATE_ SESSION	AVG_ SAMPLE	AVG_ SESSION
CHARP					

SMC_NAME_ACT_STP_ID

说明 报告活动存储过程的标识号。

版本兼容性 11.0 和更高版本

数据项类型 键

服务器级 无

必需的键 SMC_NAME_ACT_STP_DB_ID

需要此键的结果数据项

SMC_NAME_ACT_ STP_NAME
SMC_NAME_ACT_ STP_OWNER_NAME
SMC_NAME_STP_CPU_TIME
SMC_NAME_STP_ELAPSED_TIME
SMC_NAME_STP_EXECUTION_CLASS
SMC_NAME_STP_LINE_TEXT
SMC_NAME_STP_NUM_TIMES_EXECUTED

可选用此键的结果数据项

SMC_NAME_LOCKS_GRANTED_IMMED
SMC_NAME_LOCKS_GRANTED_WAITED
SMC_NAME_LOCKS_NOT_GRANTED
SMC_NAME_PAGE_INDEX_LOGICAL_READ
SMC_NAME_PAGE_INDEX_PHYSICAL_READ
SMC_NAME_PAGE_HIT_PCT
SMC_NAME_PAGE_IO
SMC_NAME_PAGE_LOGICAL_READ
SMC_NAME_PAGE_PHYSICAL_READ
SMC_NAME_PAGE_WRITE

统计类型和数据类型

VALUE_ SAMPLE	VALUE_ SESSION	RATE_ SAMPLE	RATE_ SESSION	AVG_ SAMPLE	AVG_ SESSION
LONG					

SMC_NAME_ACT_STP_NAME

说明　报告活动存储过程的名称。

版本兼容性　11.0 和更高版本

数据项类型　结果

服务器级　无

必需的键　SMC_NAME_ACT_STP_DB_ID 和 SMC_NAME_ACT_STP_ID

可选键　无

统计类型和数据类型

VALUE_ SAMPLE	VALUE_ SESSION	RATE_ SAMPLE	RATE_ SESSION	AVG_ SAMPLE	AVG_ SESSION
CHARP					

SMC_NAME_ACT_STP_OWNER_NAME

说明　报告活动存储过程所有者的名称。

版本兼容性　11.0 和更高版本

数据项类型　结果

服务器级　无

必需的键　SMC_NAME_ACT_STP_DB_ID 和 SMC_NAME_ACT_STP_ID

可选键　无

统计类型和数据类型

VALUE_ SAMPLE	VALUE_ SESSION	RATE_ SAMPLE	RATE_ SESSION	AVG_ SAMPLE	AVG_ SESSION
CHARP					

SMC_NAME_APPLICATION_NAME

说明

报告为其累计其它统计信息的每个应用程序名。包含 SMC_NAME_APPLICATION_NAME 的视图仅报告采样周期末尾活动的进程。

SMC_NAME_APPLICATION_NAME 和 SMC_NAME_SPID 在同一视图中相互排斥。

版本兼容性

11.0 和更高版本

数据项类型

键

服务器级

无

需要此键的结果数据项

SMC_NAME_APP_EXECUTION_CLASS

可选用此键的结果数据项

SMC_NAME_CPU_PCT
SMC_NAME_CPU_TIME
SMC_NAME_LOCKS_GRANTED_IMMED
SMC_NAME_LOCKS_GRANTED_WAITED
SMC_NAME_LOCKS_NOT_GRANTED
SMC_NAME_NUM_PROCESSES
SMC_NAME_PAGE_INDEX_LOGICAL_READ
SMC_NAME_PAGE_INDEX_PHYSICAL_READ
SMC_NAME_PAGE_LOGICAL_READ
SMC_NAME_PAGE_PHYSICAL_READ
SMC_NAME_PAGE_WRITE

统计类型和数据类型

VALUE_ SAMPLE	VALUE_ SESSION	RATE_ SAMPLE	RATE_ SESSION	AVG_ SAMPLE	AVG_ SESSION
CHARP					

SMC_NAME_APP_EXECUTION_CLASS

说明

报告为给定的应用程序名配置的执行类 （如果有）。其名称以下列格式之一返回：

- 如果应用程序仅在作用域 NULL 内绑定到该执行类，将返回该执行类的名称。
- 如果应用程序在 NULL 作用域以及一个或多个登录名作用域内绑定到该执行类，则在该执行类的名称后面追加一个星号 (*)。
- 如果应用程序仅用一个或多个登录作用域绑定到执行类，将返回一个星号。

版本兼容性

11.0 和更高版本

数据项类型

结果

服务器级

无

必需的键

SMC_NAME_APPLICATION_NAME

可选键

无

统计类型和数据类型

VALUE_ SAMPLE	VALUE_ SESSION	RATE_ SAMPLE	RATE_ SESSION	AVG_ SAMPLE	AVG_ SESSION
CHARP					

SMC_NAME_BLOCKING_SPID

说明

报告持有如下锁的进程的标识号：由 SMC_NAME_SPID 数据项指示的进程正在等待的锁。如果进程未被阻塞，那么阻塞 SPID 为零。

版本兼容性

11.0 和更高版本

数据项类型

结果

服务器级

无

必需的键

SMC_NAME_SPID、 SMC_NAME_DB_ID、 SMC_NAME_OBJ_ID、 SMC_NAME_LOCK_STATUS

可选键

SMC_NAME_LOCK_TYPE、 SMC_NAME_PAGE_NUM

统计类型和数据类型

VALUE_ SAMPLE	VALUE_ SESSION	RATE_ SAMPLE	RATE_ SESSION	AVG_ SAMPLE	AVG_ SESSION
LONG					

SMC_NAME_CONNECT_TIME

说明 报告自进程开始后经历的时间 （以秒计）。如果在开始监控之前进程已经启动，连接时间就是已经监控此进程的时间。

版本兼容性 11.0 和更高版本

数据项类型 结果

服务器级 无

必需的键 SMC_NAME_SPID

可选键 无

统计类型和数据类型

VALUE_ SAMPLE	VALUE_ SESSION	RATE_ SAMPLE	RATE_ SESSION	AVG_ SAMPLE	AVG_ SESSION
LONG	LONG				

SMC_NAME_CPU_BUSY_PCT

说明 报告 Adaptive Server 处于忙状态时的时间百分比。

版本兼容性 11.0 和更高版本

数据项类型 结果

服务器级 有

必需的键 无

可选键 SMC_NAME_ENGINE_NUM

统计类型和数据类型

VALUE_ SAMPLE	VALUE_ SESSION	RATE_ SAMPLE	RATE_ SESSION	AVG_ SAMPLE	AVG_ SESSION
DOUBLE	DOUBLE				

SMC_NAME_CPU_PCT

说明　报告正在运行给定应用程序的一个进程或进程集处于 “运行”状态的时间 （所有进程都在 “运行”状态）百分比。

版本兼容性　11.0 和更高版本

数据项类型　结果

服务器级　无

必需的键　SMC_NAME_SPID 或 SMC_NAME_APPLICATION_NAME

注释　SMC_NAME_SPID 和 SMC_NAME_APPLICATION_NAME 是互斥的。

可选键　SMC_NAME_ENGINE_NUM

统计类型和数据类型

VALUE_ SAMPLE	VALUE_ SESSION	RATE_ SAMPLE	RATE_ SESSION	AVG_ SAMPLE	AVG_ SESSION
DOUBLE	DOUBLE				

SMC_NAME_CPU_TIME

说明　在服务器级 （没有键），报告服务器上总的 CPU “忙”时间。与键一起使用时，报告该忙时间中由每个进程、应用程序或引擎使用的时间。

版本兼容性　11.0 和更高版本

数据项类型　结果

服务器级　有

必需的键　无

可选键　SMC_NAME_ENGINE_NUM、 SMC_NAME_SPID 或 SMC_NAME_APPLICATION_NAME

注释　SMC_NAME_SPID 和 SMC_NAME_APPLICATION_NAME 是互斥的。

统计类型和数据类型

VALUE_ SAMPLE	VALUE_ SESSION	RATE_ SAMPLE	RATE_ SESSION	AVG_ SAMPLE	AVG_ SESSION
DOUBLE	DOUBLE				

SMC_NAME_CPU_YIELD

说明　报告 Adaptive Server 将控制权交给操作系统的次数。

版本兼容性　11.0 和更高版本

数据项类型　结果

服务器级　有

必需的键　无

可选键　SMC_NAME_ENGINE_NUM

统计类型和数据类型

VALUE_ SAMPLE	VALUE_ SESSION	RATE_ SAMPLE	RATE_ SESSION	AVG_ SAMPLE	AVG_ SESSION
LONG	LONG	DOUBLE	DOUBLE		

SMC_NAME_CUR_APP_NAME

说明　报告正在特定进程中执行的应用程序名。

版本兼容性　11.0 和更高版本

数据项类型　结果

服务器级　无

必需的键　SMC_NAME_SPID

可选键　无

统计类型和数据类型

VALUE_ SAMPLE	VALUE_ SESSION	RATE_ SAMPLE	RATE_ SESSION	AVG_ SAMPLE	AVG_ SESSION
CHARP					

SMC_NAME_CUR_ENGINE

说明　报告当前正在运行进程的 Adaptive Server 引擎数。

版本兼容性　11.0 和更高版本

数据项类型　结果

服务器级　无

必需的键　SMC_NAME_SPID

可选键　无

统计类型和数据类型

VALUE_ SAMPLE	VALUE_ SESSION	RATE_ SAMPLE	RATE_ SESSION	AVG_ SAMPLE	AVG_ SESSION
LONG					

SMC_NAME_CUR_EXECUTION_CLASS

说明　报告当前正在其名义下运行进程的执行类名称。

版本兼容性　11.5 和更高版本

数据项类型　结果

服务器级　无

必需的键　SMC_NAME_SPID

可选键　无

统计类型和数据类型

VALUE_ SAMPLE	VALUE_ SESSION	RATE_ SAMPLE	RATE_ SESSION	AVG_ SAMPLE	AVG_ SESSION
CHARP					

SMC_NAME_CUR_PROC_STATE

说明

报告进程的当前状态。可能的状态有：

- 无
- 报警休眠
- 后台
- 无效状态
- 已感染
- 锁休眠
- 接收休眠
- 远程 I/O
- 可运行的
- 运行
- 发送休眠
- 正在休眠
- 已停止
- 同步休眠
- 正在终止
- 正在放弃

版本兼容性

11.0 和更高版本

数据项类型

结果

服务器级

无

必需的键

SMC_NAME_SPID

可选键

无

统计类型和数据类型

VALUE_ SAMPLE	VALUE_ SESSION	RATE_ SAMPLE	RATE_ SESSION	AVG_ SAMPLE	AVG_ SESSION
ENUMS					

枚举

SMC_PROC_STATE

SMC_NAME_CUR_STMT_ACT_STP_DB_ID

说明　报告存储过程 （包括触发器，一种特殊类型的存储过程）的数据库 ID，被报告的存储过程包含当前正在为特定进程执行的 SQL 语句。如果当前正在执行的 SQL 语句未包含于存储过程内，则此 ID 为零。

版本兼容性　11.5 和更高版本

数据项类型　结果

服务器级　无

必需的键　SMC_NAME_SPID

可选键　无

统计类型和数据类型

VALUE_ SAMPLE	VALUE_ SESSION	RATE_ SAMPLE	RATE_ SESSION	AVG_ SAMPLE	AVG_ SESSION
LONG					

SMC_NAME_CUR_STMT_ACT_STP_DB_NAME

说明　报告存储过程 （包括触发器，一种特殊类型的存储过程）的数据库名称，被报告的存储过程包含当前正在为特定进程执行的 SQL 语句。如果当前正在执行的 SQL 语句未包含在存储过程内，则此名称为“**NoDatabase**”。

版本兼容性　11.5 和更高版本

数据项类型　结果

服务器级　无

必需的键　SMC_NAME_SPID

可选键　无

统计类型和数据类型

VALUE_ SAMPLE	VALUE_ SESSION	RATE_ SAMPLE	RATE_ SESSION	AVG_ SAMPLE	AVG_ SESSION
CHARP					

SMC_NAME_CUR_STMT_ACT_STP_ID

说明

报告存储过程（包括触发器，一种特殊类型的存储过程）的 ID，被报告的存储过程包含当前正在为特定进程执行的 SQL 语句。如果当前正在执行的 SQL 语句未包含于存储过程内，则此 ID 为零。

版本兼容性

11.5 和更高版本

数据项类型

结果

服务器级

无

必需的键

SMC_NAME_SPID

可选键

无

统计类型和数据类型

VALUE_ SAMPLE	VALUE_ SESSION	RATE_ SAMPLE	RATE_ SESSION	AVG_ SAMPLE	AVG_ SESSION
LONG					

SMC_NAME_CUR_STMT_ACT_STP_NAME

说明

报告存储过程（包括触发器，一种特殊类型的存储过程）的名称，被报告的存储过程包含当前正在为特定进程执行的 SQL 语句。如果当前正在执行的 SQL 语句未包含在存储过程内，则此名称为“**NoObject**”。

版本兼容性

11.5 和更高版本

数据项类型

结果

服务器级

无

必需的键

SMC_NAME_SPID

可选键

无

统计类型和数据类型

VALUE_ SAMPLE	VALUE_ SESSION	RATE_ SAMPLE	RATE_ SESSION	AVG_ SAMPLE	AVG_ SESSION
CHARP					

SMC_NAME_CUR_STMT_ACT_STP_OWNER_NAME

说明

报告存储过程 （包括触发器，一种特殊类型的存储过程）的所有者名称，被报告的存储过程包含当前正在为特定进程执行的 SQL 语句。如果当前正在执行的 SQL 语句未包含在存储过程内，则此名称为“**NoOwner**”。

版本兼容性

11.5 和更高版本

数据项类型

结果

服务器级

无

必需的键

SMC_NAME_SPID

可选键

无

统计类型和数据类型

VALUE_ SAMPLE	VALUE_ SESSION	RATE_ SAMPLE	RATE_ SESSION	AVG_ SAMPLE	AVG_ SESSION
CHARP					

SMC_NAME_CUR_STMT_ACT_STP_TEXT

说明

报告为特定进程执行的特定存储过程 （包括触发器，其是一种特殊类型的存储过程）的文本。如果 CUR_STMT_ACT_STP_DB_ID 和 CUR_STMT_ACT_STP_ID 都等于零，则当前没有执行存储过程，并且该文本为以空值终止的空字符串 ("")。

如果不能获得文本 （因为该存储过程已被编译，且其文本已被丢弃，或者因为文本是以加密格式存储的），那么该文本是以空值终止的空字符串 ("")。

版本兼容性

11.5 和更高版本

数据项类型

结果

服务器级

无

必需的键

SMC_NAME_SPID

可选键

无

统计类型和数据类型

VALUE_ SAMPLE	VALUE_ SESSION	RATE_ SAMPLE	RATE_ SESSION	AVG_ SAMPLE	AVG_ SESSION
CHARP					

SMC_NAME_CUR_STMT_BATCH_ID

说明 报告正为某个特定进程执行的特定查询批处理的 ID。

版本兼容性 11.5 和更高版本

数据项类型 结果

必需的键 SMC_NAME_SPID

可选键 无

统计类型和数据类型

VALUE_ SAMPLE	VALUE_ SESSION	RATE_ SAMPLE	RATE_ SESSION	AVG_ SAMPLE	AVG_ SESSION
LONG					

SMC_NAME_CUR_STMT_BATCH_TEXT

说明 报告正为某个特定进程执行的特定查询批处理的文本。该文本可以仅仅是某个查询批处理的完整文本的一个初始子字符串。此字段中存储的文本的最大数量由 Adaptive Server 配置选项 max SQL text monitored 决定，并且可以使用 SMC_NAME_CUR_STMT_BATCH_TEXXT ENABLED 加以监控。

版本兼容性 11.5 和更高版本

数据项类型 结果

服务器级 无

必需的键 SMC_NAME_SPID

可选键 无

统计类型和数据类型

VALUE_ SAMPLE	VALUE_ SESSION	RATE_ SAMPLE	RATE_ SESSION	AVG_ SAMPLE	AVG_ SESSION
CHARP					

SMC_NAME_CUR_STMT_BATCH_TEXT_ENABLED

说明

报告 Adaptive Server 是否正在保存当前执行的查询批处理的 SQL 文本，如果是，则报告保存的数量。

0 值 = 禁止保存 SQL 文本。

值为 1 或以上 = 每个服务器进程可保存的批处理文本的最大字节数。

版本兼容性

11.5 和更高版本

数据项类型

结果

服务器级

有

必需的键

无

可选键

无

统计类型和数据类型

VALUE_ SAMPLE	VALUE_ SESSION	RATE_ SAMPLE	RATE_ SESSION	AVG_ SAMPLE	AVG_ SESSION
LONG					

SMC_NAME_CUR_STMT_CONTEXT_ID

说明

报告在为特定进程执行的特定查询批处理内，唯一确定一个存储过程调用的 ID。

版本兼容性

11.5 和更高版本

数据项类型

结果

服务器级

无

必需的键

SMC_NAME_SPID

可选键

无

统计类型和数据类型

VALUE_ SAMPLE	VALUE_ SESSION	RATE_ SAMPLE	RATE_ SESSION	AVG_ SAMPLE	AVG_ SESSION
LONG					

SMC_NAME_CUR_STMT_CPU_TIME

说明 报告在运行状态下，当前正在执行的 SQL 语句所用的时间量 （以秒计）。

版本兼容性 11.5 和更高版本

数据项类型 结果

服务器级 无

必需的键 SMC_NAME_SPID

可选键 无

统计类型和数据类型：

VALUE_ SAMPLE	VALUE_ SESSION	RATE_ SAMPLE	RATE_ SESSION	AVG_ SAMPLE	AVG_ SESSION
DOUBLE	DOUBLE				

SMC_NAME_CUR_STMT_ELAPSED_TIME

说明 报告当前正在执行的 SQL 语句所用的时间量 （以秒计）。

版本兼容性 11.5 和更高版本

数据项类型 结果

服务器级 无

必需的键 SMC_NAME_SPID

可选键 无

统计类型和数据类型

VALUE_ SAMPLE	VALUE_ SESSION	RATE_ SAMPLE	RATE_ SESSION	AVG_ SAMPLE	AVG_ SESSION
LONG					

SMC_NAME_CUR_STMT_LINE_NUM

说明　报告 （在查询批处理或存储过程中）包括当前正在为特定进程执行的 SQL 语句起始语句在内的行数。如果 CUR_STMT_ACT_STP_DB_ID 和 CUR_STMT_ACT_STP_ID 都等于零，那么当前执行的 SQL 语句在查询批处理中。否则，当前执行的 SQL 语句在由这两个 ID 唯一标识的存储过程中。

版本兼容性　11.5 和更高版本

数据项类型　结果

服务器级　无

必需的键　SMC_NAME_SPID

可选键　无

统计类型和数据类型

VALUE_ SAMPLE	VALUE_ SESSION	RATE_ SAMPLE	RATE_ SESSION	AVG_ SAMPLE	AVG_ SESSION
LONG					

SMC_NAME_CUR_STMT_LOCKS_GRANTED_IMMED

说明　报告由当前执行的 SQL 语句请求且被立即授予或不再需要的 （因请求者已持有足够的锁）锁数目。

版本兼容性　11.5 和更高版本

数据项类型　结果

服务器级　无

必需的键　SMC_NAME_SPID

可选键　无

统计类型和数据类型

VALUE_ SAMPLE	VALUE_ SESSION	RATE_ SAMPLE	RATE_ SESSION	AVG_ SAMPLE	AVG_ SESSION
LONG	LONG	DOUBLE	DOUBLE		

SMC_NAME_CUR_STMT_LOCKS_GRANTED_WAITED

说明 报告由当前执行的 SQL 语句请求且在等待后授予的锁数。

版本兼容性 11.5 和更高版本

数据项类型 结果

服务器级 无

必需的键 SMC_NAME_SPID

可选键 无

统计类型和数据类型

VALUE_ SAMPLE	VALUE_ SESSION	RATE_ SAMPLE	RATE_ SESSION	AVG_ SAMPLE	AVG_ SESSION
LONG	LONG	DOUBLE	DOUBLE		

SMC_NAME_CUR_STMT_LOCKS_NOT_GRANTED

说明 报告由当前执行的 SQL 语句请求且被拒绝的锁数。

版本兼容性 11.5 和更高版本

数据项类型 结果

服务器级 无

必需的键 SMC_NAME_SPID

可选键 无

统计类型和数据类型

VALUE_ SAMPLE	VALUE_ SESSION	RATE_ SAMPLE	RATE_ SESSION	AVG_ SAMPLE	AVG_ SESSION
LONG	LONG	DOUBLE	DOUBLE		

SMC_NAME_CUR_STMT_NUM

说明

报告特定进程的当前正在执行的 SQL 语句的语句（出现在查询批处理或存储过程中）编号。如果 CUR_STMT_ACT_STP_DB_ID 和 CUR_STMT_ACT_STP_ID 都等于零，那么当前执行的 SQL 语句在查询批处理中。否则，当前执行的 SQL 语句在由这两个 ID 唯一标识的存储过程中。

零值表示当前执行的 SQL 语句的部分数据（也就是说，此 SQL 语句在监控开始前就开始执行了。可以获得有关的性能信息，但数字只能反映监控开始以后时间段内的性能）。

版本兼容性

11.5 和更高版本

数据项类型

结果

服务器级

无

必需的键

无

可选键

无

统计类型和数据类型

VALUE_ SAMPLE	VALUE_ SESSION	RATE_ SAMPLE	RATE_ SESSION	AVG_ SAMPLE	AVG_ SESSION
LONG					

SMC_NAME_CUR_STMT_PAGE_IO

说明

报告当前执行的 SQL 语句累计的组合逻辑页读取和页写入数目。

版本兼容性

11.5 和更高版本

数据项类型

结果

服务器级

无

必需的键

SMC_NAME_SPID

可选键

无

统计类型和数据类型

VALUE_ SAMPLE	VALUE_ SESSION	RATE_ SAMPLE	RATE_ SESSION	AVG_ SAMPLE	AVG_ SESSION
LONG	LONG	DOUBLE	DOUBLE		

SMC_NAME_CUR_STMT_PAGE_LOGICAL_READ

说明 报告当前执行的 SQL 语句累计的数据页读取数 （由高速缓存或设备读取满足）。

版本兼容性 11.5 和更高版本

数据项类型 结果

服务器级 无

必需的键 SMC_NAME_SPID

可选键 无

统计类型和数据类型

VALUE_ SAMPLE	VALUE_ SESSION	RATE_ SAMPLE	RATE_ SESSION	AVG_ SAMPLE	AVG_ SESSION
LONG	LONG	DOUBLE	DOUBLE		

SMC_NAME_CUR_STMT_PAGE_PHYSICAL_READ

说明 报告由当前执行的 SQL 语句累计但无法由数据高速缓存满足的数据页读取数。

版本兼容性 11.5 和更高版本

数据项类型 结果

服务器级 无

必需的键 SMC_NAME_SPID

可选键 无

统计类型和数据类型

VALUE_ SAMPLE	VALUE_ SESSION	RATE_ SAMPLE	RATE_ SESSION	AVG_ SAMPLE	AVG_ SESSION
LONG	LONG	DOUBLE	DOUBLE		

SMC_NAME_CUR_STMT_PAGE_WRITE

说明　报告当前执行的 SQL 语句累计的且已写入数据库设备的数据页数。

版本兼容性　11.5 和更高版本

数据项类型　结果

服务器级　无

必需的键　SMC_NAME_SPID

可选键　无

统计类型和数据类型

VALUE_ SAMPLE	VALUE_ SESSION	RATE_ SAMPLE	RATE_ SESSION	AVG_ SAMPLE	AVG_ SESSION
LONG	LONG	DOUBLE	DOUBLE		

SMC_NAME_CUR_STMT_QUERY_PLAN_TEXT

说明　报告为特定连接执行的特定查询的查询计划文本。

如果不能获得该文本（因为 Adaptive Server 已从其查询计划目录中删除了此计划），那么该文本是以空值终止的空字符串 ("")。

版本兼容性　11.5 和更高版本

数据项类型　结果

服务器级　无

必需的键　SMC_NAME_SPID

可选键　无

统计类型和数据类型

VALUE_ SAMPLE	VALUE_ SESSION	RATE_ SAMPLE	RATE_ SESSION	AVG_ SAMPLE	AVG_ SESSION
LONG	LONG	DOUBLE	DOUBLE		

SMC_NAME_CUR_STMT_START_TIME

说明

报告当前正在执行的 SQL 语句开始执行时的日期和时间 （以 Adaptive Server 中使用的时区表示）。

如果此 SQL 语句在开始监控前已开始运行，那么这是该语句首次遇到这种活动时的日期和时间。

版本兼容性

11.5 和更高版本

数据项类型

结果

服务器级

无

必需的键

SMC_NAME_SPID

可选键

无

统计类型和数据类型

VALUE_ SAMPLE	VALUE_ SESSION	RATE_ SAMPLE	RATE_ SESSION	AVG_ SAMPLE	AVG_ SESSION
DATM					

SMC_NAME_CUR_STMT_TEXT_BYTE_OFFSET

说明

报告在为特定进程执行的查询批处理或存储过程内，相对于某一条语句的开始位置的字节偏移。如果 CUR_STMT_ACT_STP_DB_ID 和 CUR_STMT_ACT_STP_ID 都等于 0，那么该语句就是查询批处理中当前正在执行的 SQL 语句。否则，该语句是由这两个 ID （上述）唯一确定的存储过程中当前正在执行的 SQL 语句。

版本兼容性

11.5 和更高版本

数据项类型

结果

服务器级

无

必需的键

SMC_NAME_SPID

可选键

无

统计类型和数据类型

VALUE_ SAMPLE	VALUE_ SESSION	RATE_ SAMPLE	RATE_ SESSION	AVG_ SAMPLE	AVG_ SESSION
LONG					

SMC_NAME_DATA_CACHE_CONTENTION

说明　报告被迫等待的数据高速缓存螺旋锁请求数所占的比例（*spinlock_waits* 除以 *spinlock_requests*）。

版本兼容性　11.0 和更高版本

数据项类型　结果

服务器级　无

必需的键　DATA_CACHE_ID

可选键　无

统计类型和数据类型

VALUE_ SAMPLE	VALUE_ SESSION	RATE_ SAMPLE	RATE_ SESSION	AVG_ SAMPLE	AVG_ SESSION
DOUBLE	DOUBLE				

SMC_NAME_DATA_CACHE_EFFICIENCY

说明　报告特定数据高速缓存的每兆字节的每秒高速缓存命中次数。

版本兼容性　11.0 和更高版本

数据项类型　结果

服务器级　无

必需的键　DATA_CACHE_ID

可选键　无

统计类型和数据类型

VALUE_ SAMPLE	VALUE_ SESSION	RATE_ SAMPLE	RATE_ SESSION	AVG_ SAMPLE	AVG_ SESSION
DOUBLE	DOUBLE				

SMC_NAME_DATA_CACHE_HIT

说明 报告从某个特定数据高速缓存实现的页读取次数。

版本兼容性 11.0 和更高版本

数据项类型 结果

服务器级 无

必需的键 DATA_CACHE_ID

可选键 无

统计类型和数据类型

VALUE_ SAMPLE	VALUE_ SESSION	RATE_ SAMPLE	RATE_ SESSION	AVG_ SAMPLE	AVG_ SESSION
LONG	LONG	DOUBLE	DOUBLE		

SMC_NAME_DATA_CACHE_HIT_PCT

说明 报告实现的页读取数的比率，可以用以下公式计算：

cache_hits / (*cache_hits* + *cache_misses*) * 100

版本兼容性 11.0 和更高版本

数据项类型 结果

服务器级 无

必需的键 DATA_CACHE_ID

可选键 无

统计类型和数据类型

VALUE_ SAMPLE	VALUE_ SESSION	RATE_ SAMPLE	RATE_ SESSION	AVG_ SAMPLE	AVG_ SESSION
DOUBLE	DOUBLE				

注释 当 SMC_NAME_DATA_CACHE_MISS 高估物理页读取数时，SMC_NAME_DATA_CACHE_HIT_PCT 会少报高速缓存命中百分比。

SMC_NAME_DATA_CACHE_ID

说明

报告数据高速缓存的 ID。数据库表或索引或二者都可以绑定到特定数据高速缓存上，或者一个数据库中的所有对象都可以绑定到同一个数据高速缓存上。任何对象都不能绑定到多个数据高速缓存上。

版本兼容性

11.0 和更高版本

数据项类型

键

服务器级

无

需要此键的结果数据项

SMC_NAME_DATA_CACHE_CONTENTION
SMC_NAME_DATA CACHE_EFFICIENCY
SMC_NAME_DATA_CACHE_HIT
SMC_NAME_DATA_CACHE_HIT_PCT
SMC_NAME_DATA_CACHE_LARGE_IO_DENIED
SMC_NAME_DATA_CACHE_LARGE_IO_PERFORMED
SMC_NAME_DATA_CACHE_LARGE_IO_REQUESTED
SMC_NAME_DATA_CACHE_MISS
SMC_NAME_DATA_CACHE_NAME
SMC_NAME_DATA_CACHE_PREFETCH_EFFICIENCY
SMC_NAME_DATA_CACHE_REF_AND_REUSE
SMC_NAME_DATA_CACHE_REUSE
SMC_NAME_DATA_CACHE_SIZE

可选用此键的结果数据项

SMC_NAME_DATA_CACHE_REUSE_DIRTY

统计类型和数据类型

VALUE_ SAMPLE	VALUE_ SESSION	RATE_ SAMPLE	RATE_ SESSION	AVG_ SAMPLE	AVG_ SESSION
LONG					

SMC_NAME_DATA_CACHE_LARGE_IO_DENIED

说明 报告当一次从磁盘中读取多个连续页并载入此数据高速缓存的缓冲区中时， Adaptive Server 缓冲区管理器不能满足优化程序的此类请求的次数。

版本兼容性 11.0 和更高版本

数据项类型 结果

服务器级 无

必需的键 DATA_CACHE_ID

可选键 无

统计类型和数据类型

VALUE_ SAMPLE	VALUE_ SESSION	RATE_ SAMPLE	RATE_ SESSION	AVG_ SAMPLE	AVG_ SESSION
LONG	LONG	DOUBLE	DOUBLE		

SMC_NAME_DATA_CACHE_LARGE_IO_PERFORMED

说明 报告当一次从磁盘中读取多个连续页并载入此数据高速缓存的缓冲区中时， Adaptive Server 缓冲区管理器可以满足优化程序的此类请求的次数。

版本兼容性 11.0 和更高版本

数据项类型 结果

服务器级 无

必需的键 DATA_CACHE_ID

可选键 无

统计类型和数据类型

VALUE_ SAMPLE	VALUE_ SESSION	RATE_ SAMPLE	RATE_ SESSION	AVG_ SAMPLE	AVG_ SESSION
LONG	LONG	DOUBLE	DOUBLE		

SMC_NAME_DATA_CACHE_LARGE_IO_REQUESTED

说明　报告优化程序请求 （Adaptive Server 缓冲区管理器）一次从磁盘中读取多个连续页并载入此数据高速缓存的缓冲区中的次数。

版本兼容性　11.0 和更高版本

数据项类型　结果

服务器级　无

必需的键　DATA_CACHE_ID

可选键　无

统计类型和数据类型

VALUE_ SAMPLE	VALUE_ SESSION	RATE_ SAMPLE	RATE_ SESSION	AVG_ SAMPLE	AVG_ SESSION
LONG	LONG	DOUBLE	DOUBLE		

SMC_NAME_DATA_CACHE_MISS

说明　报告从磁盘而非从某个特定数据高速缓存中实现的页读取次数。

版本兼容性　11.0 和更高版本

数据项类型　结果

服务器级　无

必需的键　DATA_CACHE_ID

可选键　无

统计类型和数据类型

VALUE_ SAMPLE	VALUE_ SESSION	RATE_ SAMPLE	RATE_ SESSION	AVG_ SAMPLE	AVG_ SESSION
LONG	LONG	DOUBLE	DOUBLE		

注释　SMC_NAME_DATA_CACHE_MISS 包括在页分配期间，尝试定位数据高速缓存中的页但失败的次数。因此，所报告的物理页读取数可能比实际值大。在此情况下，由 SMC_NAME_DATA_CACHE_HIT_PCT 报告的数据高速缓存未命中数百分比就比实际值小。

SMC_NAME_DATA_CACHE_NAME

说明

报告数据高速缓存的名称。数据库表或索引或二者都可以绑定到特定数据高速缓存上，或者一个数据库中的所有对象都可以绑定到同一个数据高速缓存上。任何对象都不能绑定到多个高速缓存上。

版本兼容性

11.0 和更高版本

数据项类型

结果

服务器级

无

必需的键

DATA_CACHE_ID

可选键

无

统计类型和数据类型

VALUE_ SAMPLE	VALUE_ SESSION	RATE_ SAMPLE	RATE_ SESSION	AVG_ SAMPLE	AVG_ SESSION
CHARP					

SMC_NAME_DATA_CACHE_PREFETCH_EFFICIENCY

说明

报告缓冲区中既被引用又被重用的页数比，相对于重用的给定高速缓存内缓冲区中的所有页。

如果该比值较大，则预取是有效的，否则预取就没有什么优点。这可能表示应该删除某个缓冲池 （或可能意味着某个表的聚簇索引是零散的，因此应删除该索引并重新创建它）。

注释 SMC_NAME_DATA_CACHE_PREFETCH_EFFICIENCY 会忽略每个高速缓存内的缺省缓冲池中的缓冲区。

版本兼容性

11.0 和更高版本

数据项类型

结果

服务器级

无

必需的键

DATA_CACHE_ID

可选键

无

统计类型和数据类型

VALUE_ SAMPLE	VALUE_ SESSION	RATE_ SAMPLE	RATE_ SESSION	AVG_ SAMPLE	AVG_ SESSION
DOUBLE	DOUBLE				

SMC_NAME_DATA_CACHE_REUSE

说明　报告重用缓冲区中的页数。该值大则表示高速缓存中的缓冲区周转率高，并说明某个缓冲池可能太小了。零值表明缺省缓冲池以外的一个缓冲池太大了。

版本兼容性　11.0 和更高版本

数据项类型　结果

服务器级　无

必需的键　DATA_CACHE_ID

可选键　无

统计类型和数据类型

VALUE_ SAMPLE	VALUE_ SESSION	RATE_ SAMPLE	RATE_ SESSION	AVG_ SAMPLE	AVG_ SESSION
LONG	LONG	DOUBLE	DOUBLE		

SMC_NAME_DATA_CACHE_REUSE_DIRTY

说明　报告被重用的缓冲区发生写入变化的次数。非零值表示清洗空间太小。

版本兼容性　11.0 和更高版本

数据项类型　结果

服务器级　无

必需的键　DATA_CACHE_ID

可选键　无

统计类型和数据类型

VALUE_ SAMPLE	VALUE_ SESSION	RATE_ SAMPLE	RATE_ SESSION	AVG_ SAMPLE	AVG_ SESSION
LONG	LONG	DOUBLE	DOUBLE		

SMC_NAME_DATA_CACHE_REF_AND_REUSE

说明 报告缓冲区中既被引用又被重用的页数。当确定预取缓冲区的效率时，将使用该计数（请参见 SMC_NAME_DATA_CACHE_PREFETCH_EFFICIENCY）。

版本兼容性 11.0 和更高版本

数据项类型 结果

服务器级 无

必需的键 DATA_CACHE_ID

可选键 无

统计类型和数据类型

VALUE_ SAMPLE	VALUE_ SESSION	RATE_ SAMPLE	RATE_ SESSION	AVG_ SAMPLE	AVG_ SESSION
LONG	LONG	DOUBLE	DOUBLE		

SMC_NAME_DATA_CACHE_SIZE

说明 报告数据高速缓存的大小（以兆字节为单位）。

版本兼容性 11.0 和更高版本

数据项类型 结果

服务器级 无

必需的键 DATA_CACHE_ID

可选键 无

统计类型和数据类型

VALUE_ SAMPLE	VALUE_ SESSION	RATE_ SAMPLE	RATE_ SESSION	AVG_ SAMPLE	AVG_ SESSION
DOUBLE					

SMC_NAME_DB_ID

说明　报告数据库的标识号。

版本兼容性　11.0 和更高版本

数据项类型　键

服务器级　无

需要此键的结果数据项

SMC_NAME_BLOCKING_SPID
SMC_NAME_DB_NAME
SMC_NAME_DEMAND_LOCK
SMC_NAME_LOCKS_BEING_BLOCKED_CNT
SMC_NAME_OBJ_NAME
SMC_NAME_OBJ_TYPE
SMC_NAME_OWNER_NAME
SMC_NAME_TIME_WAITED_ON_LOCK

可选用此键的结果数据项

SMC_NAME_LOCKS_GRANTED_IMMED
SMC_NAME_LOCKS_GRANTED_WAITED
SMC_NAME_LOCKS_NOT_GRANTED
SMC_NAME_PAGE_INDEX_LOGICAL_READ
SMC_NAME_PAGE_INDEX_PHYSICAL_READ
SMC_NAME_PAGE_HIT_PCT
SMC_NAME_PAGE_IO
SMC_NAME_PAGE_LOGICAL_READ
SMC_NAME_PAGE_PHYSICAL_READ
SMC_NAME_PAGE_WRITE

统计类型和数据类型

VALUE_ SAMPLE	VALUE_ SESSION	RATE_ SAMPLE	RATE_ SESSION	AVG_ SAMPLE	AVG_ SESSION
LONG					

SMC_NAME_DB_NAME

说明　报告数据库的名称。

版本兼容性　11.0 和更高版本

数据项类型　结果

服务器级　无

必需的键　DB_ID

可选键　无

统计类型和数据类型

VALUE_ SAMPLE	VALUE_ SESSION	RATE_ SAMPLE	RATE_ SESSION	AVG_ SAMPLE	AVG_ SESSION
CHARP					

SMC_NAME_DEADLOCK_CNT

说明　报告死锁的数量。

版本兼容性　11.0 和更高版本

数据项类型　结果

服务器级　有

必需的键　无

可选键　无

统计类型和数据类型

VALUE_ SAMPLE	VALUE_ SESSION	RATE_ SAMPLE	RATE_ SESSION	AVG_ SAMPLE	AVG_ SESSION
LONG	LONG				

SMC_NAME_DEMAND_LOCK

说明　报告表示某个锁是否已升级到请求锁状态的字符串 （Y 或 N）。

版本兼容性　11.0 和更高版本

数据项类型　结果

服务器级　无

必需的键　SMC_NAME_SPID、 SMC_NAME_DB_ID、 SMC_NAME_OBJ_ID、 SMC_NAME_LOCK_STATUS

可选键　SMC_NAME_LOCK_TYPE、 SMC_NAME_PAGE_NUM

统计类型和数据类型

VALUE_ SAMPLE	VALUE_ SESSION	RATE_ SAMPLE	RATE_ SESSION	AVG_ SAMPLE	AVG_ SESSION
CHARP					

SMC_NAME_DEV_HIT

说明　报告获得设备访问授权的次数。

版本兼容性　11.0 和更高版本

数据项类型　结果

服务器级　有

必需的键　无

可选键　SMC_NAME_DEV_NAME

统计类型和数据类型

VALUE_ SAMPLE	VALUE_ SESSION	RATE_ SAMPLE	RATE_ SESSION	AVG_ SAMPLE	AVG_ SESSION
LONG	LONG	DOUBLE	DOUBLE		

SMC_NAME_DEV_HIT_PCT

说明 报告已获准的设备请求数所占的比例，计算方法是用 SMC_NAME_DEV_HIT 除以 SMC_NAME_DEV_MISS 与 100 之积。

版本兼容性 11.0 和更高版本

数据项类型 结果

服务器级 有

必需的键 无

可选键 SMC_NAME_DEV_NAME

统计类型和数据类型

VALUE_ SAMPLE	VALUE_ SESSION	RATE_ SAMPLE	RATE_ SESSION	AVG_ SAMPLE	AVG_ SESSION
DOUBLE	DOUBLE				

SMC_NAME_DEV_IO

说明 报告设备读取数和设备写入数总计。

版本兼容性 11.0 和更高版本

数据项类型 结果

服务器级 有

必需的键 无

可选键 SMC_NAME_DEV_NAME

统计类型和数据类型

VALUE_ SAMPLE	VALUE_ SESSION	RATE_ SAMPLE	RATE_ SESSION	AVG_ SAMPLE	AVG_ SESSION
LONG	LONG	DOUBLE	DOUBLE		

SMC_NAME_DEV_MISS

说明　报告需等待获得设备访问授权的次数。

版本兼容性　11.0 和更高版本

数据项类型　结果

服务器级　有

必需的键　无

可选键　SMC_NAME_DEV_NAME

统计类型和数据类型

VALUE_ SAMPLE	VALUE_ SESSION	RATE_ SAMPLE	RATE_ SESSION	AVG_ SAMPLE	AVG_ SESSION
LONG	LONG	DOUBLE	DOUBLE		

SMC_NAME_DEV_NAME

说明　报告每个数据库设备的名称。

版本兼容性　11.0 和更高版本

数据项类型　键

服务器级　无

需要此键的结果数据项　无

可选用此键的结果数据项

SMC_NAME_DEV_HIT
SMC_NAME_DEV_HIT_PCT
SMC_NAME_DEV_IO
SMC_NAME_DEV_MISS
SMC_NAME_DEV_READ
SMC_NAME_DEV_WRITE

统计类型和数据类型

VALUE_ SAMPLE	VALUE_ SESSION	RATE_ SAMPLE	RATE_ SESSION	AVG_ SAMPLE	AVG_ SESSION
CHARP					

SMC_NAME_DEV_READ

说明 报告从某个数据库设备读取的数目。

版本兼容性 11.0 和更高版本

数据项类型 结果

服务器级 有

必需的键 无

可选键 SMC_NAME_DEV_NAME

统计类型和数据类型

VALUE_ SAMPLE	VALUE_ SESSION	RATE_ SAMPLE	RATE_ SESSION	AVG_ SAMPLE	AVG_ SESSION
LONG	LONG	DOUBLE	DOUBLE		

SMC_NAME_DEV_WRITE

说明 报告向某个数据库设备写入的数目。

版本兼容性 11.0 和更高版本

数据项类型 结果

服务器级 有

必需的键 无

可选键 SMC_NAME_DEV_NAME

统计类型和数据类型

VALUE_ SAMPLE	VALUE_ SESSION	RATE_ SAMPLE	RATE_ SESSION	AVG_ SAMPLE	AVG_ SESSION
LONG	LONG	DOUBLE	DOUBLE		

SMC_NAME_ELAPSED_TIME

说明　报告以秒计的时间增量，可以是从一次数据刷新到下一次 （采样）的时间，或者是从创建视图到当前会话的时间。

版本兼容性　11.0 和更高版本

数据项类型　结果

服务器级　有

必需的键　无

可选键　无

统计类型和数据类型

VALUE_ SAMPLE	VALUE_ SESSION	RATE_ SAMPLE	RATE_ SESSION	AVG_ SAMPLE	AVG_ SESSION
LONG	LONG				

SMC_NAME_ENGINE_NUM

说明　报告 Adaptive Server 引擎的编号。

版本兼容性　11.0 和更高版本

数据项类型　键

服务器级　无

需要此键的结果数据项　无

可选用此键的结果数据项

SMC_NAME_CPU_BUSY_PCT
SMC_NAME_CPU_PCT
SMC_NAME_CPU_TIME
SMC_NAME_CPU_YIELD
SMC_NAME_PAGE_INDEX_LOGICAL_READ
SMC_NAME_PAGE_INDEX_PHYSICAL_READ
SMC_NAME_PAGE_HIT_PCT
SMC_NAME_PAGE_IO
SMC_NAME_PAGE_LOGICAL_READ
SMC_NAME_PAGE_PHYSICAL_READ
SMC_NAME_PAGE_WRITE

统计类型和数据类型

VALUE_ SAMPLE	VALUE_ SESSION	RATE_ SAMPLE	RATE_ SESSION	AVG_ SAMPLE	AVG_ SESSION
LONG					

SMC_NAME_HOST_NAME

说明	报告与 Adaptive Server 建立特定连接的主机的名称。
版本兼容性	11.0 和更高版本
数据项类型	结果
服务器级	无
必需的键	SMC_NAME_SPID
可选键	无

统计类型和数据类型

VALUE_ SAMPLE	VALUE_ SESSION	RATE_ SAMPLE	RATE_ SESSION	AVG_ SAMPLE	AVG_ SESSION
CHARP					

SMC_NAME_KPID

说明	报告在长时间内保持唯一的 Adaptive Server 进程标识号。
版本兼容性	11.0 和更高版本
数据项类型	结果
服务器级	无
必需的键	SMC_NAME_SPID
可选键	无

统计类型和数据类型

VALUE_ SAMPLE	VALUE_ SESSION	RATE_ SAMPLE	RATE_ SESSION	AVG_ SAMPLE	AVG_ SESSION
LONG					

SMC_NAME_LOCK_CNT

说明　报告锁的数量。这是一个累计值。

版本兼容性　11.0 和更高版本

数据项类型　结果

服务器级　有

必需的键　无

可选键　SMC_NAME_SPID、 SMC_NAME_LOCK_TYPE、 SMC_NAME_LOCK_RESULT、 SMC_NAME_LOCK_RESULT_SUMMARY

统计类型和数据类型

VALUE_ SAMPLE	VALUE_ SESSION	RATE_ SAMPLE	RATE_ SESSION	AVG_ SAMPLE	AVG_ SESSION
LONG	LONG	DOUBLE	DOUBLE		

SMC_NAME_LOCK_HIT_PCT

说明　报告成功请求到锁的百分比。

版本兼容性　11.0 和更高版本

数据项类型　结果

服务器级　有

必需的键　无

可选键　无

统计类型和数据类型

VALUE_ SAMPLE	VALUE_ SESSION	RATE_ SAMPLE	RATE_ SESSION	AVG_ SAMPLE	AVG_ SESSION
DOUBLE	DOUBLE				

SMC_NAME_LOCK_RESULT

说明

报告逻辑锁请求的结果。锁结果值有：

- 立即授予。
- 不需要；请求者已拥有一个足够的锁。
- 已等待；请求者已等待。
- 不等待；不能立即获得锁，且请求者不想将该锁请求放入队列中。
- 死锁；请求者已成为死锁的牺牲品。
- 中断；锁请求被关注情况所中断。

版本兼容性

11.0 和更高版本

数据项类型

键

服务器级

无

需要此键的结果数据项

无

可选用此键的结果数据项

SMC_NAME_LOCK_CNT

统计类型和数据类型

VALUE_ SAMPLE	VALUE_ SESSION	RATE_ SAMPLE	RATE_ SESSION	AVG_ SAMPLE	AVG_ SESSION
ENUMS					

枚举

SMC_LOCK_RESULT

SMC_NAME_LOCK_RESULT_SUMMARY

说明

报告在一个已授予或未授予的等级上汇总的锁结果。

- 授予的锁结果摘要包括已授予、不需要和已等待锁结果。
- 未授予的锁结果摘要包括未等待、发生死锁和已中断的锁结果。

版本兼容性

11.0 和更高版本

数据项类型

键

服务器级

无

需要此键的结果数据项

无

可选用此键的结果数据项

SMC_NAME_LOCK_CNT

统计类型和数据类型

VALUE_ SAMPLE	VALUE_ SESSION	RATE_ SAMPLE	RATE_ SESSION	AVG_ SAMPLE	AVG_ SESSION
ENUMS					

枚举　SMC_LOCK_RESULT_SUMMARY

SMC_NAME_LOCK_STATUS

说明　报告锁的当前状态。锁的状态值为：

- 持有和阻塞
- 持有和未阻塞
- 已请求和已阻塞
- 已请求和未阻塞

版本兼容性　11.0 和更高版本

数据项类型　键

服务器级　无

需要此键的结果数据项

SMC_NAME_BLOCKING_SPID
SMC_NAME_DEMAND_LOCK
SMC_NAME_LOCK_STATUS_CNT
SMC_NAME_LOCKS_BEING_BLOCKED_CNT
SMC_NAME_TIME_WAITED_ON_LOCK

可选用此键的结果数据项　无

统计类型和数据类型

VALUE_ SAMPLE	VALUE_ SESSION	RATE_ SAMPLE	RATE_ SESSION	AVG_ SAMPLE	AVG_ SESSION
ENUMS					

枚举　SMC_LOCK_STATUS

SMC_NAME_LOCK_STATUS_CNT

说明 报告每个锁状态下的锁数目。这是一个快照值。

版本兼容性 11.0 和更高版本

数据项类型 结果

服务器级 无

必需的键 LOCK_STATUS

可选键 无

统计类型和数据类型

VALUE_ SAMPLE	VALUE_ SESSION	RATE_ SAMPLE	RATE_ SESSION	AVG_ SAMPLE	AVG_ SESSION
LONG	LONG	DOUBLE	DOUBLE		

SMC_NAME_LOCK_TYPE

说明 报告 Adaptive Server 使用的锁类型。 Adaptive Server 通过锁定来保护活动事务所使用的表和数据页。 Adaptive Server 使用以下锁类型：

- 排他表
- 共享表
- 排它意图
- 共享意图
- 排他页
- 共享页
- 更新页

版本兼容性 11.0 和更高版本

数据项类型 键

服务器级 无

需要此键的结果数据项 无

可选用此键的结果数据项

SMC_NAME_BLOCKING_SPID
SMC_NAME_DEMAND_LOCK
SMC_NAME_LOCK_CNT
SMC_NAME_LOCKS_BEING_BLOCKED_CNT
SMC_NAME_TIME_WAITED_ON_LOCK

统计类型和数据类型

VALUE_ SAMPLE	VALUE_ SESSION	RATE_ SAMPLE	RATE_ SESSION	AVG_ SAMPLE	AVG_ SESSION
ENUMS					

枚举　SMC_LOCK_TYPE

SMC_NAME_LOCKS_BEING_BLOCKED_CNT

说明　报告持有此 “hold_and_blocking”锁的进程所阻塞的锁数目。

版本兼容性　11.0 和更高版本

数据项类型　结果

服务器级　无

必需的键　SMC_NAME_SPID、 SMC_NAME_DB_ID、 SMC_NAME_OBJ_ID、 SMC_NAME_LOCK_STATUS

可选键　SMC_NAME_LOCK_TYPE、 SMC_NAME_PAGE_NUM

统计类型和数据类型

VALUE_ SAMPLE	VALUE_ SESSION	RATE_ SAMPLE	RATE_ SESSION	AVG_ SAMPLE	AVG_ SESSION
LONG					

SMC_NAME_LOCKS_GRANTED_IMMED

说明 报告无需等待释放其它锁就可立即授予的锁数目。

版本兼容性 11.5 和更高版本

数据项类型 结果

服务器级 有

必需的键 无

可选键 SMC_NAME_SPID、 SMC_NAME_APPLICATION_NAME、 [SMC_NAME_DB_ID + SMC_NAME_OBJ_ID]、 [SMC_NAME_CUR_STMT_ACT_STP_DB_ID + SMC_NAME_CUR_STMT_ACT_STP_ID]、 [SMC_NAME_ACT_STP_DB_ID + SMC_NAME_ACT_STP_ID]

注释 SMC_NAME_SPID 和 SMC_NAME_APPLICATION_NAME 是互斥的。如果使用 SMC_NAME_CUR_STMT_ACT_STP_DB_ID + SMC_NAME_CUR_STMT_ACT_STP_ID 组合键，则不能使用任何其它键。

统计类型和数据类型

VALUE_ SAMPLE	VALUE_ SESSION	RATE_ SAMPLE	RATE_ SESSION	AVG_ SAMPLE	AVG_ SESSION
LONG	LONG	DOUBLE	DOUBLE		

SMC_NAME_LOCKS_GRANTED_WAITED

说明 报告需等待释放其它锁后才授予的锁数目。

版本兼容性 11.5 和更高版本

数据项类型 结果

服务器级 有

必需的键 无

可选键

SMC_NAME_SPID、 SMC_NAME_APPLICATION_NAME、[SMC_NAME_DB_ID + SMC_NAME_OBJ_ID]、[SMC_NAME_CUR_STMT_ACT_STP_DB_ID + SMC_NAME_CUR_STMT_ACT_STP_ID]、[SMC_NAME_ACT_STP_DB_ID + SMC_NAME_ACT_STP_ID]

注释　SMC_NAME_SPID 和 SMC_NAME_APPLICATION_NAME 是互斥的。如果使用 SMC_NAME_CUR_STMT_ACT_STP_DB_ID + SMC_NAME_CUR_STMT_ACT_STP_ID 组合键，则不能使用任何其它键。

统计类型和数据类型

VALUE_ SAMPLE	VALUE_ SESSION	RATE_ SAMPLE	RATE_ SESSION	AVG_ SAMPLE	AVG_ SESSION
LONG	LONG	DOUBLE	DOUBLE		

SMC_NAME_LOCKS_NOT_GRANTED

说明

报告已请求但未授予的锁数目。

版本兼容性

11.5 和更高版本

数据项类型

结果

服务器级

有

必需的键

无

可选键

SMC_NAME_SPID、 SMC_NAME_APPLICATION_NAME、[SMC_NAME_DB_ID + SMC_NAME_OBJ_ID]、[SMC_NAME_CUR_STMT_ACT_STP_DB_ID + SMC_NAME_CUR_STMT_ACT_STP_ID]、[SMC_NAME_ACT_STP_DB_ID + SMC_NAME_ACT_STP_ID]

注释　SMC_NAME_SPID 和 SMC_NAME_APPLICATION_NAME 是互斥的。如果使用 SMC_NAME_CUR_STMT_ACT_STP_DB_ID + SMC_NAME_CUR_STMT_ACT_STP_ID 组合键，则不能使用任何其它键。

统计类型和数据类型

VALUE_ SAMPLE	VALUE_ SESSION	RATE_ SAMPLE	RATE_ SESSION	AVG_ SAMPLE	AVG_ SESSION
LONG	LONG	DOUBLE	DOUBLE		

SMC_NAME_LOG_CONTENTION_PCT

说明

报告当将用户日志高速缓存刷新到事务日志中时，必须等待日志信号的次数占总次数的百分比。

此值过高可能表示应增加用户日志高速缓存的大小。

版本兼容性

11.0 和更高版本

数据项类型

结果

服务器级

有

必需的键

无

可选键

无

统计类型和数据类型

VALUE_ SAMPLE	VALUE_ SESSION	RATE_ SAMPLE	RATE_ SESSION	AVG_ SAMPLE	AVG_ SESSION
DOUBLE	DOUBLE				

SMC_NAME_LOGIN_NAME

说明

报告与 Adaptive Server 进程相关联的登录名。

版本兼容性

11.0 和更高版本

数据项类型

结果

服务器级

无

必需的键

SMC_NAME_SPID

可选键

无

统计类型和数据类型

VALUE_ SAMPLE	VALUE_ SESSION	RATE_ SAMPLE	RATE_ SESSION	AVG_ SAMPLE	AVG_ SESSION
CHARP					

SMC_NAME_MEM_CODE_SIZE

说明　报告为 Adaptive Server 分配的内存字节数。

版本兼容性　11.0 和更高版本

数据项类型　结果

服务器级　有

必需的键　无

可选键　无

统计类型和数据类型

VALUE_ SAMPLE	VALUE_ SESSION	RATE_ SAMPLE	RATE_ SESSION	AVG_ SAMPLE	AVG_ SESSION
LONG					

SMC_NAME_MEM_KERNEL_STRUCT_SIZE

说明　报告为内核结构分配的内存字节数。

版本兼容性　11.0 和更高版本

数据项类型　结果

服务器级　有

必需的键　无

可选键　无

统计类型和数据类型

VALUE_ SAMPLE	VALUE_ SESSION	RATE_ SAMPLE	RATE_ SESSION	AVG_ SAMPLE	AVG_ SESSION
LONG					

SMC_NAME_MEM_PAGE_CACHE_SIZE

说明 报告为页高速缓存分配的内存字节数。

版本兼容性 11.0 和更高版本

数据项类型 结果

服务器级 有

必需的键 无

可选键 无

统计类型和数据类型

VALUE_ SAMPLE	VALUE_ SESSION	RATE_ SAMPLE	RATE_ SESSION	AVG_ SAMPLE	AVG_ SESSION
LONG					

SMC_NAME_MEM_PROC_BUFFER

说明 报告为过程缓冲区分配的内存字节数。

版本兼容性 11.0 和更高版本

数据项类型 结果

服务器级 有

必需的键 无

可选键 无

统计类型和数据类型

VALUE_ SAMPLE	VALUE_ SESSION	RATE_ SAMPLE	RATE_ SESSION	AVG_ SAMPLE	AVG_ SESSION
LONG					

SMC_NAME_MEM_PROC_HEADER

说明　报告为过程标题分配的内存字节数。

版本兼容性　11.0 和更高版本

数据项类型　结果

服务器级　有

必需的键　无

可选键　无

统计类型和数据类型

VALUE_ SAMPLE	VALUE_ SESSION	RATE_ SAMPLE	RATE_ SESSION	AVG_ SAMPLE	AVG_ SESSION
LONG					

SMC_NAME_MEM_SERVER_STRUCT_SIZE

说明　报告为 Adaptive Server 结构分配的内存字节数。

版本兼容性　11.0 和更高版本

数据项类型　结果

服务器级　有

必需的键　无

可选键　无

统计类型和数据类型

VALUE_ SAMPLE	VALUE_ SESSION	RATE_ SAMPLE	RATE_ SESSION	AVG_ SAMPLE	AVG_ SESSION
LONG					

SMC_NAME_MOST_ACT_DEV_IO

说明 报告在给定的时间段内，最频繁地读取和写入设备的组合数目。

版本兼容性 11.0 和更高版本

服务器级 有

数据项类型 结果

必需的键 无

可选键 无

统计类型和数据类型

VALUE_ SAMPLE	VALUE_ SESSION	RATE_ SAMPLE	RATE_ SESSION	AVG_ SAMPLE	AVG_ SESSION
LONG	LONG	DOUBLE	DOUBLE		

SMC_NAME_MOST_ACT_DEV_NAME

说明 报告在给定的时间段内，具有最大的读取和写入组合数目的设备名称。

版本兼容性 11.0 和更高版本

数据项类型 结果

服务器级 有

必需的键 无

可选键 无

统计类型和数据类型

VALUE_ SAMPLE	VALUE_ SESSION	RATE_ SAMPLE	RATE_ SESSION	AVG_ SAMPLE	AVG_ SESSION
CHARP	CHARP				

SMC_NAME_NET_BYTE_IO

说明　报告已发送和接收到的组合网络字节数。

版本兼容性　11.0 和更高版本

数据项类型　结果

服务器级　有

必需的键　无

可选键　无

统计类型和数据类型

VALUE_ SAMPLE	VALUE_ SESSION	RATE_ SAMPLE	RATE_ SESSION	AVG_ SAMPLE	AVG_ SESSION
LONG	LONG	DOUBLE	DOUBLE		

SMC_NAME_NET_BYTES_RCVD

说明　报告接收到的网络字节数。

版本兼容性　11.0 和更高版本

数据项类型　结果

服务器级　有

必需的键　无

可选键　无

统计类型和数据类型

VALUE_ SAMPLE	VALUE_ SESSION	RATE_ SAMPLE	RATE_ SESSION	AVG_ SAMPLE	AVG_ SESSION
LONG	LONG	DOUBLE	DOUBLE		

SMC_NAME_NET_BYTES_SENT

说明　报告发送出的网络字节数。

版本兼容性　11.0 和更高版本

数据项类型　结果

服务器级　有

必需的键　无

可选键　无

统计类型和数据类型

VALUE_ SAMPLE	VALUE_ SESSION	RATE_ SAMPLE	RATE_ SESSION	AVG_ SAMPLE	AVG_ SESSION
LONG	LONG	DOUBLE	DOUBLE		

SMC_NAME_NET_DEFAULT_PKT_SIZE

说明　报告网络包的缺省大小。

类型　结果

服务器级　有

必需的键　无

可选键　无

统计类型和数据类型

VALUE_ SAMPLE	VALUE_ SESSION	RATE_ SAMPLE	RATE_ SESSION	AVG_ SAMPLE	AVG_ SESSION
LONG					

SMC_NAME_NET_MAX_PKT_SIZE

说明　报告为网络包配置的最大空间。

版本兼容性　11.0 和更高版本

数据项类型　结果

服务器级　有

必需的键　无

可选键　无

统计类型和数据类型

VALUE_ SAMPLE	VALUE_ SESSION	RATE_ SAMPLE	RATE_ SESSION	AVG_ SAMPLE	AVG_ SESSION
LONG					

SMC_NAME_NET_PKT_SIZE_RCVD

说明　报告接收到的网络包的平均大小。

版本兼容性　11.0 和更高版本

数据项类型　结果

服务器级　有

必需的键　无

可选键　无

统计类型和数据类型

VALUE_ SAMPLE	VALUE_ SESSION	RATE_ SAMPLE	RATE_ SESSION	AVG_ SAMPLE	AVG_ SESSION
DOUBLE	DOUBLE				

SMC_NAME_NET_PKT_SIZE_SENT

说明 报告发送出的网络包的平均大小。

版本兼容性 11.0 和更高版本

数据项类型 结果

服务器级 有

必需的键 无

可选键 无

统计类型和数据类型

VALUE_ SAMPLE	VALUE_ SESSION	RATE_ SAMPLE	RATE_ SESSION	AVG_ SAMPLE	AVG_ SESSION
DOUBLE	DOUBLE				

SMC_NAME_NET_PKTS_RCVD

说明 报告接收到的网络包的数量。

版本兼容性 11.0 和更高版本

数据项类型 结果

服务器级 有

必需的键 无

可选键 无

统计类型和数据类型

VALUE_ SAMPLE	VALUE_ SESSION	RATE_ SAMPLE	RATE_ SESSION	AVG_ SAMPLE	AVG_ SESSION
LONG	LONG	DOUBLE	DOUBLE		

SMC_NAME_NET_PKTS_SENT

说明　报告发送出的网络包的数量。

版本兼容性　11.0 和更高版本

数据项类型　结果

服务器级　有

必需的键　无

可选键　无

统计类型和数据类型

VALUE_ SAMPLE	VALUE_ SESSION	RATE_ SAMPLE	RATE_ SESSION	AVG_ SAMPLE	AVG_ SESSION
LONG	LONG	DOUBLE	DOUBLE		

SMC_NAME_NUM_ENGINES

说明　报告在 Adaptive Server 上运行的引擎数。

版本兼容性　11.0 和更高版本

数据项类型　结果

服务器级　有

必需的键　无

可选键　无

统计类型和数据类型

VALUE_ SAMPLE	VALUE_ SESSION	RATE_ SAMPLE	RATE_ SESSION	AVG_ SAMPLE	AVG_ SESSION
LONG					

SMC_NAME_NUM_PROCESSES

说明 报告当前在 Adaptive Server 上运行的进程数，或如果使用键 SMC_NAME_APPLICATION_NAME，则报告当前运行给定应用程序的进程数。

版本兼容性 11.0 和更高版本

数据项类型 结果

服务器级 有

必需的键 无

可选键 SMC_NAME_APPLICATION_NAME

统计类型和数据类型

VALUE_ SAMPLE	VALUE_ SESSION	RATE_ SAMPLE	RATE_ SESSION	AVG_ SAMPLE	AVG_ SESSION
LONG					

SMC_NAME_OBJ_ID

说明 报告数据库对象的标识号，其中返回的对象是一个表或一个存储过程。

版本兼容性 11.0 和更高版本

数据项类型 键

服务器级 无

必需的键 SMC_NAME_DB_ID

需要此键的结果数据项

SMC_NAME_BLOCKING_SPID
SMC_NAME_DEMAND_LOCK
SMC_NAME_LOCKS_BEING_BLOCKED_CNT
SMC_NAME_OBJ_NAME
SMC_NAME_OBJ_TYPE
SMC_NAME_OWNER_NAME
SMC_NAME_TIME_WAITED_ON_LOCK

可选用此键的结果数据项

SMC_NAME_LOCKS_GRANTED_IMMED
SMC_NAME_LOCKS_GRANTED_WAITED
SMC_NAME_LOCKS_NOT_GRANTED
SMC_NAME_PAGE_INDEX_LOGICAL_READ
SMC_NAME_PAGE_INDEX_PHYSICAL_READ
SMC_NAME_PAGE_HIT_PCT
SMC_NAME_PAGE_IO
SMC_NAME_PAGE_LOGICAL_READ
SMC_NAME_PAGE_PHYSICAL_READ
SMC_NAME_PAGE_WRITE

统计类型和数据类型

VALUE_ SAMPLE	VALUE_ SESSION	RATE_ SAMPLE	RATE_ SESSION	AVG_ SAMPLE	AVG_ SESSION
LONG					

如果使用 SMC_NAME_OBJ_ID 数据项创建视图，您可能会发现对象 ID 为负数。负的对象 ID 如实反映了 Adaptive Server 分配的 ID。

Monitor Server 会报告*所有的*活动，包括 Adaptive Server 创建的用来执行复杂查询的临时表上的活动。Adaptive Server 分配给临时表的对象 ID 可为正，也可为负。 Adaptive Server 分配的对象 ID 是报告的内容。

SMC_NAME_OBJ_NAME

说明　报告数据库对象的名称。在显示 SMC_NAME_OBJ_NAME 的视图中，将用字符串 **TempObject** 表示临时表的报告结果。

版本兼容性　11.0 和更高版本

数据项类型　结果

服务器级　无

必需的键　SMC_NAME_DB_ID、 SMC_NAME_OBJ_ID

可选键　无

统计类型和数据类型

VALUE_ SAMPLE	VALUE_ SESSION	RATE_ SAMPLE	RATE_ SESSION	AVG_ SAMPLE	AVG_ SESSION
CHARP					

SMC_NAME_OBJ_TYPE

说明 报告数据库对象、表或存储过程的类型。

版本兼容性 11.0 和更高版本

数据项类型 结果

服务器级 无

必需的键 SMC_NAME_DB_ID、 SMC_NAME_OBJ_ID

可选键 无

统计类型和数据类型

VALUE_ SAMPLE	VALUE_ SESSION	RATE_ SAMPLE	RATE_ SESSION	AVG_ SAMPLE	AVG_ SESSION
ENUMS					

枚举 SMC_OBJ_TYPE

SMC_NAME_OWNER_NAME

说明 报告数据库对象的所有者名称。

版本兼容性 11.0 和更高版本

数据项类型 结果

服务器级 无

必需的键 SMC_NAME_DB_ID、 SMC_NAME_OBJ_ID

可选键 无

统计类型和数据类型

VALUE_ SAMPLE	VALUE_ SESSION	RATE_ SAMPLE	RATE_ SESSION	AVG_ SAMPLE	AVG_ SESSION
CHARP					

SMC_NAME_PAGE_HIT_PCT

说明　报告能够从高速缓存实现 （而不需要物理页读取）的数据页读取次数百分比。

版本兼容性　11.0 和更高版本

数据项类型　结果

服务器级　有

必需的键　无

可选键　SMC_NAME_SPID、
[SMC_NAME_DB_ID + SMC_NAME_OBJ_ID]、
[SMC_NAME_ACT_STP_DB_ID + SMC_NAME_ACT_STP_ID]、
SMC_NAME_ENGINE_NUM

统计类型和数据类型

VALUE_ SAMPLE	VALUE_ SESSION	RATE_ SAMPLE	RATE_ SESSION	AVG_ SAMPLE	AVG_ SESSION
DOUBLE	DOUBLE				

SMC_NAME_PAGE_INDEX_LOGICAL_READ

说明　报告从高速缓存或设备读取实现的索引页读取数。

版本兼容性　11.0 和更高版本

数据项类型　结果

服务器级　有

必需的键　无

可选键　SMC_NAME_SPID、 SMC_NAME_APPLICATION_NAME、
SMC_NAME_DB_ID、 SMC_NAME_OBJ_ID、
SMC_NAME_ENGINE_NUM、 [SMC_NAME_ACT_STP_DB_ID + SMC_NAME_ACT_STP_ID]

注释　SMC_NAME_SPID 和 SMC_NAME_APPLICATION_NAME 是互斥的。

统计类型和数据类型

VALUE_ SAMPLE	VALUE_ SESSION	RATE_ SAMPLE	RATE_ SESSION	AVG_ SAMPLE	AVG_ SESSION
LONG	LONG	DOUBLE	DOUBLE		

SMC_NAME_PAGE_INDEX_PHYSICAL_READ

说明 报告不能从数据高速缓存实现的索引页读取数。

版本兼容性 11.0 和更高版本

数据项类型 结果

服务器级 无

必需的键 无

可选键 SMC_NAME_SPID、 SMC_NAME_APPLICATION_NAME、 SMC_NAME_DB_ID、 SMC_NAME_OBJ_ID、 SMC_NAME_ENGINE_NUM、 [SMC_NAME_ACT_STP_DB_ID + SMC_NAME_ACT_STP_ID]

注释 SMC_NAME_SPID 和 SMC_NAME_APPLICATION_NAME 是互斥的。

统计类型和数据类型

VALUE_ SAMPLE	VALUE_ SESSION	RATE_ SAMPLE	RATE_ SESSION	AVG_ SAMPLE	AVG_ SESSION
LONG	LONG	DOUBLE	DOUBLE		

SMC_NAME_PAGE_IO

说明 报告组合逻辑页读取数和页写入数。

版本兼容性 11.0 和更高版本

数据项类型 结果

服务器级 有

必需的键 无

可选键 SMC_NAME_SPID、 SMC_NAME_APPLICATION_NAME、 [SMC_NAME_DB_ID + SMC_NAME_OBJ_ID]、 [SMC_NAME_ACT_STP_DB_ID + SMC_NAME_ACT_STP_ID]、 SMC_NAME_ENGINE_NUM

注释 SMC_NAME_SPID 和 SMC_NAME_APPLICATION_NAME 是互斥的。

统计类型和数据类型

VALUE_ SAMPLE	VALUE_ SESSION	RATE_ SAMPLE	RATE_ SESSION	AVG_ SAMPLE	AVG_ SESSION
LONG	LONG	DOUBLE	DOUBLE		

SMC_NAME_PAGE_LOGICAL_READ

说明　报告从高速缓存或数据库设备实现的数据页读取数。

版本兼容性　11.0 和更高版本

数据项类型　结果

服务器级　有

必需的键　无

可选键　SMC_NAME_SPID、 SMC_NAME_APPLICATION_NAME、 SMC_NAME_DB_ID、 SMC_NAME_OBJ_ID、 SMC_NAME_ENGINE_NUM、 [SMC_NAME_ACT_STP_DB_ID + SMC_NAME_ACT_STP_ID]

注释　SMC_NAME_SPID 和 SMC_NAME_APPLICATION_NAME 是互斥的。

统计类型和数据类型

VALUE_ SAMPLE	VALUE_ SESSION	RATE_ SAMPLE	RATE_ SESSION	AVG_ SAMPLE	AVG_ SESSION
LONG	LONG	DOUBLE	DOUBLE		

SMC_NAME_PAGE_NUM

说明 报告给定锁或锁请求的数据页数。

版本兼容性 11.0 和更高版本

数据项类型 键

服务器级 无

需要此键的结果数据项 无

可选用此键的结果数据项

SMC_NAME_BLOCKING_SPID
SMC_NAME_DEMAND_LOCK
SMC_NAME_LOCKS_BEING_BLOCKED_CNT
SMC_NAME_TIME_WAITED_ON_LOCK

统计类型和数据类型

VALUE_ SAMPLE	VALUE_ SESSION	RATE_ SAMPLE	RATE_ SESSION	AVG_ SAMPLE	AVG_ SESSION
LONG					

SMC_NAME_PAGE_PHYSICAL_READ

说明 报告不能从数据高速缓存实现的数据页读取数。

版本兼容性 11.0 和更高版本

数据项类型 结果

服务器级 有

必需的键 无

可选键 SMC_NAME_SPID、 SMC_NAME_APPLICATION_NAME、 SMC_NAME_DB_ID、 SMC_NAME_OBJ_ID、 SMC_NAME_ENGINE_NUM、 [SMC_NAME_ACT_STP_DB_ID + SMC_NAME_ACT_STP_ID]

注释 SMC_NAME_SPID 和 SMC_NAME_APPLICATION_NAME 是互斥的。

统计类型和数据类型

VALUE_ SAMPLE	VALUE_ SESSION	RATE_ SAMPLE	RATE_ SESSION	AVG_ SAMPLE	AVG_ SESSION
LONG	LONG	DOUBLE	DOUBLE		

SMC_NAME_PAGE_WRITE

说明　报告向某个数据库设备写入的数据页数。

版本兼容性　11.0 和更高版本

数据项类型　结果

服务器级　有

必需的键　无

可选键　SMC_NAME_SPID、 SMC_NAME_APPLICATION_NAME、 SMC_NAME_DB_ID、 SMC_NAME_OBJ_ID、 SMC_NAME_ENGINE_NUM、 [SMC_NAME_ACT_STP_DB_ID + SMC_NAME_ACT_STP_ID]

注释　SMC_NAME_SPID 和 SMC_NAME_APPLICATION_NAME 是互斥的。

统计类型和数据类型

VALUE_ SAMPLE	VALUE_ SESSION	RATE_ SAMPLE	RATE_ SESSION	AVG_ SAMPLE	AVG_ SESSION
LONG	LONG	DOUBLE	DOUBLE		

SMC_NAME_PROC_STATE

说明　报告进程的状态。可能的状态有：

- 无
- 报警休眠
- 后台
- 无效状态
- 已感染
- 锁休眠
- 接收休眠
- 远程 IO
- 可运行的
- 运行

- 发送休眠
- 正在休眠
- 已停止
- 同步休眠
- 正在终止
- 正在放弃

版本兼容性 11.0 和更高版本

数据项类型 键

服务器级 无

需要此键的结果数据项 SMC_NAME_PROC_STATE_CNT

可选用此键的结果数据项 无

统计类型和数据类型

VALUE_ SAMPLE	VALUE_ SESSION	RATE_ SAMPLE	RATE_ SESSION	AVG_ SAMPLE	AVG_ SESSION
ENUMS					

枚举 SMC_PROC_STATE

SMC_NAME_PROC_STATE_CNT

说明 报告处于特定状态下的进程数。

版本兼容性 11.0 和更高版本

数据项类型 结果

服务器级 无

必需的键 SMC_NAME_PROC_STATE

可选键 无

统计类型和数据类型

VALUE_ SAMPLE	VALUE_ SESSION	RATE_ SAMPLE	RATE_ SESSION	AVG_ SAMPLE	AVG_ SESSION
LONG					

SMC_NAME_SPID

说明

报告进程标识号。包含 SMC_NAME_SPID 的视图仅报告截止到采样周期结束时活动的进程。在一个视图中， SMC_NAME_SPID 和 SMC_NAME_APPLICATION_NAME 互斥。

版本兼容性

11.0 和更高版本

数据项类型

键

服务器级

无

需要此键的结果数据项

SMC_NAME_BLOCKING_SPID
SMC_NAME_CONNECT_TIME
SMC_NAME_CPU_PCT
SMC_NAME_CPU_TIME
SMC_NAME_CUR_APP_NAME
SMC_NAME_CUR_ENGINE
SMC_NAME_CUR_EXECUTION_CLASS
SMC_NAME_CUR_PROC_STATE
SMC_NAME_CUR_STMT_ACT_STP_DB_NAME
SMC_NAME_CUR_STMT_ACT_STP_NAME
SMC_NAME_CUR_STMT_ACT_STP_OWNER_NAME
SMC_NAME_CUR_STMT_ACT_STP_TEXT
SMC_NAME_CUR_STMT_BATCH_TEXT
SMC_NAME_CUR_STMT_CPU_TIME
SMC_NAME_CUR_STMT_ELAPSED_TIME
SMC_NAME_CUR_STMT_LINE_NUM
SMC_NAME_CUR_STMT_LOCKS_GRANTED_IMMED
SMC_NAME_CUR_STMT_LOCKS_GRANTED_WAITED
SMC_NAME_CUR_STMT_LOCKS_NOT_GRANTED
SMC_NAME_CUR_STMT_PAGE_IO_CNT
SMC_NAME_CUR_STMT_PAGE_CACHE_READ_CNT
SMC_NAME_CUR_STMT_PAGE_PHYSICAL_READ_CNT
SMC_NAME_CUR_STMT_PAGE_WRITE_CNT
SMC_NAME_CUR_STMT_QUERY_PLAN_TEXT
SMC_NAME_CUR_STMT_START_TIME
SMC_NAME_CUR_STMT_TEXT_BYTE_OFFSET
SMC_NAME_DEMAND_LOCK
SMC_NAME_HOST_NAME
SMC_NAME_KPID

SMC_NAME_LOCKS_BEING_BLOCKED_CNT
SMC_NAME_LOGIN_NAME
SMC_NAME_TIME_WAITED_ON_LOCK

可选用此键的结果数据项

SMC_NAME_LOCK_CNT
SMC_NAME_LOCKS_GRANTED_IMMED
SMC_NAME_LOCKS_GRANTED_WAITED
SMC_NAME_LOCKS_NOT_GRANTED
SMC_NAME_PAGE_INDEX_LOGICAL_READ
SMC_NAME_PAGE_INDEX_PHYSICAL_READ
SMC_NAME_PAGE_LOGICAL_READ
SMC_NAME_PAGE_PHYSICAL_READ
SMC_NAME_PAGE_WRITE
SMC_NAME_STP_CPU_TIME
SMC_NAME_STP_NUM_TIMES_EXECUTED

统计类型和数据类型

VALUE_ SAMPLE	VALUE_ SESSION	RATE_ SAMPLE	RATE_ SESSION	AVG_ SAMPLE	AVG_ SESSION
LONG					

SMC_NAME_SQL_SERVER_NAME

说明 报告正在被监控的 Adaptive Server 的名称，该服务器由应用程序连接的 Monitor Server 的启动命令中的 -s 参数指定。

版本兼容性 11.0 和更高版本

数据项类型 结果

服务器级 有

必需的键 无

可选键 无

统计类型和数据类型

VALUE_ SAMPLE	VALUE_ SESSION	RATE_ SAMPLE	RATE_ SESSION	AVG_ SAMPLE	AVG_ SESSION
CHARP					

SMC_NAME_SQL_SERVER_VERSION

说明　报告被监控的 Adaptive Server 的版本。有关详细信息，请参见《Transact-SQL 用户指南》中的 *@@version* 全局变量。

版本兼容性　11.0 和更高版本

数据项类型　结果

服务器级　有

必需的键　无

可选键　无

统计类型和数据类型

VALUE_ SAMPLE	VALUE_ SESSION	RATE_ SAMPLE	RATE_ SESSION	AVG_ SAMPLE	AVG_ SESSION
CHARP					

SMC_NAME_STP_CPU_TIME

说明　报告执行一个存储过程花费的 CPU 时间 （以秒计）。

版本兼容性　11.0 和更高版本

数据项类型　结果

服务器级　无

必需的键　SMC_NAME_ACT_STP_DB_ID 和 SMC_NAME_ACT_STP_ID

可选键　SMC_NAME_SPID、 SMC_NAME_STP_STMT_NUM 和 SMC_NAME_STP_LINE_NUM

统计类型和数据类型

VALUE_ SAMPLE	VALUE_ SESSION	RATE_ SAMPLE	RATE_ SESSION	AVG_ SAMPLE	AVG_ SESSION
DOUBLE	DOUBLE			DOUBLE	DOUBLE

SMC_NAME_STP_ELAPSED_TIME

说明 报告执行一个存储过程使用的时间 （以秒计）。

版本兼容性 11.0 和更高版本

数据项类型 结果

服务器级 无

必需的键 SMC_NAME_ACT_STP_DB_ID 和 SMC_NAME_ACT_STP_ID

可选键 SMC_NAME_STP_STMT_NUM 和 SMC_NAME_STP_LINE_NUM

统计类型和数据类型

VALUE_ SAMPLE	VALUE_ SESSION	RATE_ SAMPLE	RATE_ SESSION	AVG_ SAMPLE	AVG_ SESSION
DOUBLE	DOUBLE			DOUBLE	DOUBLE

SMC_NAME_STP_EXECUTION_CLASS

说明 报告给定存储过程的已配置执行类 （如果有的话）。

版本兼容性 11.5 和更高版本

数据项类型 结果

服务器级 无

必需的键 SMC_NAME_ACT_STP_DB_ID 和 SMC_NAME_ACT_STP_ID

可选键 SMC_NAME_STP_STMT_NUM 和 SMC_NAME_STP_LINE_NUM

统计类型和数据类型

VALUE_ SAMPLE	VALUE_ SESSION	RATE_ SAMPLE	RATE_ SESSION	AVG_ SAMPLE	AVG_ SESSION
CHARP					

SMC_NAME_STP_HIT_PCT

说明　报告一个存储过程执行时，在该过程的高速缓存中找到该过程的查询计划且该计划可用的次数所占的百分比。

版本兼容性　11.0 和更高版本

数据项类型　结果

服务器级　有

必需的键　无

可选键　无

统计类型和数据类型

VALUE_ SAMPLE	VALUE_ SESSION	RATE_ SAMPLE	RATE_ SESSION	AVG_ SAMPLE	AVG_ SESSION
DOUBLE	DOUBLE				

SMC_NAME_STP_LINE_NUM

说明　报告存储过程行号。

版本兼容性　11.0 和更高版本

数据项类型　键

服务器级　无

需要此键的结果数据项　无

可选用此键的结果数据项

SMC_NAME_STP_CPU_TIME
SMC_NAME_STP_ELAPSED_TIME
SMC_NAME_STP_NUM_TIMES_EXECUTED

统计类型和数据类型

VALUE_ SAMPLE	VALUE_ SESSION	RATE_ SAMPLE	RATE_ SESSION	AVG_ SAMPLE	AVG_ SESSION
LONG					

SMC_NAME_STP_LINE_TEXT

说明 报告存储过程的全部文本。

版本兼容性 11.0 和更高版本

数据项类型 结果

服务器级 无

必需的键 SMC_NAME_ACT_STP_DB_ID 和 SMC_NAME_ACT_STP_ID

可选键 无

统计类型和数据类型

VALUE_ SAMPLE	VALUE_ SESSION	RATE_ SAMPLE	RATE_ SESSION	AVG_ SAMPLE	AVG_ SESSION
CHARP					

SMC_NAME_STP_LOGICAL_READ

说明 报告执行一个存储过程的请求数目，包括从过程高速缓存实现或从 *sysprocedures* 读取。

版本兼容性 11.0 和更高版本

数据项类型 结果

服务器级 有

必需的键 无

可选键 无

统计类型和数据类型

VALUE_ SAMPLE	VALUE_ SESSION	RATE_ SAMPLE	RATE_ SESSION	AVG_ SAMPLE	AVG_ SESSION
LONG	LONG	DOUBLE	DOUBLE		

SMC_NAME_STP_NUM_TIMES_EXECUTED

说明　报告执行一个存储过程或执行其中某一行的次数。

版本兼容性　11.0 和更高版本

数据项类型　结果

服务器级　无

必需的键　SMC_NAME_ACT_STP_DB_ID 和 SMC_NAME_ACT_STP_ID

可选键　SMC_NAME_SPID、 SMC_NAME_STP_STMT_NUM 和 SMC_NAME_STP_LINE_NUM

统计类型和数据类型

VALUE_ SAMPLE	VALUE_ SESSION	RATE_ SAMPLE	RATE_ SESSION	AVG_ SAMPLE	AVG_ SESSION
LONG	LONG	DOUBLE	DOUBLE		

SMC_NAME_STP_PHYSICAL_READ

说明　报告执行一个存储过程的请求数目，存储过程必须有从 *sysprocedures* 的读取。

版本兼容性　11.0 和更高版本

数据项类型　结果

服务器级　有

必需的键　无

可选键　无

统计类型和数据类型

VALUE_ SAMPLE	VALUE_ SESSION	RATE_ SAMPLE	RATE_ SESSION	AVG_ SAMPLE	AVG_ SESSION
LONG	LONG	DOUBLE	DOUBLE		

SMC_NAME_STP_STMT_NUM

说明 报告一个存储过程内的编号。一个存储过程行可以包含一条或多条语句。

版本兼容性 11.0 和更高版本

数据项类型 键

服务器级 无

需要此键的结果数据项 无

可选用此键的结果数据项

SMC_NAME_STP_CPU_TIME
SMC_NAME_STP_ELAPSED_TIME
SMC_NAME_STP_NUM_TIMES_EXECUTED

统计类型和数据类型

VALUE_ SAMPLE	VALUE_ SESSION	RATE_ SAMPLE	RATE_ SESSION	AVG_ SAMPLE	AVG_ SESSION
LONG					

SMC_NAME_THREAD_EXCEEDED_MAX

说明 报告因为试图超过 Adaptive Server 中配置的服务器范围内的工作线程池的线程数限值，从而在运行时调整查询计划的次数。

版本兼容性 11.5 和更高版本

数据项类型 结果

服务器级 有

必需的键 无

可选键 无

统计类型和数据类型

VALUE_ SAMPLE	VALUE_ SESSION	RATE_ SAMPLE	RATE_ SESSION	AVG_ SAMPLE	AVG_ SESSION
LONG	LONG	DOUBLE	DOUBLE		

SMC_NAME_THREAD_EXCEEDED_MAX_PCT

说明　报告因为试图超过 Adaptive Server 中配置的服务器范围内的工作线程池的线程数限值，从而在运行时调整查询计划的时间百分比。

版本兼容性　11.5 和更高版本

数据项类型　结果

服务器级　有

必需的键　无

可选键　无

统计类型和数据类型

VALUE_ SAMPLE	VALUE_ SESSION	RATE_ SAMPLE	RATE_ SESSION	AVG_ SAMPLE	AVG_ SESSION
DOUBLE	DOUBLE				

SMC_NAME_THREAD_MAX_USED

说明　报告服务器端工作线程池中的线程，在服务器上并发使用的最大数目。

版本兼容性　11.5 和更高版本

数据项类型　结果

服务器级　有

必需的键　无

可选键　无

统计类型和数据类型

VALUE_ SAMPLE	VALUE_ SESSION	RATE_ SAMPLE	RATE_ SESSION	AVG_ SAMPLE	AVG_ SESSION
LONG					

SMC_NAME_TIME_WAITED_ON_LOCK

说明 报告等待获得锁请求授权的时间量 （以秒计）。

版本兼容性 11.0 和更高版本

数据项类型 结果

服务器级 无

必需的键 SMC_NAME_SPID、 SMC_NAME_DB_ID、 SMC_NAME_OBJ_ID、 SMC_NAME_LOCK_STATUS

可选键 SMC_NAME_LOCK_TYPE、 SMC_NAME_PAGE_NUM

统计类型和数据类型

VALUE_ SAMPLE	VALUE_ SESSION	RATE_ SAMPLE	RATE_ SESSION	AVG_ SAMPLE	AVG_ SESSION
LONG					

SMC_NAME_TIMESTAMP

说明 报告 Adaptive Server 在其时区内的日期和时间。有关详细信息，请参见《Transact-SQL 用户指南》中的 getdate() 函数。

版本兼容性 11.0 和更高版本

数据项类型 结果

服务器级 有

必需的键 无

可选键 无

统计类型和数据类型

VALUE_ SAMPLE	VALUE_ SESSION	RATE_ SAMPLE	RATE_ SESSION	AVG_ SAMPLE	AVG_ SESSION
CHARP					

SMC_NAME_TIMESTAMP_DATIM

说明　报告 Adaptive Server 在其时区内的日期和时间，并以 CS_DATETIME 结构返回。有关详细信息，请参见 《Transact-SQL 用户指南》中的 getdate() 函数。

版本兼容性　11.0 和更高版本

数据项类型　结果

服务器级　有

必需的键　无

可选键　无

统计类型和数据类型

VALUE_ SAMPLE	VALUE_ SESSION	RATE_ SAMPLE	RATE_ SESSION	AVG_ SAMPLE	AVG_ SESSION
DATIM					

SMC_NAME_XACT

说明　报告已提交的 Transact-SQL 语句块 （事务）的数量。

版本兼容性　11.0 和更高版本

数据项类型　结果

服务器级　有

必需的键　无

可选键　无

统计类型和数据类型

VALUE_ SAMPLE	VALUE_ SESSION	RATE_ SAMPLE	RATE_ SESSION	AVG_ SAMPLE	AVG_ SESSION
LONG	LONG	DOUBLE	DOUBLE		

SMC_NAME_XACT_DELETE

说明 报告从数据库表中删除的行数。

版本兼容性 11.0 和更高版本

数据项类型 结果

服务器级 有

必需的键 无

可选键 无

统计类型和数据类型

VALUE_ SAMPLE	VALUE_ SESSION	RATE_ SAMPLE	RATE_ SESSION	AVG_ SAMPLE	AVG_ SESSION
LONG	LONG	DOUBLE	DOUBLE		

SMC_NAME_XACT_DELETE_DEFERRED

说明 报告从以延迟模式完成的数据库表中删除的行数。

版本兼容性 11.0 和更高版本

数据项类型 结果

服务器级 有

必需的键 无

可选键 无

统计类型和数据类型

VALUE_ SAMPLE	VALUE_ SESSION	RATE_ SAMPLE	RATE_ SESSION	AVG_ SAMPLE	AVG_ SESSION
LONG	LONG	DOUBLE	DOUBLE		

SMC_NAME_XACT_DELETE_DIRECT

说明　报告从以直接模式完成的数据库表中删除的行数。

版本兼容性　11.0 和更高版本

数据项类型　结果

服务器级　有

必需的键　无

可选键　无

统计类型和数据类型

VALUE_ SAMPLE	VALUE_ SESSION	RATE_ SAMPLE	RATE_ SESSION	AVG_ SAMPLE	AVG_ SESSION
LONG	LONG	DOUBLE	DOUBLE		

SMC_NAME_XACT_INSERT

说明　报告插入到数据库表中的数据量。

版本兼容性　11.0 和更高版本

数据项类型　结果

服务器级　有

必需的键　无

可选键　无

统计类型和数据类型

VALUE_ SAMPLE	VALUE_ SESSION	RATE_ SAMPLE	RATE_ SESSION	AVG_ SAMPLE	AVG_ SESSION
LONG	LONG	DOUBLE	DOUBLE		

SMC_NAME_XACT_INSERT_CLUSTERED

说明 报告插入到拥有聚簇索引的数据库表中的数据量。

版本兼容性 11.0 和更高版本

数据项类型 结果

服务器级 有

必需的键 无

可选键 无

统计类型和数据类型

VALUE_ SAMPLE	VALUE_ SESSION	RATE_ SAMPLE	RATE_ SESSION	AVG_ SAMPLE	AVG_ SESSION
LONG	LONG	DOUBLE	DOUBLE		

SMC_NAME_XACT_INSERT_HEAP

说明 报告插入到不具有聚簇索引的数据库表中的数据量。

版本兼容性 11.0 和更高版本

数据项类型 结果

服务器级 有

必需的键 无

可选键 无

统计类型和数据类型

VALUE_ SAMPLE	VALUE_ SESSION	RATE_ SAMPLE	RATE_ SESSION	AVG_ SAMPLE	AVG_ SESSION
LONG	LONG	DOUBLE	DOUBLE		

SMC_NAME_XACT_SELECT

说明　报告 SELECT 或 OPEN CURSOR 语句的数量。

版本兼容性　11.0 和更高版本

数据项类型　结果

服务器级　有

必需的键　无

可选键　无

统计类型和数据类型

VALUE_ SAMPLE	VALUE_ SESSION	RATE_ SAMPLE	RATE_ SESSION	AVG_ SAMPLE	AVG_ SESSION
LONG	LONG	DOUBLE	DOUBLE		

SMC_NAME_XACT_UPDATE

说明　报告数据库表的更新。

版本兼容性　11.0 和更高版本

数据项类型　结果

服务器级　有

必需的键　无

可选键　无

统计类型和数据类型

VALUE_ SAMPLE	VALUE_ SESSION	RATE_ SAMPLE	RATE_ SESSION	AVG_ SAMPLE	AVG_ SESSION
LONG	LONG	DOUBLE	DOUBLE		

SMC_NAME_XACT_UPDATE_DEFERRED

说明 报告以延迟模式而不是直接模式执行的数据库表的更新。

版本兼容性 11.0 和更高版本

数据项类型 结果

服务器级 有

必需的键 无

可选键 无

统计类型和数据类型

VALUE_ SAMPLE	VALUE_ SESSION	RATE_ SAMPLE	RATE_ SESSION	AVG_ SAMPLE	AVG_ SESSION
LONG	LONG	DOUBLE	DOUBLE		

SMC_NAME_XACT_UPDATE_DIRECT

说明 报告昂贵的、现场的和非现场的更新数目的总和 （除延迟更新数外的一切更新）。也称做现场更新。

版本兼容性 11.0 和更高版本

数据项类型 结果

服务器级 有

必需的键 无

可选键 无

统计类型和数据类型

VALUE_ SAMPLE	VALUE_ SESSION	RATE_ SAMPLE	RATE_ SESSION	AVG_ SAMPLE	AVG_ SESSION
LONG	LONG	DOUBLE	DOUBLE		

SMC_NAME_XACT_UPDATE_EXPENSIVE

说明　报告以昂贵模式完成的数据库表的更新。在昂贵模式中，将一行从其原始位置删除并在一个新位置插入。

版本兼容性　11.0 和更高版本

数据项类型　结果

服务器级　有

必需的键　无

可选键　无

统计类型和数据类型

VALUE_ SAMPLE	VALUE_ SESSION	RATE_ SAMPLE	RATE_ SESSION	AVG_ SAMPLE	AVG_ SESSION
LONG	LONG	DOUBLE	DOUBLE		

SMC_NAME_XACT_UPDATE_IN_PLACE

说明　报告不需要删除和插入的更新。

版本兼容性　11.0 和更高版本

数据项类型　结果

服务器级　有

必需的键　无

可选键　无

统计类型和数据类型

VALUE_ SAMPLE	VALUE_ SESSION	RATE_ SAMPLE	RATE_ SESSION	AVG_ SAMPLE	AVG_ SESSION
LONG	LONG	DOUBLE	DOUBLE		

SMC_NAME_XACT_UPDATE_NOT_IN_PLACE

说明 报告需要删除和插入的更新。

版本兼容性 11.0 和更高版本

数据项类型 结果

服务器级 有

必需的键 无

可选键 无

统计类型和数据类型

VALUE_ SAMPLE	VALUE_ SESSION	RATE_ SAMPLE	RATE_ SESSION	AVG_ SAMPLE	AVG_ SESSION
LONG	LONG	DOUBLE	DOUBLE		

第 3 章

Monitor Client Library 函数

本章包含有关 Monitor Client Library 函数的信息。

库函数

使用 Monitor Client Library 函数可以编写收集 Adaptive Server 性能数据的应用程序。本章按字母顺序介绍每个 Monitor Client Library 函数。表 3-1 列出了每个函数及其简短描述。

表 3-1：Monitor Client Library 函数

函数	说明
smc_close	关闭连接
smc_connect_alloc	创建连接结构
smc_connect_drop	释放连接结构
smc_connect_ex	建立连接
smc_connect_props	设置、检索或清除连接的属性
smc_create_alarm_ex	为数据项添加报警
smc_create_filter	为数据项添加过滤器
smc_create_playback_session	在 Historical Server 连接上初始化回放会话
smc_create_recording_session	在 Historical Server 连接上初始化记录会话
smc_create_view	定义视图
smc_drop_alarm	从视图中的数据项删除一个报警
smc_drop_filter	从视图中的数据项删除一个过滤器
smc_drop_view	删除视图
smc_get_command_info	检索关于报警或错误的详细信息
smc_get_dataitem_type	检索数据项的类型
smc_get_dataitem_value	检索特定数据项和行的数据

函数	说明
smc_get_row_count	检索视图中数据的行数
smc_get_version_string	检索 Monitor Client Library 版本号
smc_initiate_playback	结束回放会话的视图定义
smc_initiate_recording	结束记录会话的视图定义
smc_refresh_ex	检索给定连接中所有视图的数据
smc_terminate_playback	结束 Historical Server 连接上的回放会话
smc_terminate_recording	取消 Historical Server 连接上的记录会话

大多数函数都适用于 Monitor Server 和 Historical Server。在本章中，除非另有说明，术语 “连接”表示对 Monitor Server 或 Historical Server 的一个连接。有关旧函数的信息，请参见附录 C “向后兼容性”。

线程

两个线程不能同时使用 Monitor Client Library 函数。在 Monitor Client Library 调用上使用全局锁 （信号）以避免任何线程覆盖或不可预测的操作。

Monitor Client Library 函数不进行重入调用保护。当在多线程环境中使用这些函数时，可利用下列特殊编程考虑事项。确保：

- 创建客户端连接的调用 (smc_connect) 和所有线程上的所有其它 Monitor Client Library 函数调用是顺序执行的。
- 断开客户端连接的调用 (smc_disconnect) 和所有线程上的所有其它 Monitor Client Library 函数调用是顺序执行的。
- 任何单个客户端连接仅存在于一个线程上。访问此客户端连接的所有 Monitor Client Library 函数调用发生于该线程中。
- 用于刷新客户端连接的调用和该线程中此连接上的所有其它 Monitor Client Library 函数调用是顺序执行的。

错误处理

Monitor Client Library 应用程序在创建连接时会安装一个错误处理程序。任何时候该连接出现错误将调用此错误处理程序。

大多数 Monitor Client Library 函数返回下列值之一：

表 3-2：返回值

返回值	说明
SMC_RET_SUCCESS	函数已成功完成。
SMC_RET_FAILURE	函数失败。更详细的信息可从错误处理程序获得。
SMC_RET_INVALID_CONNECT	函数没有执行，因为针对错误的连接请求该函数。未调用错误处理程序，因为错误处理程序仅对有效连接起作用。

其它返回值与返回它们的函数一起列出。

注释　在关于 smc_create_view 和 smc_create_alarm 中的数据项指定的某些错误条件下，不触发错误回调函数。要捕获这些错误条件，可检查这些函数的返回码。

错误处理程序

说明

错误处理程序是一个用户定义的函数。

语法

```
SMC_VOID ErrorCallback (
 SMC_CONNECT_ID clientId,
 SMC_COMMAND_ID commandId,
 SMC_VOIDP userDataHandle)
```

参数

clientId

标识监控器连接。

commandId

标识一个命令的实例。

userDataHandle

用户提供的指针。

用法

- 可随时使用 smc_change_error_handler 或 smc_connect_props 函数更改错误处理程序。有关详细信息，请参见第 132 页的 “回调函数”。

注释　C++ 成员函数不能用作回调函数。

回调函数

说明

回调函数是用户定义的函数，用于在发生事件时通知应用程序。这些函数是用 Monitor Client Library API 调用注册的，用于：

- 报警
- 错误信息

当发生上述事件之一时，将执行一个回调函数。

语法

```
SMC_VOID CallbackFunction
 (SMC_CONNECT_ID clientId,
 SMC_COMMAND_ID commandId,
 SMC_VOIDP userDataHandle)
```

参数

clientId
标识连接。

commandId
标识一个命令的实例。

userDataHandle
给定连接的用户数据指针。应用程序可使用 smc_connect_props 设置该指针。

用法

访问回调数据

当一个事件触发一个回调函数时，可请求关于该事件的信息。可通过从回调函数内部调用 smc_get_command_info 来访问数据。此函数使用一个连接 ID、一个命令 ID 和一个枚举常量 （用来标识用户感兴趣的数据片段）。可获得的数据取决于回调类型。表 3-3 描述了报警回调可获得的数据。表 3-4 描述了错误回调可获得的数据。

表 3-3：报警回调可获得的数据

信息类型	说明
SMC_INFO_ALARM_ACTION_DATA	创建报警时为 *alarmActionData* 提供的字符串。
SMC_INFO_ALARM_ALARMID	标识报警。
SMC_INFO_ALARM_CURRENT_VALUE	达到或超出报警阈值的当前值。
SMC_INFO_ALARM_DATAITEM	在其上设置报警的数据项。指向一个 SMC_DATAITEM_STRUCT。
SMC_INFO_ALARM_ROW	包含触发报警的数据项值的行。
SMC_INFO_ALARM_THRESHOLD_VALUE	为此报警定义的阈值。
SMC_INFO_ALARM_TIMESTAMP	标记采样间隔 （其数据满足报警条件）结束的时间 （在 Adaptive Server 时区中）。
SMC_INFO_ALARM_VIEWID	标识针对连接创建的视图。

表 3-4：错误回调可获得的数据

信息类型	说明
SMC_INFO_ERR_MAPSEVERITY	Monitor Client Library 严重级。
SMC_INFO_ERR_MSG	错误消息的文本。（请参见附录 D “疑难解答信息和错误消息”。）
SMC_INFO_ERR_NUM	错误的编号。
SMC_INFO_ERR_SEVERITY	错误消息的严重性。
SMC_INFO_ERR_SOURCE	错误消息的来源。下列值之一： • SMC_SRC_UNKNOWN — 未知 • SMC_SRC_HS — Historical Server • SMC_SRC_SMC — Monitor Client Library • SMC_SRC_CT — Client Library • SMC_SRC_SS — Adaptive Server • SMC_SRC_SMS — Monitor Server
SMC_INFO_ERR_STATE	错误的状态。对于技术支持诊断内部错误很有用。

smc_close

说明

关闭用 smc_connect_ex 创建的连接。此函数终止连接但不释放它。请使用 smc_connect_drop 释放连接结构。

语法

```
SMC_RETURN_CODE smc_close
 (SMC_CONNECT_ID      clientId,
 SMC_CLOSE_TYPE       closeType)
```

参数

clientId

标识连接。

closeType

关闭的类型：SMC_CLOSE_REQUEST

返回值

返回值	含义
SMC_RET_SUCCESS	函数成功。
SMC_RET_FAILURE	函数失败。
SMC_RET_INVALID_CONNECT	不存在连接。

示例

下面的示例假定您已创建了一个连接且已有 *clientId*。

```
if (smc_close(clientId,SMC_CLOSE_REQUEST)
      != SMC_RET_SUCCESS)
 {
      printf("smc_close failed\n");
      /* do some cleanup */
 }
```

用法

- 还会删除指定连接上的所有视图 （以及与视图中的数据项关联的报警和过滤器）。
- smc_close 只断开连接。调用 smc_connect_drop 释放一个连接结构。
- 如果 smc_close 返回错误，建议用户调用 smc_connect_drop。

有效的服务器模式

模式	可用性
SMC_SERVER_M_LIVE	可用
SMC_SERVER_M_HISTORICAL	可用

错误

错误	含义
SMC_RET_INTERNAL_ERROR	内部错误
SMC_RET_INVALID_API_FUNCTION	在同一连接中无效地使用了旧函数和替换函数
SMC_RET_INVALID_API_FUNC_SEQUENCE	Monitor Client Library 函数调用顺序无效

另请参见

smc_connect_drop、 smc_connect_ex

smc_connect_alloc

说明

创建带有错误回调的连接结构，但不建立连接。

语法

```
SMC_RETURN_CODE smc_connect_alloc
 (SMC_GEN_CALLBACK    ErrCallback,
 SMC_CONNECT_IDP      clientIdHandle)
```

参数

ErrCallback

指向错误回调函数的指针。

clientIdHandle

指向一个变量的指针，变量应声明为 SMC_CONNECT_ID 类型。如果对 smc_connect 的调用成功，则此变量包含该 Monitor 连接的 ID。

返回值

返回值	含义
SMC_RET_SUCCESS	函数成功。
SMC_RET_FAILURE	函数失败。

示例

下例假定您已定义了一个错误回调函数 *myErrorHandler*。

```
SMC_CONNECT_ID  clientId;
 if (smc_connect_alloc(myErrorHandler,&clientId)
       != SMC_RET_SUCCESS)
 {
       printf("smc_connect_alloc failed\n");
       exit(1);
 }
```

用法

- 错误处理程序参数不能是空值。
- 使用 smc_connect_props 可设置连接属性。
- 使用 smc_connect_ex 可建立由 *clientIdHandle* 标识的连接。
- 使用 smc_connect_drop 可释放用 smc_connect_alloc 创建的连接结构。

有效的服务器模式

模式	可用性
SMC_SERVER_M_LIVE	可用
SMC_SERVER_M_HISTORICAL	可用

错误

错误	含义
SMC_RET_INSUFFICIENT_MEMORY	内存不足
SMC_RET_INTERNAL_ERROR	内部错误

另请参见

smc_connect_drop、 smc_connect_ex、 smc_connect_props

smc_connect_drop

说明

释放用 smc_connect_alloc 创建的连接结构。

语法

SMC_RETURN_CODE smc_connect_drop
(SMC_CONNECT_ID *clientId*)

参数

clientId

标识连接。

返回值

返回值	含义
SMC_RET_SUCCESS	函数成功。
SMC_RET_FAILURE	函数失败。
SMC_RET_INVALID_CONNECT	不存在连接。

示例

下面的示例假定：

- 您已使用 smc_connect_alloc 创建一个连接且已有一个 *clientId*。
  您已成功地对该连接执行 smc_close。

```
if (smc_connect_drop(clientId) != SMC_RET_SUCCESS) {
    printf("smc_connect_drop failed\n");
    /* do some cleanup */
 }
```

用法

- 如果已成功建立一个连接，必须在调用 smc_connect_drop 之前调用 smc_close。

有效的服务器模式

模式	可用性
SMC_SERVER_M_LIVE	可用
SMC_SERVER_M_HISTORICAL	可用

错误

错误	含义
SMC_RET_CONNECT_NOT_CLOSED	连接尚未关闭
SMC_RET_INVALID_API_FUNCTION	在同一连接中无效地使用了旧函数和替换函数
SMC_RET_INVALID_API_FUNC_SEQUENCE	Monitor Client Library 函数调用顺序无效

另请参见

smc_close、 smc_connect_alloc

smc_connect_ex

说明

为用 smc_connect_alloc 创建的连接结构建立一个连接。必须已使用 smc_connect_props 设置了该连接的属性 （如服务器名和服务器模式）。

语法

SMC_RETURN_CODE smc_connect_ex
(SMC_CONNECT_ID *clientId*)

参数

clientId
标识连接。

返回值

返回值	含义
SMC_RET_SUCCESS	函数成功。
SMC_RET_FAILURE	函数失败。
SMC_RET_INVALID_CONNECT	不存在连接。

示例

下面的示例假定您已使用 smc_connect_alloc 创建了一个连接且已有一个 *clientId*。

```
if (smc_connect_ex(clientId) != SMC_RET_SUCCESS)
 {
       printf("smc_connect_ex failed\n");
       exit(1);
 }
```

用法

- 必须先调用 smc_connect_alloc 和 smc_connect_props，然后再调用 smc_connect_ex。
- 每个 Monitor Client Library 连接使用两个网络连接。如果在 PC 上运行 Monitor Client Library 应用程序，并达到网络连接的限值，可重配置网络软件以增加该限值。

有效的服务器模式

模式	可用性
SMC_SERVER_M_LIVE	可用
SMC_SERVER_M_HISTORICAL	可用

错误

错误	含义
SMC_RET_INSUFFICIENT_MEMORY	内存不足
SMC_RET_INTERNAL_ERROR	内部错误
SMC_RET_INVALID_API_FUNCTION	在同一连接中无效地使用了旧函数和替换函数
SMC_RET_INVALID_API_FUNC_SEQUENCE	Monitor Client Library 函数调用顺序无效
SMC_RET_INVALID_PROPERTY	未设置属性
SMC_RET_UNABLE_TO_CONNECT_TO_SMS	无法连接到 Monitor Server
SMC_RET_UNABLE_TO_CONNECT_TO_SS	无法连接到 Adaptive Server

另请参见 smc_close、 smc_connect_alloc

smc_connect_props

说明 设置、检索或清除连接的属性。

语法

```
SMC_RETURN_CODE smc_connect_props
 (SMC_CONNECT_ID      clientId,
SMC_PROP_ACTION      propertyAction,
SMC_PROP_TYPE        property,
SMC_VALUE_UNIONP     propertyValue,
SMC_SIZET            bufferLength,
SMC_SIZETP           outputLengthHandle)
```

参数

clientId

标识连接。

propertyAction

属性操作类型。有效类型如下：

- SMC_PROP_ACT_CLEAR — 将指定属性的值重置为其缺省值。
- SMC_PROP_ACT_GET — 检索指定属性的值。
- SMC_PROP_ACT_SET — 设置指定属性的值。

property

属性的符号名，要设置、检索或清除的就是该属性的值。有关此参数的合法值的列表，请参见第 141 页的表 3-5。

propertyValue

如果 *propertyAction* 是：

- SMC_PROP_ACT_CLEAR — 则忽略 *propertyValue*。
- SMC_PROP_ACT_GET — 则表示指向一个联合的指针，smc_connect_props 会将所请求的信息放入此联合中。
- SMC_PROP_ACT_SET — 则表示指向一个联合的指针，此联合包含要将属性设置成的值。

bufferLength

以字节为单位表示的 *(*propertyValue->stringValue*) 数据长度。仅当 *propertyValue* 是指向字符串的指针时才使用。如果 *propertyAction* 是：

- SMC_PROP_ACT_CLEAR — 则忽略 *bufferLength*，且必须将 SMC_UNUSED 传递给它。
- SMC_PROP_ACT_GET — 则忽略 *bufferLength*，且必须将 SMC_UNUSED 传递给它。
- SMC_PROP_ACT_SET — *bufferLength* 必须包含 *(*propertyValue->stringValue*) 的字节数或 SMC_NULLTERM，以便通过一个起终止作用的空值字节来指示字符串的长度。

outputLengthHandle

指向整型变量的指针。仅当 *propertyValue* 是指向字符串的指针时才使用。如果 *propertyAction* 是：

- SMC_PROP_ACT_CLEAR — 则忽略 *outputLengthHandle*，且必须将空值传递给它。
- SMC_PROP_ACT_GET — 则表示所请求信息的长度 （以字节为单位）。包含实际写入 *propertyValue->stringValue* 的字节数 （不包括起终止作用的空值字节）。如果不需要该信息，可传递空值。
- SMC_PROP_ACT_SET — 则忽略 *outputLengthHandle*，且必须将空值传递给它。

返回值

返回值	含义
SMC_RET_SUCCESS	函数成功。
SMC_RET_FAILURE	函数失败。
SMC_RET_INVALID_CONNECT	不存在连接。

示例

下面的示例假定您此前已使用 smc_connect_alloc 分配了一个连接，且已有一个 *clientId*。

```
SMC_VALUE_UNION  value.sizetValue = 512;
 if (smc_connect_props(clientId,
      SMC_PROP_ACT_SET,
      SMC_PROP_PACKETSIZE,
      &value,
      0,
      NULL) != SMC_RET_SUCCESS)
 {
       printf("smc_connect_props failed\n");
       /* do some cleanup */
 }
```

用法

- 清除属性时会将其重置为其缺省值。
- smc_connect_props 必须在调用 smc_connect_alloc 之后调用。
- 调用 smc_connect_ex 前，必须设置连接的以下属性：
  - SMC_PROP_PASSWORD
  - SMC_PROP_SERVERNAME
  - SMC_PROP_USERNAME
- *serverMode* 确定可对该连接应用其它哪些 Monitor Client Library 函数。例如， smc_create_recording_session 不能应用于活动连接。
- *serverMode* （在创建连接时指定）确定公用函数的行为。例如，smc_create_view 可用于创建一个活动视图或历史视图。
- 对于用于定义记录会话的活动连接和历史连接，SMC_PROP_USERNAME 属性必须设置为 “sa”（即拥有 sa_role 的 Adaptive Server 帐户的名称），或者设置为对存储过程 master.dbo.mon_rpc_connect 拥有执行权限的 Adaptive Server 帐户的名称。
- 若只检索字符串的长度，请为 *propertyValue* 传递空值，为 *outputLengthHandle* 传递一个有效指针。
- 有关 SMC_VALUE_UNION 结构的定义，请参见第 240 页的“联合：SMC_VALUE_UNION”。
- 对于类型为 SMC_CHARP 的数据， *stringValue* 指向该值。 Monitor Client Library 为该字符串分配内存，且主调应用程序必须使用 free() 释放它。

- 下列属性仅在建立连接前有效：
  - SMC_PROP_APPNAME
  - SMC_PROP_IFILE
  - SMC_PROP_PASSWORD
  - SMC_PROP_SERVERMODE
  - SMC_PROP_SERVERNAME
  - SMC_PROP_USERNAME.

  如果在建立连接后改变了这些属性，它们将在下次调用 smc_connect_ex 时生效。
- 表 3-5 总结了 Monitor Client Library 的各个属性，能否设置、检索或清除它们，以及每个属性值的数据类型：

表 3-5：Monitor Client Library 连接属性

属性	设置、获取或清除	*propertyValue 是	缺省值
SMC_PROP_APPNAME	全部	SMC_CHARP	一个空字符串
SMC_PROP_ERROR_CALLBACK	设置 / 获取	一个函数指针 （使用 SMC_VALUE_UNION 的 *voidpValue* 成员）	
SMC_PROP_IFILE	全部	SMC_CHARP	空字符串，表示 *SYBASE* 环境变量所指向的目录中的 *interfaces* 文件 （在 Windows 中，指 *ini* 子目录中的 *sql.ini*）
SMC_PROP_LOGIN_TIMEOUT	全部	SMC_SIZET	0 （使用服务器缺省值）
SMC_PROP_PACKETSIZE	全部	SMC_SIZET	0 （使用服务器缺省值）
SMC_PROP_PASSWORD	设置 / 清除	SMC_CHARP	一个空字符串
SMC_PROP_SERVERMODE	全部	SMC_INT	SMC_SERVER_M_LIVE
SMC_PROP_SERVERNAME	全部	SMC_CHARP	一个空字符串
SMC_PROP_TIMEOUT	全部	SMC_SIZET	0 （使用服务器缺省值）
SMC_PROP_USERDATA	全部	SMC_VOIDP	NULL
SMC_PROP_USERNAME	全部	SMC_CHARP	一个空字符串

属性

属性	说明
SMC_PROP_APPNAME	使用 Monitor Client Library 的应用程序名。可以随时修改此属性，但仅在调用 smc_connect_ex 时生效。
SMC_PROP_ERROR_CALLBACK	错误回调函数。在连接期间，可以随时修改此属性。
SMC_PROP_IFILE	*interfaces* 文件。可以随时修改此属性，但仅在调用 smc_connect_ex 时生效。
SMC_PROP_LOGIN_TIMEOUT	登录时使用的超时值 （单位为秒）。可以随时修改此属性，但仅在调用 smc_connect_ex 时生效。
SMC_PROP_PACKETSIZE	与服务器通信时使用的包大小。在连接期间，可以随时修改此属性。
SMC_PROP_PASSWORD	口令。可以随时修改此属性，但仅在调用 smc_connect_ex 时生效。
SMC_PROP_SERVERMODE	服务器模式。只能在建立连接前设置此属性。可以随时修改此属性，但仅在调用 smc_connect_ex 时生效。该值是一个枚举：SMC_SERVER_MODE。请参见第 239 页的 “枚举：SMC_SERVER_MODE”。
SMC_PROP_SERVERNAME	服务器名。可以随时修改此属性，但仅在调用 smc_connect_ex 时生效。
SMC_PROP_TIMEOUT	发送到服务器的请求所用的超时值。在连接期间，可以随时修改此属性。
SMC_PROP_USERDATA	用户提供的指针。此指针传递回回调函数。在可用的连接上可以随时修改此属性。
SMC_PROP_USERNAME	此连接使用的 *username*。可以随时修改此属性，但仅在调用 smc_connect_ex 时生效。

有效的服务器模式

模式	可用性
SMC_SERVER_M_LIVE	可用
SMC_SERVER_M_HISTORICAL	可用

错误

错误	含义
SMC_RET_INVALID_API_FUNCTION	在程序中无效地使用了旧函数和替换函数。
SMC_RET_INVALID_PARAMETER	参数值无效。

另请参见 smc_connect_alloc、 smc_connect_ex

smc_create_alarm_ex

说明

在连接视图内针对一个数据项创建报警。

语法

```
SMC_RETURN_CODE smc_create_alarm_ex
(SMC_CONNECT_ID          clientId,
SMC_VIEW_ID              viewId,
SMC_DATAITEM_STRUCTP     dataItemHandle,
SMC_VALUE_UNIONP         alarmValueDataHandle,
SMC_DATAITEM_TYPE        alarmDatatype,
SMC_ALARM_ACTION_TYPE    alarmActionType,
SMC_CHARP                alarmActionData,
SMC_VOIDP                userDataHandle,
SMC_GEN_CALLBACK         alarmCallback,
SMC_ALARM_IDP            alarmIdHandle)
```

参数

clientId

标识连接。

viewId

标识针对连接创建的视图。

dataItemHandle

指向数据项和统计类型的指针。

alarmValueDataHandle

指向一个阈值的指针，达到或高于此阈值则触发报警。

alarmDatatype

报警值的类型必须是下列之一，且必须与给定数据项的预期数据类型匹配：

- SMC_DI_TYPE_DOUBLE
- SMC_DI_TYPE_INT
- SMC_DI_TYPE_LONG

alarmActionType

- SMC_ALARM_A_NOTIFY
  （仅适用于 SMC_SERVER_ M_LIVE 模式） — 调用报警回调。
- SMC_ALARM_A_EXEC_PROC （仅适用于 SMC_SERVER_ M_HISTORICAL 模式） — 调用指定的外部程序。
- SMC_ALARM_A_LOG_TO_FILE （仅适用于 SMC_SERVER_ M_HISTORICAL 模式） — 向日志文件写入消息。

alarmActionData

指向以空值终止的字符串的指针，字符串的内容取决于 *alarmActionType*。如果 *alarmActionType* 等于：

- SMC_ALARM_A_NOTIFY — 则忽略 *alarmActionData*。
- SMC_ALARM_A_EXEC_PROC — 则表示一个以空值终止的字符串，其中包含了要调用的程序的文件名和可选参数列表。
- SMC_ALARM_A_LOG_TO_FILE — 则表示包含日志文件名且以空值终止的字符串。

这些文件位于正在运行 Historical Server 的系统中 （此系统不必是正在运行应用程序的系统）。 Historical Server 必须拥有这些文件的访问权限。

userDataHandle

用户提供的指针。

alarmCallback

标识 *alarmActionType* 使用的通知函数 SMC_ALARM_A_NOTIFY。

alarmIdHandle

指向一个变量的指针，变量应声明为 SMC_ALARM_ID 类型。如果对 smc_create_alarm 的调用成功，则此变量包含该报警的 ID。

返回值

返回值	含义
SMC_RET_SUCCESS	函数成功。
SMC_RET_FAILURE	函数失败。
SMC_RET_INVALID_CONNECT	不存在连接。

示例

下面的示例假定：

- 您已使用 smc_connect_ex 创建一个连接，且已有一个 *clientId*。
- 已针对该连接创建一个视图，且具有一个 *viewId*。
- 该视图包含 *dataItem* SMC_NAME_PAGE_LOGICAL_READ、SMC_STAT_VALUE_SAMPLE。

- 您已定义了一个报警处理程序函数 *myAlarmHandler*。

```
SMC_DATAITEM_STRUCT  dataItem =
                     { SMC_NAME_PAGE_LOGICAL_READ,
                       SMC_STAT_VALUE_SAMPLE };
SMC_DATAITEM_STRUCTP dataItemHandle = &dataItem;
SMC_VALUE_UNION alarmValue;
SMC_VALUE_UNIONP alarmValueHandle = &alarmValue;
SMC_ALARM_ID  alarmId;
SMC_ALARM_IDP alarmIdHandle = &alarmId;
alarmValue.longValue = 10L;

if (smc_create_alarm_ex(clientId,
        viewId,
        dataItemHandle,
        alarmValueHandle,
        SMC_DI_TYPE_LONG,
        SMC_ALARM_A_NOTIFY,
        NULL, /* ignored */
        NULL,/* no user data */
        myAlarmHandler,
        alarmIdHandle) != SMC_RET_SUCCESS)
{
  printf("smc_create_alarm_ex failed\n");
  /* do some cleanup */
}
```

用法

- 可以针对结果数据项创建报警，但不能针对键数据项创建。
- *alarmId* 仅在一个给定视图内是唯一的。
- 对于视图中每个行，若其数据项值达到或超出阈值，将触发报警。
- 在刷新调用环境中，报警在过滤器之后应用。
- 在每次刷新时，报警是根据数据项的值 （状态）触发的，而不是根据数据项值的变化 （转换）来触发。
- 针对同一数据项可创建多个报警。
- 当在记录会话定义期间在一个 Historical Server 连接中使用时，smc_create_alarm_ex 定义一个报警，报警将在记录会话执行期间创建。
- 在回放会话期间，不能在 Historical Server 连接中定义报警。

- 创建 “记录到文件”报警时，如果为日志文件的位置指定了一个 UNIX 目录，应确保该目录有效并装入到正在运行 Historical Server 的计算机上。还要确保对该目录有写的权限。如果指定的目录无效、未装入或不可写， Historical Server 将不创建日志文件，也不会发布消息以警告位置无效。

报警回调的语法如下：

```
SMC_VOID AlarmCallback
(SMC_CONNECT_ID      clientId,
SMC_COMMAND_ID       commandId,
SMC_VOIDP            userDataHandle)
```

有效的服务器模式

模式	可用性
SMC_SERVER_M_LIVE	可用
SMC_SERVER_M_HISTORICAL	可用 （用于记录）

错误

错误	含义
SMC_RET_INSUFFICIENT_MEMORY	内存不足
SMC_RET_INVALID_ALARM_VALUE	无效的报警值
SMC_RET_INVALID_API_FUNCTION	在同一程序中无效地使用了旧函数和替换函数
SMC_RET_INVALID_DATAITEM_FOR_ALARM	数据项统计类型或报警值不匹配
SMC_RET_INVALID_DATATYPE	无效数据类型
SMC_RET_INVALID_DINAME	数据项不存在
SMC_RET_INVALID_DISTAT	数据项统计类型不存在
SMC_RET_INVALID_PARAMETER	无效的参数值
SMC_RET_INVALID_VIEWID	视图不存在
SMC_RET_INTERNAL_ERROR	内部错误

回调参数

参数	说明
clientId	标识连接。
commandId	标识一个命令的实例。
userDataHandle	为此报警调用的 smc_create_alarm 设置的指针。

报警回调函数使用 smc_get_command_info 获得有关触发报警的情况的信息。

另请参见 smc_connect_ex、 smc_drop_alarm、 smc_get_command_info

smc_create_filter

说明

针对视图中的数据项创建过滤器。视图中的每个数据项只能有一个过滤器。

此函数可用于 Monitor Server 和 Historical Server 中。当用于 Historical Server 时 （即当连接模式是 SMC_SERVER_M_HISTORICAL 时），它为要定义的记录会话创建一个过滤器。

语法

```
SMC_RETURN_CODE smc_create_filter
(SMC_CONNECT_ID       clientId,
SMC_VIEW_ID           viewId,
SMC_DATAITEM_STRUCTP  dataItemHandle,
SMC_FILTER_TYPE       filterType,
SMC_VALUE_UNIONP      filterValueListHandle,
SMC_SIZET             filterValueListLength,
SMC_DATAITEM_TYPE     filterDatatype,
SMC_FILTER_IDP        filterIdHandle)
```

参数

clientId

标识连接。

viewId

标识针对连接创建的视图。

dataItemHandle

数据项和统计类型。如果过滤器类型是以下任一种，则数据项必须是数字：

- SMC_FILT_T_GE
- SMC_FILT_T_LE
- SMC_FILT_T_GE_AND_LE
- SMC_FILT_TOP_N

filterType

要应用的过滤器的类型。有效的过滤器类型是：

- SMC_FILT_T_EQ — 等于。
- SMC_FILT_T_NEQ — 不等于。
- SMC_FILT_T_GE — 大于或等于。
- SMC_FILT_T_LE — 小于或等于。
- SMC_FILT_T_GE_AND_LE — 一个下限，后跟一个上限。
- SMC_FILT_T_TOP_N — 前 N 个。

filterValueListHandle

指向一个过滤器值数组的指针。过滤器值的数量取决于过滤器类型：

- SMC_FILT_T_EQ — 一个或多个。
- SMC_FILT_T_NEQ — 一个或多个。
- SMC_FILT_T_GE — 一个。
- SMC_FILT_T_LE — 一个。
- SMC_FILT_T_GE_AND_LE — 两个；下限必须是列表中的第一个元素，上限是第二个元素。
- SMC_FILT_T_TOP_N — 一个。

filterValueListLength

filterValueListHandle 中列出的过滤器值的数目。

filterDataType

过滤器值的数据类型，是下列类型之一：

- SMC_DI_TYPE_CHARP
- SMC_DI_TYPE_DATIM
- SMC_DI_TYPE_DOUBLE
- SMC_DI_TYPE_ENUMS
- SMC_DI_TYPE_INT
- SMC_DI_TYPE_LONG

必须与数据项的数据类型匹配。过滤器值必须也是此类型，以下情形除外：

- 如果过滤器类型是 SMC_FILT_T_TOP_N， *filterValueListHandle* 中的过滤器值必须是 SMC_INT 类型。
- 如果数据类型是 SMC_DI_TYPE_ENUMS， *filterValueListHandle* 中的过滤器值必须使用 *intValue* 成员加以传递。

filterIdHandle

指向一个变量的指针，变量应声明为 SMC_FILTER_ID 类型。如果对 smc_create_filter 的调用成功，则此变量包含该过滤器的 ID。

返回值

返回值	含义
SMC_RET_SUCCESS	函数成功。
SMC_RET_FAILURE	函数失败。
SMC_RET_INVALID_CONNECT	不存在连接。

示例

下例假定：

- 已创建一个连接并具有一个 *clientId*。
- 已针对该连接创建一个视图，且具有一个 *viewId*。
- 该视图包含示例中定义的 *dataItem*。

```
SMC_DATAITEM_STRUCT  dataItem =
                       { SMC_NAME_PAGE_LOGICAL_READ,
                          SMC_STAT_VALUE_SAMPLE };
 SMC_DATAITEM_STRUCTP dataItemHandle = &dataItem;
 SMC_VALUE_UNION filterValue;
 SMC_VALUE_UNIONP filterValueHandle = &filterValue;
 SMC_FILTER_ID filterId;
 SMC_FILTER_IDP filterIdHandle = &filterId;
 filterValue.longValue = 10L;

if (smc_create_filter(clientId,
          viewId,
   dataItemHandle,
          SMC_FILT_T_GE,
          filterValueHandle,
          1,   /* just one filterValue */
          SMC_DI_TYPE_LONG,
          filterIdHandle) != SMC_RET_SUCCESS)
 {
   printf("smc_create_filter failed\n");
   /* do some cleanup */
 }
```

用法

- 应用程序可以对适用于字符串数据类型的所有过滤器使用通配符 (%)。
- 在刷新调用环境中，过滤器在报警之前应用。
- 一个数据项上只能创建一个过滤器。
- 直到执行记录会话时，才会创建为记录会话定义的过滤器。
- 在回放时不允许创建。

- 对于数据库对象，可以针对对象名定义 SMC_FILT_T_EQ 过滤器，即针对 SMC_NAME_OBJ_NAME 或 SMC_NAME_ACT_STP_NAME 数据项进行定义。字符串值必须包含全限定对象名，如 *database.owner.object*。但可以对该名称的每个组成部分使用通配符。

有效的服务器模式

模式	可用性
SMC_SERVER_M_LIVE	可用
SMC_SERVER_M_HISTORICAL	可用 （仅用于记录）

错误

错误	含义
SMC_RET_INSUFFICIENT_MEMORY	内存不足
SMC_RET_INVALID_COMPOSITE_FILTER	无效的复合过滤器
SMC_RET_MISSING_DATAITEM	缺少数据项
SMC_RET_INVALID_DATATYPE	无效数据类型
SMC_RET_INVALID_DINAME	数据项无效
SMC_RET_INVALID_DISTAT	数据项统计类型无效
SMC_RET_INVALID_FILTER_VALUE	无效的过滤器值
SMC_RET_INVALID_FILTER_RANGE	无效范围值
SMC_RET_INVALID_VALUE_COUNT	*filterValueListLength* 的值无效
SMC_RET_INVALID_VIEWID	视图不存在

另请参见 smc_drop_filter

smc_create_playback_session

说明

在 Historical Server 上初始化回放会话。

语法

```
SMC_RETURN_CODE smc_create_playback_session
(SMC_CONNECT_ID          clientId,
SMC_SESSION_IDP          sessionIdArray,
SMC_SIZET                numInputSessions,
SMC_CHARP                startTime,
SMC_CHARP                endTime,
SMC_HS_PLAYBACK_OPT      playbackType,
SMC_SIZET                summarizationInterval,
SMC_HS_ESTIM_OPT         estimationOption,
SMC_HS_MISSDATA_OPT      missingDataOption,
SMC_HS_TARGET_OPT        playbackTarget,
SMC_CHARP                directoryName,
SMC_HS_SESS_PROT_LEVEL   protectionLevel,
SMC_HS_SESS_SCRIPT_OPT   scriptOption,
SMC_HS_SESS_DELETE_OPT   deleteOption,
SMC_SESSION_IDP sessionIdHandle)
```

参数

clientId

标识连接。

sessionIdArray

会话编号数组，其中包含的会话编号标识 Historical Server 上为该回放会话提供数据的现有记录会话。如果指定了多个输入会话，那么它们必须都已定义为记录来自相同 Adaptive Server 的数据，并且它们必须按发生的时间顺序排序。

如果 *playbackTarget* 是 SMC_HS_TARGET_FILE，则在由多个输入会话占用的时间之间不能有任何间隙。输入会话必须包含介于 *startTime* 和 *endTime* 参数之间的所有时间内的数据。

numInputSessions

输入会话的数目，即 *sessionIdArray* 的长度。必须至少为一个。

startTime

以空值终止的字符串，包含开始回放的时间，使用的格式为：

```
yyyy/mm/dd hh:mm[:ss] [time zone]
```

缺省值是从第一个输入会话起始时间处开始。

endTime

以空值终止的字符串，包含结束回放的时间，使用的格式为：

```
yy/mm/dd hh:mm[:ss] [time zone]
```

缺省值是在最后一个输入会话结束时间处停止。

playbackType

指定回放的详细程度。有效值为：

- SMC_HS_PBTYPE_RAW — 按照数据的收集次序回放数据，回放时使用的是输入会话中包含的所有间隔 （可能会发生变化）。此选项可包括快照数据，例如当前 SQL 语句数据和锁或进程的状态。仅对 *playbackTarget* SMC_HS_TARGET_CLIENT 有效。
- SMC_HS_PBTYPE_ACTUAL — 按输入会话中包含的所有间隔（可能会发生变化）回放数据。此选项不能包括快照数据。
- SMC_HS_PBTYPE_INTERVAL — 回放已汇总到采样间隔内的数据，间隔长度在 *summarizationInterval* 中指定。
- SMC_HS_PBTYPE_ENTIRE — 回放被汇总成单个采样的每个输入记录会话的数据。采样间隔是介于请求的回放 *startTime* 和 *endTime* 之间的时间。

如果 *playbackTarget* 是 SMC_HS_TARGET_FILE，那么 *playbackType* 必须是 SMC_HS_PBTYPE_INTERVAL 或 SMC_HS_PBTYPE_ENTIRE。

summarizationInterval

如果 *playbackType* 是 SMC_HS_PBTYPE_INTERVAL，那么该参数以秒为单位指定回放间隔的长度，在此间隔内汇总输入数据。

对于 *playbackType* 的其它值，应用程序必须为此参数指定 SMC_UNUSED。

estimationOption

指定回放是否可以对不能精确计算的数据项值进行估计。有效值为：

- SMC_HS_ESTIM_ALLOW
- SMC_HS_ESTIM_DISALLOW

如果指定了 SMC_HS_ESTIM_DISALLOW，那么随后为此回放会话调用 smc_create_view 时，如果它包括需要估计的数据项，则将返回一个错误。

如果 *playbackType* 是 SMC_HS_PBTYPE_RAW，该选项将被忽略。

missingDataOption

指定对于输入会话中无可用数据的时间段， Monitor Client Library 是否返回回放采样。有效值为：

- SMC_HS_MISSDATA_SHOW — Monitor Client Library 将为无可用数据的时间段返回一个采样。
- SMC_HS_MISSDATA_SKIP — Monitor Client Library 不会为无可用数据的时间段返回一个采样，而是为下一个具有可用数据的可用时间间隔返回数据。

如果 *playbackTarget* 是 SMC_HS_TARGET_FILE，则忽略此参数。

playbackTarget

指定回放会话是否将数据返回给应用程序，或者回放是否在 Historical Server 上创建新会话。有效值为：

- SMC_HS_TARGET_CLIENT — 回放会话通过调用 smc_refresh_ex 将数据返回给应用程序。
- SMC_HS_TARGET_FILE — 回放在 Historical Server 上创建一个新会话。

directoryName

如果 *playbackTarget* 是 SMC_HS_TARGET_FILE，此参数指定一个目录，在此目录中， Historical Server 为要创建的新会话创建数据文件和错误文件。

protectionLevel

如果 *playbackTarget* 是 SMC_HS_TARGET_FILE，此参数指定要创建的新会话的保护级别。有效值为：

- SMC_HS_SESS_PROT_PUBLIC
- SMC_HS_SESS_PROT_PRIVATE

如果 *playbackTarget* 是 SMC_HS_TARGET_CLIENT，则忽略此参数。

scriptOption

如果 *playbackTarget* 是 SMC_HS_TARGET_FILE，此参数指定 Historical Server 是否必须创建一个脚本，以便让该脚本创建用于将结果 （来自新会话）装载到 Adaptive Server 中的表。选项有：

- SMC_HS_SESS_SCRIPT_NONE — 无脚本。
- SMC_HS_SESS_SCRIPT_SYBASE — Sybase 脚本。

如果 *playbackTarget* 是 SMC_HS_TARGET_CLIENT，则忽略此参数。

deleteOption

如果 *playbackTarget* 是 SMC_HS_TARGET_FILE，此参数指定在成功创建新会话后， Historical Server 是否必须删除输入会话。选项有：

- SMC_HS_DELETE_FILES
- SMC_HS_RETAIN_FILES

如果 *playbackTarget* 是 SMC_HS_TARGET_CLIENT，则忽略此参数。

sessionIdHandle

如果 *playbackTarget* 是 SMC_HS_TARGET_FILE，此参数必须是一个指向 SMC_SESSION_ID 类型变量的指针，Monitor Client Library 将新会话的标识符写到其中。

如果 *playbackTarget* 是 SMC_HS_TARGET_CLIENT，则忽略此参数。

返回值

返回值	含义
SMC_RET_SUCCESS	函数成功。
SMC_RET_FAILURE	函数失败。
SMC_RET_INVALID_CONNECT	不存在连接。

示例

下面的示例假定您已创建了一个到 Historical Server 的连接，且已有一个 *clientId*。

```
SMC_SESSION_ID  inputSessions[2];

if (smc_create_playback_session(clientId,
         inputSessions,
         2,/* number of input sessions */
"",/* default start time */
"",/* default end time */
         SMC_HS_PBTYPE_INTERVAL,/* summarize at */
         60,/* uniform minute intervals */
         SMC_HS_ESTIM_ALLOW,/* allow estimation */
         SMC_HS_MISSDATA_SHOW, /* produce a sample*/
              /* every minute even if no data is */
              /* available for that interval */
         SMC_HS_TARGET_CLIENT, /* do playback */
         "",/* directory name */
         SMC_HS_SESS_PROT_PUBLIC,/* so next 5 */
         SMC_HS_SESS_SCRIPT_SYBASE,/* are */
         SMC_HS_DELETE_FILES,/* unused */
         NULL)/* No output session ID */
    != SMC_RET_SUCCESS)
 {
```

```
        printf("smc_create_playback_session failed\n");
            /* do some cleanup */
    }
```

用法

- 在一个 Historical Server 连接中，记录会话和回放会话是相互排斥的。连接到 Historical Server 并定义记录会话的应用程序，在创建回放会话之前，必须使用函数 smc_initiate_recording 完成记录会话的定义。
- 如果 *playbackType* 是 SMC_HS_PBTYPE_RAW，应用程序只能指定一个输入会话。否则，倘若会话是针对相同的 Adaptive Server 安装和 Monitor Server 而记录的，应用程序可指定任何数目的输入会话（但至少有一个）。
- 如果 *playbackType* 是 SMC_HS_PBTYPE_RAW，则对回放视图的定义应用不同的规则。有关视图的详细信息，请参见 《Adaptive Server Enterprise Monitor Historical Server 用户指南》。
- 不能将 *playbackTarget* SMC_HS_TARGET_FILE 与 *playbackType* SMC_HS_PBTYPE_RAW 或 SMC_HS_PBTYPE_ACTUAL 组合使用。
- 输入会话可包括仍在执行记录的记录会话，除非 *playbackTarget* 是 SMC_HS_TARGET_FILE。
- 如果 *playbackTarget* 是 SMC_HS_TARGET_FILE，那么输入会话必须包含从 *startTime* 到 *endTime* 的整段时间内的性能数据，且在输入会话之间没有间隙。
- 有关 hs_create_playback_session 命令的详细信息，请参见 《Monitor Historical Server 用户指南》。

有效的服务器模式

模式	可用性
SMC_SERVER_M_LIVE	不可用
SMC_SERVER_M_HISTORICAL	可用

错误

错误	含义
SMC_RET_INTERNAL_ERROR	内部错误
SMC_INVALID_SVR_MODE	服务器模式无效

另请参见　smc_initiate_playback

smc_create_recording_session

说明

在 Historical Server 上启动对记录会话的定义。

仅当连接模式是 SMC_SERVER_M_HISTORICAL 时，此函数才适用。

语法

```
SMC_RETURN_CODE smc_create_recording_session
(SMC_CONNECT_ID          clientId,
SMC_CHARP                SMSName,
SMC_INT                  sampleInterval,
SMC_CHARP                directoryName,
SMC_CHARP                startTime,
SMC_CHARP                endTime,
SMC_HS_SESS_PROT_LEVEL   protectionLevel,
SMC_HS_SESS_ERR_OPT      errOption,
SMC_HS_SESS_SCRIPT_OPT   scriptOption,
SMC_SESSION_IDP          sessionIdHandle)
```

参数

clientId

标识连接。

SMSName

以空值终止的字符串，包含 Monitor Server 名。

sampleInterval

两个连续数据采样之间的等待时间 （单位为秒）。

directoryName

以空值终止的字符串，包含一个目录的完整路径名，该目录包含在执行此记录会话期间由 Historical Server 创建的数据和错误文件。

在运行 Historical Server 的系统中，该目录必须可写入。此系统不必与运行调用此函数的客户端应用程序的系统相同。

startTime

以空值终止的字符串，包含开始记录的时间，使用的格式为：

```
yyyy/mm/dd hh:mm[:ss] [time zone]
```

缺省值是立即开始。

endTime

以空值终止的字符串，包含停止记录的时间，使用的格式为：

```
yy/mm/dd hh:mm[:ss] [time zone]
```

缺省是 *startTime* 后 24 小时停止。

protectionLevel

所记录数据的保护级别。有效值为：

- SMC_HS_SESS_PROT_PUBLIC
- SMC_HS_SESS_PROT_PRIVATE

errOption

指示当遇到非致命错误时 Historical Server 该作何处理。选项有：

- SMC_HS_SESS_ERR_CONT — 继续执行该会话。
- SMC_HS_SESS_ERR_HALT — 停止该会话。

scriptOption

表示 Historical Server 是否必须创建一个脚本文件，以让该文件创建用于将结果 （来自此记录会话）装载到 Adaptive Server 中的表。选项有：

- SMC_HS_SESS_SCRIPT_NONE — 无脚本。
- SMC_HS_SESS_SCRIPT_SYBASE — Sybase 脚本。

sessionIdHandle

指向一个应声明为 SMC_SESSION_ID 类型的变量的指针。如果对 smc_create_recording_session 的调用成功，此变量包含该记录会话的 ID。

返回值

返回值	含义
SMC_RET_SUCCESS	函数成功。
SMC_RET_FAILURE	函数失败。
SMC_RET_INVALID_CONNECT	不存在连接。

示例

下面的示例假定您已创建了一个到 Historical Server 的连接，且已有一个 *clientId*。

```
SMC_SESSION_ID  sessionId;
 SMC_SESSION_IDP sessionIdHandle = &sessionId;
if (smc_create_recording_session(clientId,
          "myMonitorServer",
          60,   /* sample interval (seconds) */
          "/usr/hist_serv_home_dir",
          "95/07/22 15:00",  /* start time */
          "95/07/23 15:30",  /* end time */
          SMC_HS_SESS_PROT_PUBLIC,
          SMC_HS_SESS_ERR_CONT,
          SMC_HS_SESS_SCRIPT_SYBASE,
         sessionIdHandle) != SMC_RET_SUCCESS)
 {
```

```
        printf("smc_create_recording_session failed\n");
              /* do some cleanup */
        }
```

用法

- 在一个 Historical Server 连接中，记录会话和回放会话是相互排斥的。在创建记录会话之前，连接到 Historical Server 并创建回放会话的应用程序必须使用函数 smc_terminate_playback 结束回放会话。
- 有关 hs_create_recording_session 命令的详细信息，请参见《Adaptive Server Enterprise Monitor Historical Server 用户指南》。

有效的服务器模式

模式	可用性
SMC_SERVER_M_LIVE	不可用
SMC_SERVER_M_HISTORICAL	可用

错误

错误	含义
SMC_RET_INTERNAL_ERROR	内部错误
SMC_RET_INVALID_SVR_MODE	服务器模式无效

另请参见 smc_initiate_recording

smc_create_view

说明

创建可包含一个或多个数据项的视图。

有关数据项的信息，请参见第 2 章 “数据项和统计类型”。

可以与 Monitor Server 和 Historical Server 一起使用 smc_create_view 函数。当与 Historical Server (SMC_SERVER_M_HISTORICAL) 一起使用时，它为要定义的记录会话或回放会话创建一个视图。

语法

```
SMC_RETURN_CODE smc_create_view
(SMC_CONNECT_ID       clientId,
SMC_DATAITEM_STRUCTP  dataItemListHandle,
SMC_SIZET             dataItemListLength,
SMC_CHARP             viewName,
SMC_VIEW_IDP          viewIdHandle)
```

参数

clientId
标识连接。

dataItemListHandle
指向 SMC_DATAITEM_STRUCTs 数组的指针。

dataItemListLength
dataItemListHandle 指向的数组中的数据项数目。

viewName
以空值终止的字符串，包含该视图的描述性名称。此名称可包含 a - z、 A - Z、 0 - 9 和下划线 (_) 字符，也可以是 NULL。

仅对 Historical Server 连接使用。对于活动连接，将忽略视图名。

viewIdHandle
指向一个变量的指针，变量应声明为 SMC_VIEW_ID 类型。如果对 smc_create_view 的调用成功，则此变量包含该视图的 ID。

返回值

返回值	含义
SMC_RET_SUCCESS	函数成功。
SMC_RET_FAILURE	函数失败。
SMC_RET_INVALID_CONNECT	不存在连接。

示例

下面的示例假定您已创建了一个连接且已有 *clientId*。

```
SMC_DATAITEM_STRUCT  dataItem =
                       { SMC_NAME_PAGE_LOGICAL_READ,
                         SMC_STAT_VALUE_SAMPLE };
 SMC_DATAITEM_STRUCTP dataItemHandle = &dataItem;
 SMC_VIEW_ID viewId;
 SMC_VIEW_IDP viewHandle = &viewId;

if (smc_create_view(clientId,
          dataItemHandle,
          1,   /* just one dataItem */
          NULL,   /* this is a Monitor Server view */
          viewIdHandle) != SMC_RET_SUCCESS)
 {
       printf("smc_create_view failed\n");
       /* do some cleanup */
 }
```

用法

- 请参见第 2 章 “数据项和统计类型”以了解有关使用含有活动视图的视图的规则。
- 当针对 Historical Monitor 连接调用 smc_create_view 时，必须在调用 smc_create_recording_session 或 smc_create_playback_session 之后调用它。
- 当在定义记录会话期间用于 Historical Server 时，它定义了一个在记录会话期间由 Historical Server 记录的视图。
- 当在回放会话期间用于 Historical Server 时，它从以前在记录会话中记录的视图中为回放选择一个视图。如果回放会话使用多个输入视图，那么所选的视图必须存在于所有输入会话中，并使用相同的名称、数据项和过滤器。
- 根据回放会话是为 “原始”回放还是为汇总回放创建的，回放视图可能包含、也可能不包含来自原始视图的数据项。有关 hs_create_playback_view 命令的详细信息，请参见 《Adaptive Server Enterprise Monitor Historical Server 用户指南》。

有效的服务器模式

模式	可用性
SMC_SERVER_M_LIVE	可用
SMC_SERVER_M_HISTORICAL	可用

错误

错误	含义
SMC_RET_INVALID_API_FUNC_SEQUENCE	Monitor Client Library 函数调用顺序无效
SMC_RET_INVALID_DINAME	数据项无效
SMC_RET_INVALID_DI_STATTYPE	数据项统计类型无效
SMC_RET_INSUFFICIENT_MEMORY	内存不足

另请参见

smc_create_recording_session、 smc_create_playback_session、 smc_initiate_recording、 smc_initiate_playback、 smc_drop_view

smc_drop_alarm

说明

删除针对视图中某一数据项的报警。

语法

SMC_RETURN_CODE smc_drop_alarm
(SMC_CONNECT_ID *clientId*,
SMC_VIEW_ID *viewId*,
SMC_ALARM_ID *alarmId*)

参数

clientId
标识连接。

viewId
标识针对连接创建的视图。

alarmId
标识报警。

返回值

返回值	含义
SMC_RET_SUCCESS	函数成功。
SMC_RET_FAILURE	函数失败。
SMC_RET_BUSY	函数未执行，连接正忙。
SMC_RET_INVALID_CONNECT	不存在连接。

示例

下例假定：

- 已创建一个连接并具有一个 *clientId*。
- 已针对该连接创建一个视图，且具有一个 *viewId*。
- 已针对该视图创建一个报警，并已有一个 *alarmId*。

```
if (smc_drop_alarm(clientId,
                   viewId,
                   alarmId) != SMC_RET_SUCCESS)
 {
   printf("smc_drop_alarm_failed\n");
   exit(1);
 }
```

用法

不能删除在定义历史会话时 （即当连接模式是 SMC_SERVER_M_HISTORICAL 时）创建的报警。

有效的服务器模式

模式	可用性
SMC_SERVER_M_LIVE	可用
SMC_SERVER_M_HISTORICAL	不可用

错误

错误	含义
SMC_RET_INVALID_VIEWID	函数失败。
SMC_RET_INVALID_ALARMID	报警不存在。

另请参见

smc_create_alarm_ex、 smc_drop_view

smc_drop_filter

说明

删除针对数据项的过滤器。

语法

```
SMC_RETURN_CODE smc_drop_filter
(SMC_CONNECT_ID     clientId,
SMC_VIEW_ID         viewId,
SMC_FILTER_ID       filterId)
```

参数

clientId

标识连接。

viewId

标识针对连接创建的视图。

filterId

标识要删除的过滤器。

返回值

返回值	含义
SMC_RET_SUCCESS	函数成功。
SMC_RET_FAILURE	函数失败。
SMC_RET_INVALID_CONNECT	不存在连接。

示例

下例假定：

- 已创建一个连接并具有一个 *clientId*。
- 已针对该连接创建一个视图，且具有一个 *viewId*。
- 已在针对视图创建一个过滤器，且具有一个 *filterId*。

```
if (smc_drop_filter(clientId,
                    viewId,
                    filterId) != SMC_RET_SUCCESS)
 {
   printf("smc_drop_filter_failed\n");
   /* do some cleanup */
 }
```

用法

- 调用 smc_drop_filter 之后，下一次调用 smc_refresh 时，过滤器删除才生效。
- 不能删除在定义 Historical Server 会话时 （即当连接模式是 SMC_SERVER_M_HISTORICAL 时）创建的过滤器。

有效的服务器模式

模式	可用性
SMC_SERVER_M_LIVE	可用
SMC_SERVER_M_HISTORICAL	不可用

错误

错误	含义
SMC_RET_INVALID_VIEWID	视图不存在。
SMC_RET_INVALID_FILTERID	过滤器不存在。

另请参见

smc_create_filter、 smc_drop_view

smc_drop_view

说明

从连接删除视图。

语法

SMC_RETURN_CODE smc_drop_view
(SMC_CONNECT_ID *clientId*,
SMC_VIEW_ID *viewId*)

参数

clientId

标识连接。

viewId

标识针对连接创建的视图。

返回值

返回值	含义
SMC_RET_SUCCESS	函数成功。
SMC_RET_FAILURE	函数失败。
SMC_RET_INVALID_CONNECT	不存在连接。

示例

下例假定：

- 已创建一个连接并具有一个 *clientId*。
- 已针对该连接创建一个视图，且具有一个 *viewId*。

```
if (smc_drop_view(clientId,
                  viewId) != SMC_RET_SUCCESS)
 {
   printf("smc_drop_view_failed\n");
   /* do some cleanup */
 }
```

用法

- 将删除与该视图中的数据项关联的所有报警和过滤器。
- 不能删除在 Historical Server 会话中 （即当连接模式是 SMC_SERVER_M_HISTORICAL 时）创建的视图。

有效的服务器模式

模式	可用性
SMC_SERVER_M_LIVE	可用
SMC_SERVER_M_HISTORICAL	不可用

错误

错误	含义
SMC_RET_INVALID_VIEWID	视图不存在。

另请参见

smc_create_view、 smc_drop_alarm、 smc_drop_filter

smc_get_command_info

说明

检索关于报警或错误通知的详细信息。

语法

```
SMC_RETURN_CODE smc_get_command_info
(SMC_CONNECT_ID          clientId,
SMC_COMMAND_ID           commandId,
SMC_INFO_TYPE            infoType,
SMC_VALUE_UNIONP         infoValue,
SMC_SIZETP               outputLengthHandle)
```

参数

clientId

标识连接。

commandId

标识对一个回调函数的调用。

infoType

描述所请求信息的类型。请参见第 132 页的表 3-3。

infoValue

指向 SMC_VALUE_UNION 结构的指针，接收 *infoType* 的值。

outputLengthHandle

指向整型变量的指针。在成功调用 smc_get_command_info 后，Monitor Client Library 将其写入该变量中。要复制到 **infoValue* 中的数据的实际字节数（不包括空值终止符字节）。如果 *infoValue* 数据类型不是 SMC_CHARP，将忽略此参数。如果不需要该信息，可传递空值。

返回值

返回值	含义
SMC_RET_SUCCESS	函数成功。
SMC_RET_FAILURE	函数失败。
SMC_RET_INVALID_API_FUNCTION	在同一程序中无效地使用了旧函数和替换函数。
SMC_RET_INVALID_COMMAND	命令的实例不存在。
SMC_RET_INVALID_CONNECT	不存在连接。
SMC_RET_INVALID_INFOTYPE	所请求的信息类型的环境无效。
SMC_RET_INVALID_PARAMETER	参数值无效。

示例

下面的示例假定：

- 一个错误回调函数正在执行。
- 已创建一个连接并具有一个 *clientId*。
- 示例代码是在 Monitor Client Library API 回调函数的环境中使用的，该函数提供 *commandId*。

```
SMC_VALUE_UNION myValue;
 SMC_VALUE_UNIONP myValuePtr = &myValue;
 if (smc_get_command_info(clientId,
                  commandId,
                  SMC_INFO_ERR_NUM,
                  myValuePtr,
                  NULL) != SMC_RET_SUCCESS)
 {
   printf("smc_get_command_info failed\n");
   /* do some cleanup */
 }
```

用法

- 有关 SMC_VALUE_UNION 结构的定义，请参见第 240 页的 “联合：SMC_VALUE_UNION”。
- 对于类型为 SMC_CHARP 的数据， *stringValue* 指向该值。 Monitor Client Library 为该字符串分配内存，且主调应用程序必须使用 free() 释放它。
- 如仅想检索字符串的字节数，可给 *infoValue* 传递 null 值，给 *outputLengthHandle* 传递一个有效指针。
- 表 3-6 列出了命令 *infoType* 及其关联的数据类型：

表 3-6：Monitor Client Library 命令信息类型

信息类型	infoValue 数据类型	可用
SMC_INFO_ALARM_ACTION_DATA	SMC_CHARP	在一个报警回调函数中
SMC_INFO_ALARM_ALARMID	SMC_SIZET	在一个报警回调函数中
SMC_INFO_ALARM_CURRENT_VALUE	取决于数据项和统计类型组合。（请参见第 2 章 “数据项和统计类型”）	在一个报警回调函数中
SMC_INFO_ALARM_DATAITEM	SMC_VOIDP	在一个报警回调函数中
SMC_INFO_ALARM_ROW	SMC_SIZET	在一个报警回调函数中
SMC_INFO_ALARM_THRESHOLD_VALUE	取决于数据项 / 统计类型组合。（请参见第 2 章 “数据项和统计类型”）	在一个报警回调函数中
SMC_INFO_ALARM_TIMESTAMP	SMC_CHARP	在一个报警回调函数中
SMC_INFO_ALARM_VALUE_DATATYPE	SMC_INT	在一个报警回调函数中

信息类型	infoValue 数据类型	可用
SMC_INFO_ALARM_VIEWID	SMC_SIZET	在一个报警回调函数中
SMC_INFO_ERR_MAPSEVERITY	SMC_SIZET	在一个错误回调函数中
SMC_INFO_ERR_MSG	SMC_CHARP	在一个错误回调函数中
SMC_INFO_ERR_NUM	SMC_SIZET	在一个错误回调函数中
SMC_INFO_ERR_SEVERITY	SMC_SIZET	在一个错误回调函数中
SMC_INFO_ERR_SOURCE	SMC_SIZET	在一个错误回调函数中
SMC_INFO_ERR_STATE	SMC_SIZET	在一个错误回调函数中

有效的服务器模式

模式	可用性
SMC_SERVER_M_LIVE	可用
SMC_SERVER_M_HISTORICAL	可用

错误

此函数不调用错误回调函数。

另请参见　smc_create_alarm_ex

smc_get_dataitem_type

说明　返回指定数据项的数据类型。

语法

```
SMC_RETURN_CODE smc_get_dataitem_type
(SMC_DATAITEM_STRUCTP    dataItemHandle,
SMC_DATAITEM_TYPEP       ptrType)
```

参数

dataItemHandle
　指向数据项和统计类型的指针。

ptrType
　指向数据值类型的指针。

返回值

返回值	含义
SMC_RET_SUCCESS	函数成功。
SMC_RET_FAILURE	函数失败。

示例

```
SMC_DATAITEM_STRUCT  dataItem =
                        { SMC_NAME_PAGE_LOGICAL_READ,
                          SMC_STAT_VALUE_SAMPLE };
 SMC_DATAITEM_STRUCTP dataItemHandle = &dataItem;
 SMC_DATAITEM_TYPE dataItemType;
 SMC_DATAITEM_TYPEP dataItemTypeHandle = &dataItemType
;

if (smc_get_dataitem_type(dataItemHandle,
          dataItemTypeHandle) != SMC_RET_SUCCESS)
 {
       printf("smc_get_dataitem_type failed\n");
       /* do some cleanup */
 }
```

用法

- 数据项类型如下：

数据项类型	说明
SMC_DI_TYPE_CHARP	指向一个字符串的指针。
SMC_DI_TYPE_DATIM	Sybase 日期和时间。
SMC_DI_TYPE_DOUBLE	双精度浮点数。
SMC_DI_TYPE_ENUMS	与数据项相关的枚举数据类型。枚举类型在 *mctype.sh* 头文件和附录“数据类型和结构”中定义。
SMC_DI_TYPE_INT	整型。
SMC_DI_TYPE_LONG	长整型。

- 如果提供了 Monitor Client Library 不支持的数据项和统计类型，输出参数类型将设置为 SMC_DI_TYPE_NONE。

另请参见 smc_create_view

smc_get_dataitem_value

说明

返回刷新后的数据。此数据一次返回一行中的一个数据项。

语法

```
SMC_RETURN_CODE smc_get_dataitem_value
(SMC_CONNECT_ID        clientId,
SMC_VIEW_ID            viewId,
SMC_DATAITEM_STRUCTP   dataItemHandle,
SMC_SIZET              row,
SMC_VALUE_UNIONP       returnVal)
```

参数

clientId

标识连接。

viewId

标识针对连接创建的视图。

dataItemHandle

指向数据项和统计类型的指针。

row

所请求数据的行号。

returnVal

包含一个数据项的值的返回值。

返回值

返回值	含义
SMC_RET_SUCCESS	函数成功。
SMC_RET_FAILURE	函数失败。
SMC_RET_INVALID_CONNECT	不存在具有指定 ID 的连接。

示例

下例假定：

- 已创建一个连接并具有一个 *clientId*。
- 已针对该连接创建一个视图，且具有一个 *viewId*。
- 该视图包含示例中定义的 *dataItem*。
- 已成功执行了一次刷新调用。
- 行数大于零。

```
SMC_DATAITEM_STRUCT  dataItem =
                       { SMC_NAME_PAGE_LOGICAL_READ,
                          SMC_STAT_VALUE_SAMPLE };
 SMC_DATAITEM_STRUCTP dataItemHandle = &dataItem;
SMC_VALUE_UNION returnValue;
 SMC_VALUE_UNIONP returnValueHandle = &returnValue;
```

```
if (smc_get_dataitem_value(clientId,
        viewId,
        dataItemHandle,
        0,/* row number */
        returnValueHandle) != SMC_RET_SUCCESS)
 {
   printf("smc_get_dataitem_value failed\n");
   /* do some cleanup */
 }
```

用法

- 数据的第一行用行号 0 进行索引，第二行用 1 进行索引，依此类推。
- Monitor Client Library 为 SMC_DI_TYPE_CHARP 类型的数据分配内存。主调应用程序必须使用 free() 释放内存。
- 有关 SMC_VALUE_UNION 成员的列表，请参见附录 B “数据类型和结构”。
- 请参见 *mctype.sh* 头文件或附录 B “数据类型和结构”了解枚举类型的值。

错误

错误	含义
SMC_RET_INVALID_VIEWID	视图不存在。
SMC_RET_INVALID_DINAME	数据项无效。
SMC_RET_INVALID_DISTAT	数据项统计类型无效。
SMC_RET_INVALID_PARAMETER	参数无效。

另请参见 smc_refresh_ex、 smc_get_dataitem_type

smc_get_row_count

说明

返回给定的视图在刷新后返回的行数。

语法

SMC_RETURN_CODE smc_get_row_count
(SMC_CONNECT_ID　*clientId*,
SMC_VIEW_ID　*viewId*,
SMC_SIZETP　*rowCountHandle*)

参数

clientId
　标识连接。

viewId
　标识针对连接创建的视图。

rowCountHandle
　指向 Monitor Client Library 将视图中的行数写入到的变量的指针。

返回值

返回值	含义
SMC_RET_SUCCESS	函数成功。
SMC_RET_FAILURE	函数失败。
SMC_RET_INVALID_CONNECT	不存在连接。

示例

下例假定：

- 已创建一个连接并具有一个 *clientId*。
- 已针对该连接创建一个视图，且具有一个 *viewId*。
- 已成功执行了一次刷新调用。

```
SMC_SIZET rowCount;
 SMC_SIZETP rowCountHandle = &rowCount;
if (smc_get_row_count(clientId,
          nviewId,
          rowCountHandle) != SMC_RET_SUCCESS)
 {
   printf("smc_get_row_count failed\n");
   /* do some cleanup */
 }
```

用法

数据的第一行用行号 0 进行索引，第二行用 1 进行索引，依此类推。

有效的服务器模式

模式	可用性
SMC_SERVER_M_LIVE	可用
SMC_SERVER_M_HISTORICAL	可用 （在回放期间）

错误

错误	含义
SMC_RET_INVALID_VIEWID	视图不存在。

另请参见

smc_refresh_ex、 smc_get_dataitem_value

smc_get_version_string

说明

返回 Monitor Client Library 版本号。

语法

SMC_RETURN_CODE smc_get_version_string
(SMC_CHARPP *versionBuffer*)

参数

versionBuffer

包含版本字符串的返回值。

返回值

返回值	含义
SMC_RET_SUCCESS	函数成功。
SMC_RET_FAILURE	函数失败。

示例

```
SMC_CHARP versionBufferHandle;
if (smc_get_version_string(&versionBufferHandle)
                                != SMC_RET_SUCCESS)
 {
       printf("smc_get_version_string failed\n");
       /* do some cleanup */
 }
 printf("%s\n",versionBufferHandle);
free(versionBufferHandle);
```

用法

- Monitor Client Library 为此字符串分配内存。主调应用程序必须使用 free() 释放此内存。
- 此函数不需要连接。

smc_initiate_playback

说明

结束 Historical Server 上回放会话的视图定义，并准备开始回放。

语法

SMC_RETURN_CODE smc_initiate_playback
(SMC_CONNECT_ID　　*clientId*)

参数

clientId

标识连接。

返回值

返回值	含义
SMC_RET_SUCCESS	函数成功。
SMC_RET_FAILURE	函数失败。
SMC_RET_INVALID_CONNECT	不存在连接。

示例

下面的示例假定：

- 已创建了到 Historical Server 的一个连接，并具有一个 *clientId*。
- 已成功执行了 smc_create_playback_session。
- 已针对该连接创建了至少一个视图。

```
if (smc_initiate_playback(clientId) !=
   SMC_RET_SUCCESS)
 {
   printf("smc_initiate_playback failed\n");
   /* do some cleanup */
 }
```

用法

- 回放会话的数据由在调用 smc_create_playback_session 后、调用 smc_initiate_playback 前对 smc_create_view 进行的调用加以定义。
- 如果此回放会话定义用来从回放中创建一个新会话 （即如果调用 smc_create_playback_session 时 *playbackTarget* 为 SMC_HS_TARGET_FILE），那么 smc_initiate_playback 将创建该新会话。应用程序然后必须调用 smc_terminate_playback 来结束回放会话。
- 如果回放会话定义用来将数据回放到应用程序中 （即调用 smc_create_playback_session 时 *playbackTarget* 参数为 SMC_HS_TARGET_CLIENT），那么应用程序调用 smc_refresh_ex 检索每个回放采样，并调用 smc_terminate_playback 结束回放会话。
- 在成功调用 smc_terminate_playback 后，可利用 Historical Server 连接定义其它回放会话或创建记录会话。

有效的服务器模式

模式	可用性
SMC_SERVER_M_LIVE	不可用
SMC_SERVER_M_HISTORICAL	可用

错误

错误	含义
SMC_RET_INVALID_SVR_MODE	服务器模式无效。
SMC_RET_INTERNAL_ERROR	内部错误。

另请参见 smc_create_view、 smc_create_playback_session、 smc_refresh_ex、 smc_terminate_playback

smc_initiate_recording

说明 完成针对 Historical Server （即仅有 SMC_SERVER_M_HISTORICAL 连接）的记录会话的定义。

语法 SMC_RETURN_CODE smc_initiate_recording (SMC_CONNECT_ID *clientId*)

参数 *clientId*

标识连接。

返回值

返回值	含义
SMC_RET_SUCCESS	函数成功。
SMC_RET_FAILURE	函数失败。
SMC_RET_INVALID_CONNECT	不存在连接。

示例 下例假定：

- 已创建了到 Historical Server 的一个连接，并具有一个 *clientId*。
- 已成功执行了 smc_create_recording_session。
- 已针对该连接创建了至少一个视图。

```
if (smc_initiate_recording(clientId) !=
   SMC_RET_SUCCESS)
 {
   printf("smc_initiate_recording failed\n");
   /* do some cleanup */
 }
```

用法

- 记录会话的数据由在调用 smc_create_recording_session 后、调用 smc_initiate_recording 前对 smc_create_view 和 smc_create_filter 进行的调用加以定义。
- 在成功调用 smc_initiate_recording 后，可利用 Historical Server 连接定义其它记录会话或创建回放会话。

有效的服务器模式

模式	可用性
SMC_SERVER_M_LIVE	不可用
SMC_SERVER_M_HISTORICAL	可用

错误

错误	含义
SMC_RET_INVALID_SVR_MODE	服务器模式无效。
SMC_RET_INTERNAL_ERROR	内部错误。

另请参见　smc_create_alarm_ex、 smc_create_filter、 smc_create_view、 smc_create_recording_session、 smc_terminate_recording_session

smc_refresh_ex

说明　获得针对某一连接的所有视图的数据采样。

语法

```
SMC_RETURN_CODE smc_refresh_ex
(SMC_CONNECT_ID  clientId,
SMC_SIZET        step)
```

参数

clientId
标识连接。

step
在 Historical Server 连接上的回放期间，允许向前跳过指定数目的采样。通常，回放的 *step* 值是正 1，检索下一个采样 （*step* 不允许使用负值）。

不适用于活动连接，对活动连接应使用 SMC_UNUSED。

返回值

返回值	含义
SMC_RET_SUCCESS	函数成功。
SMC_RET_FAILURE	函数失败。
SMC_RET_INVALID_CONNECT	不存在连接。

示例

下面的示例假定：

- 已创建一个连接并具有一个 *clientId*。
- 已针对该连接创建了至少一个视图。

```
if (smc_refresh_ex(clientId,SMC_UNUSED)
    != SMC_RET_SUCCESS)
 {
   printf("smc_refresh_ex failed\n");
   /* do some cleanup */
 }
```

用法

- 在回放会话中，必须在调用 smc_initiate_playback 之后调用 smc_refresh_ex。
- 如果在有人创建数据库的同时，您试图去刷新一个视图，刷新可能会失败。
- 如果 Adaptive Server 上的一个或多个数据库处于单用户模式，刷新视图可能会失败。

有效的服务器模式

模式	可用性
SMC_SERVER_M_LIVE	可用
SMC_SERVER_M_HISTORICAL	可用 （对于回放）

错误

错误	含义
SMC_RET_INVALID_API_FUNCTION	在程序中无效地使用了旧函数和替换函数。
SMC_RET_INVALID_SVR_MODE	服务器模式无效。

另请参见

smc_connect_ex

smc_terminate_playback

说明

结束 Historical Server 上的回放会话。

语法

SMC_RETURN_CODE smc_terminate_playback
(SMC_CONNECT_ID *clientId*)

参数

clientId

标识连接。

返回值

返回值	含义
SMC_RET_SUCCESS	函数成功。
SMC_RET_FAILURE	函数失败。
SMC_RET_INVALID_CONNECT	不存在连接。

示例

下面的示例假定：

- 已创建了到 Historical Server 的一个连接，并具有一个 *clientId*。
- 已成功执行了 smc_create_playback_session。
- 已针对该连接创建了至少一个视图。
- 已成功执行了 smc_initiate_playback。

```
if (smc_terminate_playback(clientId)
    != SMC_RET_SUCCESS)
 {
    printf("smc_terminate_playback failed\n");
    /* do some cleanup */
 }
```

用法

- 在成功调用 smc_terminate_playback 后，可利用 Historical Server 连接创建其它回放会话或定义记录会话。

有效的服务器模式

模式	可用性
SMC_SERVER_M_LIVE	不可用
SMC_SERVER_M_HISTORICAL	可用

错误

错误	含义
SMC_RET_INVALID_SVR_MODE	服务器模式无效。
SMC_RET_INTERNAL_ERROR	内部错误。

另请参见

smc_create_playback_session、 smc_initiate_playback

smc_terminate_recording

说明

取消 Historical Server 连接上的记录会话。

语法

```
SMC_RETURN_CODE smc_terminate_playback(
SMC_CONNECT_ID clientId,
SMC_SESSION_ID sessionId
SMC_HS_SESS_DELETE_OPT  deleteOption,
)
```

参数

clientId

标识 Monitor 连接。

sessionId

标识要取消的记录会话。

deleteOption

指定 Historical Server 是否应删除与会话关联的数据文件 （如果有的话）。可选值是 SMC_HS_DELETE_FILES 和 SMC_HS_RETAIN_FILES。

如果会话还未启动或还未开始记录，将忽略此参数。

返回值

返回值	含义
SMC_RET_SUCCESS	函数成功。
SMC_RET_FAILURE	函数失败。
SMC_RET_INVALID_CONNECT	Monitor 连接不存在。

示例

下面的示例假定：

- 已创建了到 Historical Server 的一个连接，并具有一个 *clientId*。
- 已成功执行了 smc_create_recording_session，并拥有一个 *sessionId*。

```
if (smc_terminate_recording(
       clientId,
       sessionId,
   SMC_HS_DELETE_FILES)
   != SMC_RET_SUCCESS)
 {
   printf("smc_terminate_recording failed\n");
   /* do some cleanup */
 }
```

用法

- 如果记录会话已经启动， smc_terminate_recording 将取消该会话。如果已安排了记录会话但还未实际开始记录，那么 smc_terminate_recording 将取消对该会话做出的计划安排。如果该会话已实际开始记录，那么 smc_terminate_recording 将使会话提前结束，即在预定的结束时间之前结束。
- 如果记录会话尚未启动， smc_terminate_recording 将取消该记录会话的定义。在成功调用 smc_terminate_recording 后，可利用 HISTORICAL 连接创建其它记录会话或定义回放会话。

有效的服务器模式

模式	可用性
SMC_SERVER_M_LIVE	不可用
SMC_SERVER_M_HISTORICAL	可用

错误

错误	含义
SMC_RET_INVALID_SVR_MODE	服务器模式无效。
SMC_RET_INTERNAL_ERROR	内部错误。

另请参见

smc_create_recording_session、 smc_initiate_recording

第 4 章

构建 Monitor Client Library 应用程序

本章包含有关构建 Monitor Client Library 应用程序的信息。

本章介绍了构建 Monitor Client Library 应用程序所需的步骤，包括：

- 编译
- 链接
- 运行

Monitor Client Library 提供了两个示例程序：

- *testmon*，用于从 Monitor Server 获取数据
- *testhist*，用于创建 Historical Server 记录会话，并将数据写入文件

可使用这些示例程序提供的构建过程，作为其它应用程序的模板。将分别讨论用于 UNIX 和 Windows 平台的示例程序。

注释 以下说明假定 Monitor Client Library 安装在 Sybase 根目录下，并且已将 SYBASE 环境变量设为此根目录。

在 UNIX 平台上构建

本节说明如何编译、链接、运行及构建用于 UNIX 平台的示例应用程序。

编译应用程序

每个使用 Monitor Client Library 的源文件必须包括下列行：

```
#include "mcpublic.h"
```

缺省情况下， Monitor Client Library 的头文件安装在 SYBASE 环境变量指定的目录下的 *OCS-15_0/include* 目录中。

编译时需要的 Open Client 头文件也安装在此目录下。请在编译命令行中包括此目录。例如，可输入：

```
cc -I$SYBASE/OCS-15_0/include myprog.c
```

如果头文件已安装在非缺省的目录下，请在编译命令行中替代这些目录。

链接应用程序

Monitor Client Library 安装在由 SYBASE 环境变量指定的目录下的 *OCS-15_0/lib* 目录中。此外，链接 Monitor Client Library 时需要的 Open Client 库安装在 *OCS-15_0/lib* 目录中。要查找必须与应用程序链接的库名，请参见随示例提供的 *make* 文件。

运行应用程序

若要运行 Monitor Client Library 应用程序，请将 SYBASE 环境变量设置为 Open Client 的安装目录，该目录包含 *locales*、*charsets* 和 *lib* 目录。这些目录在安装 Monitor Client Library 期间装载。

注释 在运行 Monitor Client Library 应用程序之前，必须配置好 Adaptive Server 和 Monitor Server，并在网络上运行它们。

构建示例应用程序

缺省情况下，示例程序和构建它们的过程安装在 *$SYBASE/OCS-15_0/sample/monclt* 目录中。构建过程的两个版本是：

- *Makefile*，使用本机 ANSI 编译器和链接器
- *Makefile_gcc*，使用 GNU C 编译器和链接器

要构建并运行示例程序，使用如下步骤：

1 如果 *interfaces* 文件中没有要与示例程序一起使用的 Adaptive Server、 Monitor Server 和 Historical Server 条目，则添加这些条目。您可以使用 monclt/bin/dsedit 编辑 *interfaces* 文件。

2 从 *monclt/sample* 目录将示例文件复制到另一个目录下，保留原始示例程序，用作将来参考，并且您可以直接编辑复制的副本。

3 如果还没有在复制的目录下，请将目录更改为包含示例程序副本的目录。

4 编辑 *example.h* 文件，提供以下各项的名称：

   - Adaptive Server
   - Monitor Server
   - Historical Server
   - Adaptive Server 上的登录名
   - 口令
   - *interfaces* 文件位置

   如果使用的是 SYBASE 环境变量指定的目录下的缺省 *interfaces* 文件，则可以用缺省的空字符串 ("") 来表示 *interfaces* 文件名。如果没有使用缺省的 *interfaces* 文件，请指定 *interfaces* 文件的完整路径名。

5 将 MONCLTLIBDIR 环境变量设置为 Monitor Client Library 的根安装目录，缺省为 Sybase 根安装目录下的 *OCS-15_0* 目录：

```
setenv MONCLTLIBDIR $SYBASE/OCS-15_0
```

6 可编辑 *make* 文件，更改 SYBASE 变量的值，使其指向其它 Sybase 根目录。缺省情况下，它指向 *$MONCLTLIBDIR*。

7 使用 make 实用程序构建测试程序。

如果使用本机的 UNIX make 实用程序，请输入：

```
make all
```

如果使用 GNU 编译器，请输入：

```
make -f Makefile_gcc
```

8 运行示例程序。

要运行从 Monitor Server 检索并显示活动数据的程序，请输入：

```
./testmon
```

要运行使用 Historical Server 创建记录会话的程序，请输入：

```
./testhist
```

在 Windows 平台上构建应用程序

本节说明如何在 Windows 平台上编译、链接、运行及构建示例应用程序。

编译应用程序

在 Windows 平台上编译 Monitor Client Library 应用程序：

1 将下面的命令行包含在使用 Monitor Client Library 的每个源文件中：

```
#include "mcpublic.h"
```

2 将包含 Monitor Client Library 和 Open Client 头文件的目录的路径包括在目录列表中（有时称为 Include 路径），C 编译器的预处理程序将在该路径中查找所需的头文件。缺省情况下， Monitor Client Library 和 Open Client 的头文件安装在 *%SYBASE%\OCS-15_0\inlcude* 目录中。

3 设置编译器的预处理程序选项，定义 *_WIN* 和 *WIN32* 预处理程序宏。

4 将代码生成选项设置为使用 *__cdecl* 调用约定。

注释 要使用非缺省的调用约定，必须在每个使用它的回调函数中声明它。

链接应用程序

Monitor Client Library 包含在 *smcapi32.lib* 文件中，该文件安装在 *%SYBASE%\OCS-15_0\lib* 目录下。

可在链接器的库列表中指定您的应用程序使用的库或 *smcapi32.lib* 文件名的完整路径名。但如果只包含文件名，则必须在目录列表中包括 *C:\SYBASE\LIB*，链接器会在其中查找库。

运行应用程序

运行 Monitor Client Library 应用程序所需的软件，请参见 Adaptive Server Enterprise Monitor 的发行公告。

定义 SYBASE 环境变量，指出 Sybase 客户端软件的安装目录。此目录中的 *ini* 目录必须包含 *sql.ini* 文件。使用 SQLEDIT 实用程序设置此文件，将应用程序使用的所有 Adaptive Server、 Monitor Server 以及 Historical Server （可选）的名称都包含在此文件中。

注释　在运行 Monitor Client Library 应用程序之前，必须配置好 Adaptive Server 和 Monitor Server，并在网络上运行它们。

构建示例应用程序

示例程序和构建它们的过程安装在 *%SYBASE%\OCS-15_0\SAMPLE\MONCLT\TESTMON* 和 *%SYBASE%\OCS-15_0\SAMPLE\MONCLT\TESTHIST* 目录下。

对于每个示例程序，都有一个项目 (*.mak*) 文件。对于要使用 Microsoft Visual C/C++ 版本 4.0 构建的应用程序和要作为主控台程序运行在 Windows NT 或 Windows 95 下的应用程序，两个项目文件分别是 *TESTMO32.MAK* 和 *TESTHI32.MAK*。

要构建并运行示例程序，使用如下步骤：

1　修改 PATH 环境变量，使其包含 Sybase DLL 安装到的 *C:\SYBASE\DLL* 目录。

2　如果尚未完成该操作，请将 SYBASE 环境变量设置为 Sybase *\SYBASE* 根安装目录。

3 如果在 *sql.ini* 文件中没有合适的服务器名，请将要使用的 Adaptive Server 安装、 Monitor Server 和 Historical Server 条目添加到 *C:\SYBASE\INI\SQL.INI* 文件中。

4 编辑 *%SYBASE%\OCS-15_0\sample\monclt\testmon\example.h* 和 *%SYBASE%\OCS-15_0\sample\monclt\testhist\example.h* 文件，以提供 Adaptive Server、 Monitor Server、 Historical Server （仅适用于 *TESTHIST*）的名称， Adaptive Server 上的登录名及口令。

5 打开要构建的示例应用程序的项目 (*.mak*) 文件。

- 要使用测试到 Monitor Server 的活动连接的程序，请输入：

```
%SYBASE%\OCS-15_0\sample\monclt\testhist\testhi32.mac
```

- 要使用测试 Historical Server 的程序，请输入：

```
%SYBASE%\OCS-15_0\sample\monclt\testhist\testhi32.mak
```

6 如果 Monitor Client Library 安装在 *\SYBASE* 之外的其它目录下：

- 修改编译器预处理程序选项以在目录列表中包括安装目录下的 *INCLUDE* 子目录，以代替缺省的 *\SYBASE\INCLUDE* 目录， C 编译器的预处理程序会在此子目录中查找头文件。
- 编辑要用于应用程序的链接器的库列表，使其指定库的完整路径名，而不是缺省目录路径名 *\SYBASE\LIB\SMCAPI32.LIB*。

7 构建项目。

8 运行应用程序。

要在 Windows NT 或 Windows 95 下运行应用程序，请从命令提示窗口中输入可执行程序名。例如：

```
%SYBASE%\OCS-15_0\SAMPLE\MONCLT\TESTMON\WinDebug\TESTMO32
```

第 5 章

Monitor Client Library 配置说明

本章介绍了 Monitor Client Library 的安装和配置过程。

装载 Monitor Client Library

要将 Monitor Client Library 文件从分发介质上移动到您的计算机中，请使用 InstallShield。此实用程序允许在一个 InstallShield 会话中，将您订购的所有产品装载到一台计算机中；或者运行单独的 InstallShield 会话，将软件分配到不同的授权计算机中。

使用 InstallShield

如果还没有装载，请遵循安装指南中的说明，将 Monitor Client Library 装载到计算机上。

软件装载后，返回到本章，完成 Monitor Client Library 的安装和配置。

装载的结果

InstallShield 实用程序将 Monitor Client Library 软件放入在安装过程中为 InstallShield 指定的装载目录中。缺省的装载目录是 $SYBASE 目录。

装载目录包含 Monitor Client Library 的所有软件和其它文件，包括 Monitor Client Library 正确版本级别的 *locales* 和 *charsets* 子目录。

配置登录帐户和权限

要执行本章中介绍的任务，必须使用 “sybase”帐户或对装载目录拥有读取、写入和搜索 （执行）权限的其它某个帐户进行登录。装载目录是指，在将 Monitor Client Library 软件装载到计算机中时为 InstallShield 提供的目录名。缺省的装载目录是 $SYBASE 目录。

修改 *interfaces* 文件

Monitor Client Library 应用程序运行之前，它必须有访问 *interfaces* 文件的权限，该文件包含 Adaptive Server Enterprise Monitor 的条目。*interfaces* 文件可存在于本地或远程计算机中，只要 Monitor Client Library 应用程序能访问包含 *interfaces* 文件的文件系统。

如果在将要运行 Monitor Client Library 应用程序的计算机中不存在 *interfaces* 文件，同时又不能远程访问 *interfaces* 文件，则必须创建一个此类文件。

Monitor Client Library 应用程序访问的 *interfaces* 文件必须包含如下服务器的对应条目：

- 被监控的 Adaptive Server 安装
- Monitor Viewer 正在使用的 Monitor Server
- Monitor Historical Server（可选，如果有被使用的 Monitor Historical Server）

Monitor Client Library 应用程序要访问的 *interfaces* 文件中添加的条目必须与服务器计算机上的 *interfaces* 文件中已存在的服务器条目匹配。这些条目定义服务器名、它们的主机名和以及端口号。必须在客户端计算机上使用相同的值。咨询安装 Monitor Server 和 Monitor Historical Server 的人员，获取服务器的条目。

添加到客户端 *interfaces* 文件中的条目的一般格式为：

```
sql_server_name
query entry
master entry
monitor_server_name
query entry
master entry
historical_server_name
query entry
master entry
```

可以使用 dsedit 实用程序或文本编辑器将这些条目添加到 *interfaces* 文件中。

如果使用文本编辑器更新 *interfaces* 文件，条目必须符合如下规则：

- 条目不能包含空行。
- *server_name* 行必须从 *interfaces* 文件的第一列开始。
- query 和 master 条目前必须有一个制表符。必须使用 Tab 键缩进 query 和 master 行，不要使用空格键来缩进这两行。

有关编辑 *interfaces* 文件的信息、关于 *interfaces* 文件格式的细节，以及有关 *interfaces* 文件条目内参数的详细信息，请参见适用于您平台的*配置 Adaptive Server Enterprise*。

设置用户环境

启动时， Monitor Client Library 应用程序必须知道下列信息：

- *locales* 和 *charsets* 目录的正确版本
- *interfaces* 文件

SYBASE 环境变量定义 *locales* 和 *charsets* 目录的位置。SYBASE 变量还定义 *interfaces* 文件的缺省位置，但 Monitor Client Library 应用程序可能需要覆盖该缺省位置。

设置 *SYBASE* 环境变量

当用户启动 Monitor Client Library 应用程序时， SYBASE 环境变量指向的目录必须包含 *locales* 和 *charsets* 目录的正确版本。因此，用户必须设置 SYBASE 环境变量，使之指向装载目录中的 *monclt* 子目录（InstallShield 用来安装 Monitor Client Library 软件的目录）。

覆盖 *interfaces* 文件的缺省位置

interfaces 文件的缺省位置是 SYBASE 环境变量指向的目录。既然 SYBASE 环境变量必须指向装载目录，那么用户运行 Monitor Client Library 应用程序时，下面的其中一个条件也必须成立：

- *interfaces* 文件必须位于装载目录内，或者
- Monitor Client Library 应用程序代码必须覆盖 *interfaces* 文件的缺省位置。

  若要覆盖缺省位置， Monitor Client Library 应用程序必须调用 smc_connect 函数，在 *interfaceFile* 参数中显式指定一个值。多数情况下，可通过下列方式获取 *interfaceFile* 参数的值：在用户启动时获得 （作为命令行参数），从 X 资源文件获得，或从交互式对话框获得。

有关 smc_connect 函数的详细信息，请参见 《Adaptive Server Enterprise Monitor Client Library 程序员指南》。

使用 Monitor Client Library

完成安装和设置用户环境后，可构建并运行提供的示例程序。有关示例程序的详细信息，请参见 《Adaptive Server Enterprise Monitor Client Library 程序员指南》。

如果尚未执行上述操作，请阅读适合您所用平台的 《Adaptive Server Enterprise Monitor Client Library 发行公告》。

注释

- 在运行 Monitor Client Library 应用程序之前，必须配置好 Adaptive Server 和 Monitor Server，并在网络上运行它们。
- 为获得最快的响应速度，Sybase 建议不要在运行 Adaptive Server 和 Monitor Server 的计算机上运行 Monitor Client 应用程序。

附 录 A

视图示例

本附录包括一些视图示例。这些视图也显示在随 Historical Server 安装的示例视图文件中。

您可能会发现某些视图恰好收集了您感兴趣的信息，而另外一些视图可以作为建立您所需视图的模板。

一些示例视图只是在累计数据的时间间隔上有所差别 （最近采样间隔持续时间或整个会话）。其它视图可能包括类似的数据项，但顺序不同。数据项在视图中的显示顺序很重要，因为数据是按照键字段排序的。第一个键字段显示在视图的定义中并充当主排序键，第二个键字段是次要排序键，依此类推。

```
#include mcpublic.h

SMC_VOID
ErrorCallback(
SMC_SIZET    id,
SMC_SIZET    error_number,
SMC_SIZET    severity,
SMC_SIZET    map_severity,
SMC_SIZET    source,
SMC_CCHARP   error_msg,
SMC_SIZET    state);

SMC_VOID
RefreshCallback(
SMC_SIZET   id,
SMC_VOIDP   user_msg,
SMC_CHARP   msg);
SMC_CHARP
SMC_DATAITEM_NAME   value);

SMC_CHARP
LookupDataItemStat(
SMC_DATAITEM_STATTYPE   value);

SMC_CHARP
LookupLockResult(
SMC_LOCK_RESULT   value);

SMC_CHARP
LookupLockResultSummary(
SMC_LOCK_RESULT_SUMMARY   value);

SMC_CHARP
LookupLockStatus(
SMC_LOCK_STATUS   value);
```

```
SMC_CHARP
LookupLockType(
SMC_LOCK_TYPE   value);

SMC_CHARP
LookupObjectType(
SMC_OBJ_TYPE   value);

SMC_CHARP
LookupProcessState(
SMC_PROCESS_STATE   value);
SMC_INT
main(
SMC_INT     argc,
SMC_CHARP   argv[])
{
```

高速缓存性能摘要

本视图显示 Adaptive Server 高速缓存在最近一次采样间隔期间的总体效率。它显示由 Adaptive Server 数据高速缓存满足的数据页读取数所占的百分比，以及由 Adaptive Server 过程高速缓存满足的过程执行请求数所占的百分比。

```
SMC_SIZET cache_perf_sum_count = 2;
SMC_DATAITEM_STRUCT cache_perf_sum_view[] = {
{ SMC_NAME_PAGE_HIT_PCT,           SMC_STAT_VALUE_SAMPLE },
{ SMC_NAME_STP_HIT_PCT,            SMC_STAT_VALUE_SAMPLE }
};
```

当前语句摘要

本视图显示 Adaptive Server 当前执行的语句的有关信息，不考虑它是存储过程的一部分还是批处理文本的一部分。如果您要尝试确定某个应用程序在其执行过程中的某个特定时刻在执行何种操作，可以使用此类视图。

```
SMC_SIZET cur_stmt_act_count = 11;
SMC_DATAITEM_STRUCT cur_stmt_act_view[] = {
{ SMC_NAME_SPID,                    SMC_STAT_VALUE_SAMPLE },
{ SMC_NAME_CUR_STMT_ACT_STP_DB_ID,  SMC_STAT_VALUE_SAMPLE },
{ SMC_NAME_CUR_STMT_ACT_STP_DB_NAME, SMC_STAT_VALUE_SAMPLE },
{ SMC_NAME_CUR_STMT_ACT_STP_ID,     SMC_STAT_VALUE_SAMPLE },
{ SMC_NAME_CUR_STMT_ACT_STP_NAME,   SMC_STAT_VALUE_SAMPLE },
{ SMC_NAME_CUR_STMT_ACT_STP_TEXT,   SMC_STAT_VALUE_SAMPLE },
{ SMC_NAME_CUR_STMT_BATCH_ID,       SMC_STAT_VALUE_SAMPLE },
{ SMC_NAME_CUR_STMT_CONTEXT_ID,     SMC_STAT_VALUE_SAMPLE },
{ SMC_NAME_CUR_STMT_NUM,            SMC_STAT_VALUE_SAMPLE },
{ SMC_NAME_CUR_STMT_QUERY_PLAN_TEXT, SMC_STAT_VALUE_SAMPLE },
{ SMC_NAME_CUR_STMT_START_TIME,     SMC_STAT_VALUE_SAMPLE },
};
```

数据库对象锁状态

本视图显示到最近一次采样间隔结束时 Adaptive Server 进程所持有或请求的数据库对象锁的状态。每个锁由以下内容标识：

- 被锁定对象的名称和 ID
- 包含该对象的数据库的名称和 ID
- 锁应用到的页号 （如果是页锁）

与锁关联的每个 Adaptive Server 进程也由它的登录名、进程 ID 和内核进程 ID 加以标识。将同时显示锁的类型、锁的当前状态以及这是否是一个请求锁的指示。

如果该进程正在请求该锁，将显示该进程为获得该锁已经等待的时间和已经持有该锁的进程的进程 ID。如果该进程已经持有该锁，将显示等待获得该锁的其它进程数量。

```
SMC_SIZET object_lock_status_count = 14;
SMC_DATAITEM_STRUCT object_lock_status_view[] = {
{ SMC_NAME_DB_ID,                    SMC_STAT_VALUE_SAMPLE },
{ SMC_NAME_DB_NAME,                  SMC_STAT_VALUE_SAMPLE },
{ SMC_NAME_OBJ_ID,                   SMC_STAT_VALUE_SAMPLE },
{ SMC_NAME_OBJ_NAME,                 SMC_STAT_VALUE_SAMPLE },
{ SMC_NAME_PAGE_NUM,                 SMC_STAT_VALUE_SAMPLE },
{ SMC_NAME_LOGIN_NAME,               SMC_STAT_VALUE_SAMPLE },
{ SMC_NAME_SPID,                     SMC_STAT_VALUE_SAMPLE },
{ SMC_NAME_KPID,                     SMC_STAT_VALUE_SAMPLE },
{ SMC_NAME_LOCK_TYPE,                SMC_STAT_VALUE_SAMPLE },
{ SMC_NAME_LOCK_STATUS,              SMC_STAT_VALUE_SAMPLE },
{ SMC_NAME_DEMAND_LOCK,              SMC_STAT_VALUE_SAMPLE },
{ SMC_NAME_TIME_WAITED_ON_LOCK,      SMC_STAT_VALUE_SAMPLE },
{ SMC_NAME_BLOCKING_SPID,            SMC_STAT_VALUE_SAMPLE },
{ SMC_NAME_LOCKS_BEING_BLOCKED_CNT,  SMC_STAT_VALUE_SAMPLE }
};
```

数据库对象页 I/O

本视图显示 Adaptive Server 数据库中的对象及其关联的页 I/O 数。它显示 Adaptive Server 数据库的名称和 ID，以及每个数据库中的对象名和 ID。对于每个对象，本视图显示最近一次采样间隔期间以及会话期间发生的与其关联的逻辑读取次数、物理读取次数和页写入次数。

```
SMC_SIZET object_page_io_count = 10;
SMC_DATAITEM_STRUCT object_page_io_view[] = {
{ SMC_NAME_DB_ID,                SMC_STAT_VALUE_SAMPLE },
{ SMC_NAME_DB_NAME,              SMC_STAT_VALUE_SAMPLE },
{ SMC_NAME_OBJ_ID,               SMC_STAT_VALUE_SAMPLE },
{ SMC_NAME_OBJ_NAME,             SMC_STAT_VALUE_SAMPLE },
{ SMC_NAME_PAGE_LOGICAL_READ,    SMC_STAT_VALUE_SAMPLE },
{ SMC_NAME_PAGE_PHYSICAL_READ,   SMC_STAT_VALUE_SAMPLE },
{ SMC_NAME_PAGE_WRITE,           SMC_STAT_VALUE_SAMPLE },
{ SMC_NAME_PAGE_LOGICAL_READ,    SMC_STAT_VALUE_SESSION },
{ SMC_NAME_PAGE_PHYSICAL_READ,   SMC_STAT_VALUE_SESSION },
{ SMC_NAME_PAGE_WRITE,           SMC_STAT_VALUE_SESSION }
};
```

各个高速缓存的数据高速缓存活动

本视图显示关于各个数据高速缓存的性能的信息。

对于每个命名高速缓存（包括在 Adaptive Server 中配置的缺省数据高速缓存），本视图收集该高速缓存的名称以及绑定到该高速缓存的对象自记录会话开始以来由该高速缓存满足的页读取数所占的百分比。

本视图还显示：

- 该高速缓存的空间利用效率
- 自该会话开始以来，为获得该高速缓存的螺旋锁而被迫等待的时间所占的百分比
- 该会话期间的高速缓存命中数和未命中数

```
SMC_SIZET data_cache_activity_count = 7;
SMC_DATAITEM_STRUCT data_cache_activity__view[] = {
{ SMC_NAME_DATA_CACHE_NAME,          SMC_STAT_VALUE_SAMPLE },
{ SMC_NAME_DATA_CACHE_ID,            SMC_STAT_VALUE_SAMPLE },
{ SMC_NAME_DATA_CACHE_HIT_PCT,       SMC_STAT_VALUE_SESSION },
{ SMC_NAME_DATA_CACHE_EFFICIENCY,    SMC_STAT_VALUE_SESSION },
{ SMC_NAME_DATA_CACHE_CONTENTION,    SMC_STAT_RATE_SESSION },
{ SMC_NAME_DATA_CACHE_HIT,           SMC_STAT_VALUE_SESSION },
{ SMC_NAME_DATA_CACHE_MISS,          SMC_STAT_RATE_SESSION }
};
```

会话的数据高速缓存统计信息

本视图显示自会话开始以来 Adaptive Server 数据高速缓存的效率。它显示：

- 该会话期间由高速缓存满足的页读取请求数所占的百分比
- 该会话期间发生的逻辑读取次数、物理读取次数和页写入次数
- 该会话期间发生的逻辑读取、物理读取和页写入操作的速率

```
SMC_SIZET session_page_cache_stats_count = 7;
SMC_DATAITEM_STRUCT session_page_cache_stats_view[] = {
{ SMC_NAME_PAGE_HIT_PCT,          SMC_STAT_VALUE_SESSION },
{ SMC_NAME_PAGE_LOGICAL_READ,     SMC_STAT_VALUE_SESSION },
{ SMC_NAME_PAGE_LOGICAL_READ,     SMC_STAT_RATE_SESSION },
{ SMC_NAME_PAGE_PHYSICAL_READ,    SMC_STAT_VALUE_SESSION },
{ SMC_NAME_PAGE_PHYSICAL_READ,    SMC_STAT_RATE_SESSION },
{ SMC_NAME_PAGE_WRITE,            SMC_STAT_VALUE_SESSION },
{ SMC_NAME_PAGE_WRITE,            SMC_STAT_RATE_SESSION }
};
```

采样间隔期间的数据高速缓存统计信息

本视图显示最近一次采样间隔期间 Adaptive Server 的数据高速缓存的效率。它显示：

- 最近一次采样间隔期间由相应高速缓存满足的页读取请求数所占的百分比
- 最近一次采样间隔期间发生的逻辑读取次数、物理读取次数和页写入次数
- 最近一次采样间隔期间发生的逻辑读取、物理读取和页写入操作的速率

```
SMC_SIZET sample_page_cache_stats_count = 7;
SMC_DATAITEM_STRUCT sample_page_cache_stats_view[] = {
{ SMC_NAME_PAGE_HIT_PCT,          SMC_STAT_VALUE_SAMPLE },
{ SMC_NAME_PAGE_LOGICAL_READ,     SMC_STAT_VALUE_SAMPLE },
{ SMC_NAME_PAGE_LOGICAL_READ,     SMC_STAT_RATE_SAMPLE },
{ SMC_NAME_PAGE_PHYSICAL_READ,    SMC_STAT_VALUE_SAMPLE },
{ SMC_NAME_PAGE_PHYSICAL_READ,    SMC_STAT_RATE_SAMPLE },
{ SMC_NAME_PAGE_WRITE,            SMC_STAT_VALUE_SAMPLE },
{ SMC_NAME_PAGE_WRITE,            SMC_STAT_RATE_SAMPLE }
};
```

会话的设备 I/O

本视图显示自会话开始以来在 Adaptive Server 数据库设备上发生的 I/O 活动。它用名称标识每个设备。设备 I/O 级别有两种表示方法：一种是以自会话开始以来发生的设备 I/O、读取和写入操作总计数表示；另一种是以总速率表示，即自会话开始以来每秒发生的 I/O、读取和写入操作总数。

```
SMC_SIZET session_device_io_count = 7;
SMC_DATAITEM_STRUCT session_device_io_view[] = {
{ SMC_NAME_DEV_NAME,       SMC_STAT_VALUE_SAMPLE },
{ SMC_NAME_DEV_READ,       SMC_STAT_VALUE_SESSION },
{ SMC_NAME_DEV_WRITE,      SMC_STAT_VALUE_SESSION },
{ SMC_NAME_DEV_IO,         SMC_STAT_VALUE_SESSION },
{ SMC_NAME_DEV_READ,       SMC_STAT_RATE_SESSION },
{ SMC_NAME_DEV_WRITE,      SMC_STAT_RATE_SESSION },
{ SMC_NAME_DEV_IO,         SMC_STAT_RATE_SESSION }
};
```

采样间隔期间的设备 I/O

本视图显示在最近一次采样间隔期间 Adaptive Server 数据库设备上发生的 I/O 活动。它用名称标识每个设备。设备 I/O 级别有两种表示方法：一种是以最近一次采样间隔期间发生的设备 I/O、读取和写入操作总计数表示，另一种是以速率表示，即在此采样间隔期间每秒发生的 I/O、读取和写入操作总数。

```
SMC_SIZET sample_device_io_count = 7;
SMC_DATAITEM_STRUCT sample_device_io_view[] = {
{ SMC_NAME_DEV_NAME,       SMC_STAT_VALUE_SAMPLE },
{ SMC_NAME_DEV_IO,         SMC_STAT_VALUE_SAMPLE },
{ SMC_NAME_DEV_READ,       SMC_STAT_VALUE_SAMPLE },
{ SMC_NAME_DEV_WRITE,      SMC_STAT_VALUE_SAMPLE },
{ SMC_NAME_DEV_IO,         SMC_STAT_RATE_SAMPLE },
{ SMC_NAME_DEV_READ,       SMC_STAT_RATE_SAMPLE },
{ SMC_NAME_DEV_WRITE,      SMC_STAT_RATE_SAMPLE }
};
```

设备 I/O 性能摘要

本视图显示自会话开始以来， Adaptive Server 对数据库设备执行的读取数和写入数。它显示：

- 自该会话开始以来，对数据库设备执行的读取和写入操作的总速率
- 该时间段内最活跃的数据库设备
- 对最活跃设备执行的读取和写入操作的速率

```
SMC_SIZET device_perf_sum_count = 3;
SMC_DATAITEM_STRUCT device_perf_sum_view[] = {
{ SMC_NAME_DEV_IO,                SMC_STAT_RATE_SESSION },
{ SMC_NAME_MOST_ACT_DEV_NAME,     SMC_STAT_VALUE_SESSION },
{ SMC_NAME_MOST_ACT_DEV_IO,       SMC_STAT_RATE_SESSION }
};
```

引擎活动

该视图显示最近一次采样间隔期间，每个活动 Adaptive Server 引擎的活动级别。本视图为每个引擎显示：

- 引擎占用 CPU 的时间占该采样间隔的百分比
- 锁请求数
- 该采样间隔期间引擎产生的逻辑页读取数、物理页读取数和页写入数

```
SMC_SIZET engine_activity_count = 6;
SMC_DATAITEM_STRUCT engine_activity_view[] = {
{ SMC_NAME_ENGINE_NUM,             SMC_STAT_VALUE_SAMPLE },
{ SMC_NAME_CPU_BUSY_PCT,           SMC_STAT_VALUE_SAMPLE },
{ SMC_NAME_LOCK_CNT,               SMC_STAT_VALUE_SAMPLE },
{ SMC_NAME_PAGE_LOGICAL_READ,      SMC_STAT_VALUE_SAMPLE },
{ SMC_NAME_PAGE_PHYSICAL_READ,     SMC_STAT_VALUE_SAMPLE },
{ SMC_NAME_PAGE_WRITE,             SMC_STAT_VALUE_SAMPLE }
};
```

锁性能摘要

该视图显示最近一次采样间隔期间，请求和授予的每种锁的总数。

```
SMC_SIZET lock_perf_sum_count = 3;
SMC_DATAITEM_STRUCT lock_perf_sum_view[] = {
{ SMC_NAME_LOCK_TYPE,              SMC_STAT_VALUE_SAMPLE },
{ SMC_NAME_LOCK_RESULT_SUMMARY,    SMC_STAT_VALUE_SAMPLE },
{ SMC_NAME_LOCK_CNT,               SMC_STAT_VALUE_SAMPLE }
};
```

会话的网络活动

本视图显示自会话开始以来，通过所有 Adaptive Server 网络连接进行的网络活动。它显示：

- 缺省包大小
- 最大包大小
- 从会话开始已发送和接收的包的平均大小
- 发送出的包数量
- 接收到的包数量
- 发送和接收包的速率
- 发送出的字节数
- 接收到的字节数
- 发送和接收字节的速率

```
SMC_SIZET session_network_activity_count = 12;
SMC_DATAITEM_STRUCT session_network_activity_view[] = {
{ SMC_NAME_NET_DEFAULT_PKT_SIZE,   SMC_STAT_VALUE_SAMPLE },
{ SMC_NAME_NET_MAX_PKT_SIZE,       SMC_STAT_VALUE_SAMPLE },
{ SMC_NAME_NET_PKT_SIZE_SENT,      SMC_STAT_VALUE_SESSION },
{ SMC_NAME_NET_PKT_SIZE_RCVD,      SMC_STAT_VALUE_SESSION },
{ SMC_NAME_NET_PKTS_SENT,          SMC_STAT_VALUE_SESSION },
{ SMC_NAME_NET_PKTS_RCVD,          SMC_STAT_VALUE_SESSION },
{ SMC_NAME_NET_PKTS_SENT,          SMC_STAT_RATE_SESSION },
{ SMC_NAME_NET_PKTS_RCVD,          SMC_STAT_RATE_SESSION },
{ SMC_NAME_NET_BYTES_SENT,         SMC_STAT_VALUE_SESSION },
{ SMC_NAME_NET_BYTES_RCVD,         SMC_STAT_VALUE_SESSION },
{ SMC_NAME_NET_BYTES_SENT,         SMC_STAT_RATE_SESSION },
{ SMC_NAME_NET_BYTES_RCVD,         SMC_STAT_RATE_SESSION }
};
```

采样间隔期间的网络活动

本视图显示在最近一次采样间隔期间，通过所有 Adaptive Server 网络连接进行的网络活动。它显示：

- 缺省包大小
- 最大包大小
- 采样间隔内已发送和接收的包的平均大小
- 发送出的包数量
- 接收到的包数量
- 发送和接收包的速率
- 发送出的字节数
- 接收到的字节数
- 发送和接收字节的速率

```
SMC_SIZET sample_network_activity_count = 12;
SMC_DATAITEM_STRUCT sample_network_activity_view[] = {
{ SMC_NAME_NET_DEFAULT_PKT_SIZE,   SMC_STAT_VALUE_SAMPLE },
{ SMC_NAME_NET_MAX_PKT_SIZE,       SMC_STAT_VALUE_SAMPLE },
{ SMC_NAME_NET_PKT_SIZE_SENT,      SMC_STAT_VALUE_SAMPLE },
{ SMC_NAME_NET_PKT_SIZE_RCVD,      SMC_STAT_VALUE_SAMPLE },
{ SMC_NAME_NET_PKTS_SENT,          SMC_STAT_VALUE_SAMPLE },
{ SMC_NAME_NET_PKTS_RCVD,          SMC_STAT_VALUE_SAMPLE },
{ SMC_NAME_NET_PKTS_SENT,          SMC_STAT_RATE_SAMPLE },
{ SMC_NAME_NET_PKTS_RCVD,           SMC_STAT_RATE_SAMPLE },
{ SMC_NAME_NET_BYTES_SENT,          SMC_STAT_VALUE_SAMPLE },
{ SMC_NAME_NET_BYTES_RCVD,          SMC_STAT_VALUE_SAMPLE },
{ SMC_NAME_NET_BYTES_SENT,          SMC_STAT_RATE_SAMPLE },
{ SMC_NAME_NET_BYTES_RCVD,,         SMC_STAT_RATE_SAMPLE }
};
```

网络性能摘要

本视图显示最近一次采样间隔期间， Adaptive Server 通过其所有网络连接执行的活动的速率。它显示在这段间隔内， Adaptive Server 每秒接收和发送的字节数。

```
SMC_SIZET network_perf_sum_count = 2;
SMC_DATAITEM_STRUCT network_perf_sum_view[] = {
{ SMC_NAME_NET_BYTES_RCVD,      SMC_STAT_RATE_SAMPLE },
{ SMC_NAME_NET_BYTES_SENT,      SMC_STAT_RATE_SAMPLE }
};
```

会话的过程高速缓存统计信息

本视图显示自会话开始以来， Adaptive Server 过程高速缓存的效率。它显示：

- 由过程高速缓存满足的存储过程执行请求数所占的百分比
- 自会话开始以来，该存储过程的逻辑读取次数和物理读取次数
- 自会话开始以来，该存储过程的逻辑读取和物理读取总速率

```
SMC_SIZET session_procedure_cache_stats_count = 5;
SMC_DATAITEM_STRUCT session_procedure_cache_stats_view[] = {
{ SMC_NAME_STP_HIT_PCT,         SMC_STAT_VALUE_SESSION },
{ SMC_NAME_STP_LOGICAL_READ,    SMC_STAT_VALUE_SESSION },
{ SMC_NAME_STP_LOGICAL_READ,    SMC_STAT_RATE_SESSION },
{ SMC_NAME_STP_PHYSICAL_READ,   SMC_STAT_VALUE_SESSION },
{ SMC_NAME_STP_PHYSICAL_READ,   SMC_STAT_RATE_SESSION }
};
```

采样间隔期间的过程高速缓存统计信息

本视图显示最近一次采样间隔期间， Adaptive Server 过程高速缓存的效率。它显示：

- 最近一次采样间隔期间，由过程高速缓存满足的存储过程执行请求数所占的百分比
- 最近一次采样间隔期间，存储过程的逻辑读取次数和物理读取次数
- 最近一次采样间隔期间，存储过程的逻辑读取和物理读取速率

```
SMC_SIZET sample_procedure_cache_stats_count = 5;
SMC_DATAITEM_STRUCT sample_procedure_cache_stats_view[] = {
{ SMC_NAME_STP_HIT_PCT,         SMC_STAT_VALUE_SAMPLE },
{ SMC_NAME_STP_LOGICAL_READ,    SMC_STAT_VALUE_SAMPLE },
{ SMC_NAME_STP_LOGICAL_READ,    SMC_STAT_RATE_SAMPLE },
{ SMC_NAME_STP_PHYSICAL_READ,   SMC_STAT_VALUE_SAMPLE },
{ SMC_NAME_STP_PHYSICAL_READ,   SMC_STAT_RATE_SAMPLE }
};
```

过程页 I/O

本视图显示在最近一次采样间隔期间，运行存储过程时，发生的页 I/O。它为在该采样间隔期间生成页 I/O 的每个存储过程，显示存储过程名和 ID，和含有该过程的数据库名和 ID。如果无活动的存储过程时产生了页 I/O，那么这些 I/O 与过程 ID 和数据库 ID 零值关联。

本视图在每个存储过程级别上还显示：

- 页 I/O 总数
- 可以由 Adaptive Server 数据高速缓存满足的页 I/O 请求数所占的百分比
- 最近一次采样间隔期间，在执行存储过程时发生的逻辑读取次数、物理读取次数和页写入次数。

```
SMC_SIZET procedure_page_cache_io_count = 9;
SMC_DATAITEM_STRUCT procedure_page_cache_io_view[] = {
{ SMC_NAME_ACT_STP_DB_NAME,      SMC_STAT_VALUE_SAMPLE },
{ SMC_NAME_ACT_STP_DB_ID,        SMC_STAT_VALUE_SAMPLE },
{ SMC_NAME_ACT_STP_NAME,         SMC_STAT_VALUE_SAMPLE },
{ SMC_NAME_ACT_STP_ID,           SMC_STAT_VALUE_SAMPLE },
{ SMC_NAME_PAGE_IO,              SMC_STAT_VALUE_SAMPLE },
{ SMC_NAME_PAGE_HIT_PCT,         SMC_STAT_VALUE_SAMPLE },
{ SMC_NAME_PAGE_LOGICAL_READ,    SMC_STAT_VALUE_SAMPLE },
{ SMC_NAME_PAGE_PHYSICAL_READ,   SMC_STAT_VALUE_SAMPLE },
{ SMC_NAME_PAGE_WRITE,           SMC_STAT_VALUE_SAMPLE }
};
```

进程活动

本视图显示 Adaptive Server 中所有进程的 CPU 使用情况、页 I/O 和当前进程状态。

对于最近一次采样间隔中的每个进程，它显示：

- 登录名
- 进程 ID
- 内核进程 ID
- 当前进程状态

本视图还显示自会话开始以来每个进程累计的连接时间、页 I/O 总数和 CPU 占用时间。

```
SMC_SIZET process_activity_count = 7;
SMC_DATAITEM_STRUCT process_activity_view[] = {
{ SMC_NAME_LOGIN_NAME,         SMC_STAT_VALUE_SAMPLE },
{ SMC_NAME_SPID,               SMC_STAT_VALUE_SAMPLE },
{ SMC_NAME_KPID,               SMC_STAT_VALUE_SAMPLE },
{ SMC_NAME_CONNECT_TIME,       SMC_STAT_VALUE_SESSION },
{ SMC_NAME_PAGE_IO,            SMC_STAT_VALUE_SESSION },
{ SMC_NAME_CPU_TIME,           SMC_STAT_VALUE_SESSION },
{ SMC_NAME_CUR_PROC_STATE,     SMC_STAT_VALUE_SAMPLE }
};
```

进程数据库对象页 I/O

本视图按数据库对象显示每个 Adaptive Server 进程的页 I/O 数。对于在最近一次采样间隔期间拥有页 I/O 的每个进程，它显示：

- 登录名
- 进程 ID
- 内核进程 ID

对于每个此类进程和它访问的每个数据库对象，该视图显示：

- 对象名
- 对象 ID
- 数据库名和 ID
- 页 I/O 数

本视图还显示页 I/O 总数、可以由 Adaptive Server 高速缓存满足的页 I/O 请求数所占的百分比，以及最近一次采样间隔期间发生的逻辑读取次数、物理读取次数和页写入次数。

```
SMC_SIZET process_object_page_io_count = 13;
SMC_DATAITEM_STRUCT process_object_page_io_view[] = {
{ SMC_NAME_LOGIN_NAME,           SMC_STAT_VALUE_SAMPLE },
{ SMC_NAME_SPID,                 SMC_STAT_VALUE_SAMPLE },
{ SMC_NAME_KPID,                 SMC_STAT_VALUE_SAMPLE },
{ SMC_NAME_DB_NAME,              SMC_STAT_VALUE_SAMPLE },
{ SMC_NAME_DB_ID,                SMC_STAT_VALUE_SAMPLE },
{ SMC_NAME_OBJ_NAME,             SMC_STAT_VALUE_SAMPLE },
{ SMC_NAME_OBJ_ID,               SMC_STAT_VALUE_SAMPLE },
{ SMC_NAME_OBJ_TYPE,             SMC_STAT_VALUE_SAMPLE },
{ SMC_NAME_PAGE_IO,              SMC_STAT_VALUE_SAMPLE },
{ SMC_NAME_PAGE_HIT_PCT,         SMC_STAT_VALUE_SAMPLE },
{ SMC_NAME_PAGE_LOGICAL_READ,    SMC_STAT_VALUE_SAMPLE },
{ SMC_NAME_PAGE_PHYSICAL_READ,   SMC_STAT_VALUE_SAMPLE },
{ SMC_NAME_PAGE_WRITE,           SMC_STAT_VALUE_SAMPLE }
};
```

进程的锁明细

本视图显示到最近一次采样间隔结束时， Adaptive Server 进程持有或请求的锁的状态。每个锁由以下内容标识：

- 登录名
- 进程 ID
- 与锁相关联的 Adaptive Server 进程的内核进程 ID
- 被锁定对象的名称和 ID
- 包含该对象的数据库的名称和 ID
- 锁应用到的页号 （如果是页锁）
- 每个锁的当前状态
- 这是否是请求锁的指示

如果该进程正在请求该锁，将显示该进程为获得该锁已经等待的时间和持有该锁的进程的进程 ID。如果该进程持有该锁，将显示等待获得该锁的其它进程的数量。

```
SMC_SIZET process_detail_locks_count = 13;
SMC_DATAITEM_STRUCT process_detail_locks_view[] = {
{ SMC_NAME_LOGIN_NAME,               SMC_STAT_VALUE_SAMPLE },
{ SMC_NAME_SPID,                     SMC_STAT_VALUE_SAMPLE },
{ SMC_NAME_KPID,                     SMC_STAT_VALUE_SAMPLE },
{ SMC_NAME_DB_NAME,                  SMC_STAT_VALUE_SAMPLE },
{ SMC_NAME_DB_ID,                    SMC_STAT_VALUE_SAMPLE },
{ SMC_NAME_OBJ_NAME,                 SMC_STAT_VALUE_SAMPLE },
{ SMC_NAME_OBJ_ID,                   SMC_STAT_VALUE_SAMPLE },
{ SMC_NAME_PAGE_NUM,                 SMC_STAT_VALUE_SAMPLE },
{ SMC_NAME_LOCK_STATUS,              SMC_STAT_VALUE_SAMPLE },
{ SMC_NAME_DEMAND_LOCK,              SMC_STAT_VALUE_SAMPLE },
{ SMC_NAME_TIME_WAITED_ON_LOCK,      SMC_STAT_VALUE_SAMPLE },
{ SMC_NAME_BLOCKING_SPID,            SMC_STAT_VALUE_SAMPLE },
{ SMC_NAME_LOCKS_BEING_BLOCKED_CNT,  SMC_STAT_VALUE_SAMPLE }
};
```

进程的页 I/O 明细

本视图详细显示每个 Adaptive Server 进程的页 I/O。它显示到最新一次采样间隔结束时的下列内容：

- 登录名
- 进程 ID
- 内核进程 ID
- 显示每个 Adaptiver Server 进程的进程状态和当前引擎

本视图显示采样间隔期间以及会话开始以来，可由 Adaptive Server 数据高速缓存满足的页 I/O 请求数所占的百分比。它还显示自会话开始以来发生的逻辑读取次数、物理读取次数和页写入次数。

```
SMC_SIZET process_detail_io_count = 12;
SMC_DATAITEM_STRUCT process_detail_io_view[] = {
{ SMC_NAME_LOGIN_NAME,          SMC_STAT_VALUE_SAMPLE },
{ SMC_NAME_SPID,                SMC_STAT_VALUE_SAMPLE },
{ SMC_NAME_KPID,                SMC_STAT_VALUE_SAMPLE },
{ SMC_NAME_CUR_PROC_STATE,      SMC_STAT_VALUE_SAMPLE },
{ SMC_NAME_CUR_ENGINE,          SMC_STAT_VALUE_SAMPLE },
{ SMC_NAME_CONNECT_TIME,        SMC_STAT_VALUE_SESSION },
{ SMC_NAME_CPU_TIME,            SMC_STAT_VALUE_SESSION },
{ SMC_NAME_PAGE_HIT_PCT,        SMC_STAT_VALUE_SAMPLE },
{ SMC_NAME_PAGE_HIT_PCT,        SMC_STAT_VALUE_SESSION },
{ SMC_NAME_PAGE_LOGICAL_READ,   SMC_STAT_VALUE_SESSION },
{ SMC_NAME_PAGE_PHYSICAL_READ,  SMC_STAT_VALUE_SESSION },
{ SMC_NAME_PAGE_WRITE,          SMC_STAT_VALUE_SESSION }
};
```

进程锁

本视图显示最近一次采样间隔期间，产生锁请求的每个 Adaptive Server 进程的锁请求计数。

```
SMC_SIZET process_lock_count = 4;
SMC_DATAITEM_STRUCT process_lock_view[] = {
{ SMC_NAME_LOGIN_NAME,    SMC_STAT_VALUE_SAMPLE },
{ SMC_NAME_SPID,          SMC_STAT_VALUE_SAMPLE },
{ SMC_NAME_KPID,          SMC_STAT_VALUE_SAMPLE },
{ SMC_NAME_LOCK_CNT,      SMC_STAT_VALUE_SAMPLE }
};
```

进程页 I/O

本视图针对最近一次采样汇总了每个 Adaptive Server 进程的页 I/O。对于在该间隔中产生页 I/O 的每个 Adaptive Server 进程，本视图都会显示其登录名、进程 ID 和内核进程 ID。

对于每个进程，本视图还显示：

- 页 I/O 总数
- 可以由 Adaptive Server 数据高速缓存满足的页 I/O 请求数所占的百分比
- 最近一次采样间隔期间发生的逻辑读取次数、物理读取次数和写入次数

```
SMC_SIZET process_page_io_count = 8;
SMC_DATAITEM_STRUCT process_page_io_view[] = {
{ SMC_NAME_LOGIN_NAME,             SMC_STAT_VALUE_SAMPLE },
{ SMC_NAME_SPID,                   SMC_STAT_VALUE_SAMPLE },
{ SMC_NAME_KPID,                   SMC_STAT_VALUE_SAMPLE },
{ SMC_NAME_PAGE_IO,                SMC_STAT_VALUE_SAMPLE },
{ SMC_NAME_PAGE_HIT_PCT,           SMC_STAT_VALUE_SAMPLE },
{ SMC_NAME_PAGE_LOGICAL_READ,      SMC_STAT_VALUE_SAMPLE },
{ SMC_NAME_PAGE_PHYSICAL_READ,     SMC_STAT_VALUE_SAMPLE },
{ SMC_NAME_PAGE_WRITE,             SMC_STAT_VALUE_SAMPLE }
};
```

进程状态摘要

本视图显示最近一次采样间隔结束时，处于每种进程状态的进程数目。

```
SMC_SIZET process_perf_sum_count = 2;
SMC_DATAITEM_STRUCT process_perf_sum_view[] = {
{ SMC_NAME_PROC_STATE,          SMC_STAT_VALUE_SAMPLE },
{ SMC_NAME_PROC_STATE_CNT,      SMC_STAT_VALUE_SAMPLE }
};
```

进程存储过程页 I/O

本视图显示与 Adaptive Server 进程执行的存储过程关联的页 I/O。它显示采样间隔期间，发生页 I/O 的每个进程的登录名，进程 ID 和内核进程 ID。

对于发生页 I/O 的每个进程和存储过程，它显示包含该存储过程的数据库的名称和 ID 以及该过程本身的名称和 ID。

对最近一次采样间隔，该视图显示：

- 页 I/O 总数
- 可由数据高速缓存满足的页 I/O 请求数所占的百分比
- 逻辑读取次数、物理读取次数和页写入次数

```
SMC_SIZET process_procedure_page_io_count = 12;
SMC_DATAITEM_STRUCT process_procedure_page_io_view[] = {
{ SMC_NAME_LOGIN_NAME,             SMC_STAT_VALUE_SAMPLE },
{ SMC_NAME_SPID,                   SMC_STAT_VALUE_SAMPLE },
{ SMC_NAME_KPID,                   SMC_STAT_VALUE_SAMPLE },
{ SMC_NAME_ACT_STP_DB_NAME,        SMC_STAT_VALUE_SAMPLE },
{ SMC_NAME_ACT_STP_DB_ID,          SMC_STAT_VALUE_SAMPLE },
{ SMC_NAME_ACT_STP_NAME,           SMC_STAT_VALUE_SAMPLE },
{ SMC_NAME_ACT_STP_ID,             SMC_STAT_VALUE_SAMPLE },
{ SMC_NAME_PAGE_IO,                 SMC_STAT_VALUE_SAMPLE },
{ SMC_NAME_PAGE_HIT_PCT,            SMC_STAT_VALUE_SAMPLE },
{ SMC_NAME_PAGE_LOGICAL_READ,       SMC_STAT_VALUE_SAMPLE },
{ SMC_NAME_PAGE_PHYSICAL_READ,      SMC_STAT_VALUE_SAMPLE },
{ SMC_NAME_PAGE_WRITE,              SMC_STAT_VALUE_SAMPLE }
};
```

服务器性能摘要

本视图显示 Adaptive Server 的总体性能。它显示：

- 每秒的锁请求数量
- Adaptive Server 处于繁忙状态的时间占采样间隔的百分比
- 每秒处理的事务数量
- 在最近一次采样间隔期间， Adaptive Server 检测到死锁的次数

```
SMC_SIZET server_perf_sum_count = 4;
SMC_DATAITEM_STRUCT server_perf_sum_view[] = {
{ SMC_NAME_LOCK_CNT,            SMC_STAT_RATE_SAMPLE },
{ SMC_NAME_CPU_BUSY_PCT,        SMC_STAT_VALUE_SAMPLE },
{ SMC_NAME_XACT,                SMC_STAT_RATE_SAMPLE },
{ SMC_NAME_DEADLOCK_CNT,        SMC_STAT_VALUE_SAMPLE }
};
```

存储过程活动

本视图显示过程语句的存储过程活动。在最近一次采样间隔期间执行的任意存储过程的每个语句由以下内容标识：

- 包含该过程的数据库的名称和 ID
- 该过程的名称和 ID
- 该语句在该存储过程内的相对编号
- 作为该语句起始位置的过程文本行

该视图显示：

- 在最近一次采样间隔期间内以及自该会话开始以来，每条语句执行的次数
- 在采样间隔期间以及该会话自开始到目前为止的这段时间，执行该语句平均需要的时间

```
SMC_SIZET procedure_activity_count = 10;
SMC_DATAITEM_STRUCT procedure_activity_view[] = {
{ SMC_NAME_ACT_STP_DB_ID,SMC_STAT_VALUE_SAMPLE },
{ SMC_NAME_ACT_STP_DB_NAME,          SMC_STAT_VALUE_SAMPLE },
{ SMC_NAME_ACT_STP_ID,               SMC_STAT_VALUE_SAMPLE },
{ SMC_NAME_ACT_STP_NAME,             SMC_STAT_VALUE_SAMPLE },
{ SMC_NAME_STP_LINE_NUM,             SMC_STAT_VALUE_SAMPLE },
{ SMC_NAME_STP_STMT_NUM,             SMC_STAT_VALUE_SAMPLE },
{ SMC_NAME_STP_NUM_TIMES_EXECUTED,   SMC_STAT_VALUE_SAMPLE },
{ SMC_NAME_STP_NUM_TIMES_EXECUTED,   SMC_STAT_VALUE_SESSION },
{ SMC_NAME_STP_ELAPSED_TIME,         SMC_STAT_AVG_SAMPLE },
{ SMC_NAME_STP_ELAPSED_TIME,         SMC_STAT_AVG_SESSION }
};
```

事务活动

本视图显示在采样间隔期间和会话期间 Adaptive Server 中发生的事务活动。

```
SMC_SIZET transaction_activity_count = 20;
SMC_DATAITEM_STRUCT transaction_activity_view[] = {
{ SMC_NAME_XACT,                   SMC_STAT_VALUE_SAMPLE },
{ SMC_NAME_XACT_DELETE,            SMC_STAT_VALUE_SAMPLE },
{ SMC_NAME_XACT_INSERT,            SMC_STAT_VALUE_SAMPLE },
{ SMC_NAME_XACT_UPDATE,            SMC_STAT_VALUE_SAMPLE },
{ SMC_NAME_XACT_UPDATE_DIRECT,     SMC_STAT_VALUE_SAMPLE },
{ SMC_NAME_XACT,                   SMC_STAT_VALUE_SESSION },
{ SMC_NAME_XACT_DELETE,            SMC_STAT_VALUE_SESSION },
{ SMC_NAME_XACT_INSERT,            SMC_STAT_VALUE_SESSION },
{ SMC_NAME_XACT_UPDATE,            SMC_STAT_VALUE_SESSION },
{ SMC_NAME_XACT_UPDATE_DIRECT,     SMC_STAT_VALUE_SESSION },
{ SMC_NAME_XACT,                   SMC_STAT_RATE_SAMPLE },
{ SMC_NAME_XACT_DELETE,            SMC_STAT_RATE_SAMPLE },
{ SMC_NAME_XACT_INSERT,            SMC_STAT_RATE_SAMPLE },
{ SMC_NAME_XACT_UPDATE,            SMC_STAT_RATE_SAMPLE },
{ SMC_NAME_XACT_UPDATE_DIRECT,     SMC_STAT_RATE_SAMPLE },
{ SMC_NAME_XACT,                   SMC_STAT_RATE_SESSION },
{ SMC_NAME_XACT_DELETE,            SMC_STAT_RATE_SESSION },
{ SMC_NAME_XACT_INSERT,            SMC_STAT_RATE_SESSION },
{ SMC_NAME_XACT_UPDATE,,           SMC_STAT_RATE_SESSION },
{ SMC_NAME_XACT_UPDATE_DIRECT,     SMC_STAT_RATE_SESSION }
};SMC_SIZET num_views = 27;
SMC_SIZET* view_count = (SMC_SIZET*) malloc (sizeof(SMC_SIZET)
```

```
   * num_views );
SMC_DATAITEM_STRUCT** view_list = (SMC_DATAITEM_STRUCT**)
   malloc (sizeof(SMC_DATAITEM_STRUCT*) * num_views );
SMC_SIZET** view_id_handle_list = (SMC_SIZET**) malloc
   (sizeof(SMC_SIZET*) * num_views );
  SMC_SIZET* view_id_list = (SMC_SIZET*) malloc
   (sizeof(SMC_SIZET) * num_views );
  SMC_SIZET  client_id;
  SMC_SIZETP client_id_handle = &client_id;

  SMC_SERVER_MODE server_mode = SMC_SERVER_M_LIVE;
  SMC_CHAR server_name[ 40 ];
  SMC_CHAR user_name[ 40 ];
  SMC_CHAR password[ 40 ];
  SMC_CHAR interfaces_file[ 40 ];

  SMC_RETURN_CODE ret;
  SMC_SIZET refresh_num, view_num, col_num, row_num;

  SMC_SIZET num_refreshes = 10;

  SMC_SIZET  row_count;
  SMC_SIZETP row_count_handle = &row_count;

  SMC_DATAITEM_STRUCTP dataitem_list;
  SMC_DATAITEM_NAME dataitem_name;
  SMC_CHARP dataitem_name_str;
  SMC_DATAITEM_STATTYPE dataitem_stat;

  SMC_CHARP dataitem_stat_str;
  SMC_DATAITEM_TYPE dataitem_type;

  SMC_VALUE_UNION data_union;
  SMC_VALUE_UNIONP data_union_handle = &data_union;
  SMC_CHARP data_str;
  SMC_INT ival;
printf("************************************************\n");
printf("** Test Driver for SQL Monitor Client Library **\n");
printf("************************************************\n");
if (argc != 5)
{
    printf(Usage: testcli <SQLMonitorServer> <user> <password>
   <"interfaces_file>\n");
    exit(1);
  }
```

```
strcpy(server_name, argv[1]);
strcpy(user_name, argv[2]);
strcpy(password, argv[3]);
strcpy(interfaces_file, argv[4]);

for(view_num=0; view_num<num_views; view_num++)
{
  view_id_handle_list[ view_num ] = &(view_id_list[ view_num ]);
}

view_count [  0 ] = cache_perf_sum_count;
view_list  [  0 ] = cache_perf_sum_view;
view_count [  1 ] = object_lock_status_count;
view_list  [  1 ] = object_lock_status_view;
view_count [  2 ] = object_page_io_count;
view_list  [  2 ] = object_page_io_view;
view_count [  3 ] = session_page_cache_stats_count;
view_list  [  3 ] = session_page_cache_stats_view;
view_count [  4 ] = sample_page_cache_stats_count;
view_list  [  4 ] = sample_page_cache_stats_view;
view_count [  5 ] = session_device_io_count;
view_list  [  5 ] = session_device_io_view;
view_count [  6 ] = sample_device_io_count;
view_list  [  6 ] = sample_device_io_view;
view_count [  7 ] = device_perf_sum_count;
view_list  [  7 ] = device_perf_sum_view;
view_count [  8 ] = engine_activity_count;
view_list  [  8 ] = engine_activity_view;
view_count [  9 ] = lock_perf_sum_count;
view_list  [  9 ] = lock_perf_sum_view;
view_count [ 10 ] = session_network_activity_count;
view_list  [ 10 ] = session_network_activity_view;
view_count [ 11 ] = sample_network_activity_count;
view_list  [ 11 ] = sample_network_activity_view;
view_count [ 12 ] = network_perf_sum_count;
view_list  [ 12 ] = network_perf_sum_view;
view_count [ 13 ] = session_procedure_cache_stats_count;
view_list  [ 13 ] = session_procedure_cache_stats_view;
view_count [ 14 ] = sample_procedure_cache_stats_count;
view_list  [ 14 ] = sample_procedure_cache_stats_view;
view_count [ 15 ] = procedure_page_cache_io_count;
view_list  [ 15 ] = procedure_page_cache_io_view;
view_count [ 16 ] = process_activity_count;
view_list  [ 16 ] = process_activity_view;
view_count [ 17 ] = process_object_page_io_count;
view_list  [ 17 ] = process_object_page_io_view;
```

```
  view_count [ 18 ] = process_detail_locks_count;
  view_list  [ 18 ] = process_detail_locks_view;
  view_count [ 19 ] = process_detail_io_count;
  view_list  [ 19 ] = process_detail_io_view;
  view_count [ 20 ] = process_lock_count;
  view_list  [ 20 ] = process_lock_view;
  view_count [ 21 ] = process_page_io_count;
  view_list  [ 21 ] = process_page_io_view;
  view_count [ 22 ] = process_perf_sum_count;
  view_list  [ 22 ] = process_perf_sum_view;
  view_count [ 23 ] = process_procedure_page_io_count;
  view_list  [ 23 ] = process_procedure_page_io_view;
  view_count [ 24 ] = server_perf_sum_count;
  view_list  [ 24 ] = server_perf_sum_view;
  view_count [ 25 ] = procedure_activity_count;
  view_list  [ 25 ] = procedure_activity_view;
  view_count [ 26 ] = transaction_activity_count;
  view_list  [ 26 ] = transaction_activity_view;

  printf("**********   testing smc_connect()   **********\n");
  ret = smc_connect(server_mode,
                    server_name,
                    user_name,
                    password,
                    interfaces_file,
                    ErrorCallback,
                    0,
                    0,
                    client_id_handle);
  if ( ret != SMC_RET_SUCCESS )
  {
    printf("error returned by smc_connect()\n");
    return (int) ret;
  }
  else
  {
    printf("smc_connect() succeeded\n");
  }
printf("**********   testing smc_create_view()   **********\n");
for(view_num=0; view_num<num_views; view_num++)
{
ret = smc_create_view(client_id,
          view_list[ view_num ],
          view_count[ view_num ],
                    (SMC_CHARP) 0,
                    view_id_handle_list[ view_num ]);
```

```
    if ( ret != SMC_RET_SUCCESS )
    {
      printf("error returned by smc_create_view( %d )\n",
         view_num);
      return (int) ret;
    }
    else
    {
      printf("smc_create_view( %d ) succeeded\n", view_num);
    }
  }
printf("**********   testing smc_refresh()   **********\n");
for(refresh_num=0; refresh_num<num_refreshes; refresh_num++)
{
    ret = smc_refresh(client_id,
                      (SMC_VOIDP) 0,
                      RefreshCallback,
                      0);
    if ( ret != SMC_RET_SUCCESS )
    {
      printf("error returned by smc_refresh() number %d\n",
         refresh_num);
      return (int) ret;
    }
    else
    {
      printf("smc_refresh() number %d succeeded\n", refresh_num);
    }

    for(view_num=0; view_num<num_views;  view_num++)
    {
      printf("******   testing smc_get_row_count()   ******\n");
      ret = smc_get_row_count(client_id,
                              view_id_list[ view_num ],
                              row_count_handle);
      if ( ret != SMC_RET_SUCCESS )
      {
        printf("error returned by smc_get_row_count()\n");
        return (int) ret;
      }
      else
      {
        printf("smc_get_row_count( view_id = %d ) = %d\n",
         view_id_list[view_num], row_count);
      }
```

```
dataitem_list = view_list[view_num];

/* print dataitem name headers */
for(col_num = 0; col_num<view_count[ view_num ]; col_num++)
{
  dataitem_name = (dataitem_list[col_num]).dataItemName;
  dataitem_name_str = LookupDataItemName( dataitem_name );
  printf("Col %d %s\t", col_num, dataitem_name_str);
}
printf("\n");

/* print dataitem stattype headers */
for(col_num = 0; col_num<view_count[ view_num ]; col_num++)
{
  dataitem_stat = (dataitem_list[col_num]).dataItemStatType;
  dataitem_stat_str = LookupDataItemStat( dataitem_stat );
  printf("Col %d %s\t", col_num, dataitem_stat_str);
}
printf("\n");

for(row_num = 0; row_num<row_count; row_num++)
{
  for(col_num = 0; col_num<view_count[ view_num ];
   col_num++)
  {
    dataitem_name = (dataitem_list[col_num]).dataItemName;
    dataitem_stat = (dataitem_list[col_num]).dataItemStatType;
    dataitem_name_str = LookupDataItemName( dataitem_name );
    ret = smc_get_dataitem_value(client_id,
                                 view_id_list[ view_num ],
                                 &(dataitem_list[col_num]),
                                 row_num,
                                 data_union_handle);
    if ( ret != SMC_RET_SUCCESS )
    {
      printf("error returned by smc_get_dataitem_value()\n");
      return (int) ret;
    }

    smc_get_dataitem_type(&(dataitem_list[col_num]),
                          &dataitem_type);

    switch(dataitem_type)
    {
      case SMC_DI_TYPE_CHARP:
        printf("Col %d:
```

```
      \"%s\"\t",col_num,data_union.stringValue);
  free( data_union.stringValue );
  break;
case SMC_DI_TYPE_DOUBLE:
  printf("Col %d:
     %f\t",col_num,data_union.doubleValue);
  break;
case SMC_DI_TYPE_ENUMS:
  ival = data_union.intValue;
  switch (dataitem_name)
  {
    case SMC_NAME_LOCK_RESULT_SUMMARY:
      data_str = LookupLockResultSummary(
         ((SMC_LOCK_RESULT_SUMMARY) ival) );
      printf("Col %d: \"%s\"\t",col_num, data_str );
      break;
    case SMC_NAME_LOCK_RESULT:
      data_str = LookupLockResult(
         ((SMC_LOCK_RESULT) ival) );
      printf("Col %d: \"%s\"\t",col_num, data_str );
      break;
    case SMC_NAME_LOCK_STATUS:
      data_str = LookupLockStatus(
         ((SMC_LOCK_STATUS) ival) );
      printf("Col %d: \"%s\"\t",col_num, data_str );
      break;
    case SMC_NAME_LOCK_TYPE:
      data_str = LookupLockType( ((SMC_LOCK_TYPE)
        ival) );
      printf("Col %d: \"%s\"\t",col_num, data_str );
      break;
    case SMC_NAME_OBJ_TYPE:
      data_str = LookupObjectType( ((SMC_OBJ_TYPE)
        ival) );
      printf("Col %d: \"%s\"\t",col_num, data_str );
      break;
    case SMC_NAME_CUR_PROC_STATE:
    case SMC_NAME_PROC_STATE:
      data_str = LookupProcessState(
        ((SMC_PROCESS_STATE) ival) );
      printf("Col %d: \"%s\"\t",col_num, data_str );
      break;
    default:
      printf("Col %d: \"ERR with %s\"\t",col_num,
        dataitem_name_str );
  }
```

```
              break;
            case SMC_DI_TYPE_LONG:
              printf("Col %d: %d\t",col_num,
                data_union.longValue);
              break;
            case SMC_DI_TYPE_DATIM:
            case SMC_DI_TYPE_NONE:
            default:
              printf("Col %d: \"ERR with %s\"\t",col_num,
                dataitem_name_str );
          }
        }
        printf("\n");
      }
    }
  }

  printf("**********   testing smc_disconnect()   **********\n");
  ret = smc_disconnect(client_id);
  if ( ret != SMC_RET_SUCCESS )
  {
    printf("error returned by smc_disconnect
    return (int) ret;
  }
  {
    printf("smc_disconnect() succeeded\n");
  }

  free(view_count);
  free(view_list);

  return 0;
}



SMC_VOID
ErrorCallback(
  SMC_SIZET    id,
  SMC_SIZET    error_number,
  SMC_SIZET    severity,
  SMC_SIZET    map_severity,
  SMC_SIZET    source,
  SMC_CCHARP   error_msg,
  SMC_SIZET    state
  )
```

```
{
  printf("**************************************\n");
  printf("Inside ErrorCallback()\n");

  printf("id = %d\n", id);
  printf("error_number = %d\n", error_number);
  printf("err severity = %d\n", severity);
  printf("map severity = %d\n", map_severity);
  printf("source = %d\n", source);
  printf("error msg = %s\n", error_msg);
  printf("state = %d\n", state);
  printf("**************************************\n");
  return;
}

SMC_VOID
RefreshCallback(
  SMC_SIZET   id,
  SMC_VOIDP   user_msg,
  SMC_CHARP   msg
  )
{
  printf("**************************************\n");
  printf("Inside RefreshCallback()\n");

  printf("id = %d\n", id);
  printf("user_msg = %s\n", (SMC_CHARP) user_msg);
  printf("msg = %s\n", msg);

  return;
}

SMC_CHARP
LookupDataItemName(
  SMC_DATAITEM_NAME   value
  )
{
  typedef struct {
    SMC_CHARP           str_name;
    SMC_DATAITEM_NAME   enum_name;
  } DATAITEM_NAME_MAPPER;
  DATAITEM_NAME_MAPPER   dataitem_name_map[] = {
  { "Process ID",                   SMC_NAME_SPID },
  { "Kernel Process ID",            SMC_NAME_KPID },
  { "Cache Name",                   SMC_NAME_DATA_CACHE_NAME },
  { "Database ID",                  SMC_NAME_DB_ID },
```

```
{ "Object ID",                    SMC_NAME_OBJ_ID },
{ "Procedure Database ID",        SMC_NAME_ACT_STP_DB_ID },
{ "Procedure ID",                 SMC_NAME_ACT_STP_ID },
{ "Procedure Line Number",        SMC_NAME_STP_LINE_NUM },
{ "Lock Type",                    SMC_NAME_LOCK_TYPE },
{ "Lock Result",                  SMC_NAME_LOCK_RESULT },
{ "Lock Results Summarized",      SMC_NAME_LOCK_RESULT_SUMMARY },
{ "Lock Status",                  SMC_NAME_LOCK_STATUS },
{ "Engine Number",                SMC_NAME_ENGINE_NUM },
{ "Page Number",                  SMC_NAME_PAGE_NUM },
{ "Device Name",                  SMC_NAME_DEV_NAME },
{ "Process State",                SMC_NAME_PROC_STATE },
{ "Login Name",                   SMC_NAME_LOGIN_NAME },
{ "Database Name",                SMC_NAME_DB_NAME },
{ "Owner Name",                   SMC_NAME_OWNER_NAME },
{ "Object Name",                  SMC_NAME_OBJ_NAME },
{ "Object Type",                  SMC_NAME_OBJ_TYPE },
{ "Procedure Database Name",      SMC_NAME_ACT_STP_DB_NAME },
{ "Procedure Owner Name",         SMC_NAME_ACT_STP_OWNER_NAME },
{ "Procedure Name",               SMC_NAME_ACT_STP_NAME },
{ "Blocking Process ID",          SMC_NAME_BLOCKING_SPID },
{ "Cache Efficiency",             SMC_NAME_DATA_CACHE_EFFICIENCY },
{ "Cache Hit Pct",                SMC_NAME_DATA_CACHE_HIT_PCT },
{ "Cache Hits",                   SMC_NAME_DATA_CACHE_HIT },
{ "Cache Misses",                 SMC_NAME_DATA_CACHE_MISS },
{ "Cache Spinlock Contention",    SMC_NAME_DATA_CACHE_CONTENTION },
{ "Connect Time",                 SMC_NAME_CONNECT_TIME },
{ "CPU Busy Percent",             SMC_NAME_CPU_BUSY_PCT },
{ "CPU Percent",                  SMC_NAME_CPU_PCT },
{ "CPU Time",                     SMC_NAME_CPU_TIME },
{ "Current Engine",               SMC_NAME_CUR_ENGINE },
{ "Current Process State",        SMC_NAME_CUR_PROC_STATE },
{ "Deadlock Count",               SMC_NAME_DEADLOCK_CNT },
{ "Demand Lock",                  SMC_NAME_DEMAND_LOCK },
{ "Device Hits",                  SMC_NAME_DEV_HIT },
{ "Device Hit Percent",           SMC_NAME_DEV_HIT_PCT },
{ "Device I/O",                   SMC_NAME_DEV_IO },
{ "Device Misses",                SMC_NAME_DEV_MISS },
{ "Device Reads",                 SMC_NAME_DEV_READ },
{ "Device Writes",                SMC_NAME_DEV_WRITE },
{ "Lock Count",                   SMC_NAME_LOCK_CNT },
{ "Lock Hit Percent",             SMC_NAME_LOCK_HIT_PCT },
{ "Lock Status Count",            SMC_NAME_LOCK_STATUS_CNT },
{ "Locks Being Blocked Count",    SMC_NAME_LOCKS_BEING_BLOCKED_CNT },
{ "Code Memory Size",             SMC_NAME_MEM_CODE_SIZE },
{ "Kernel Structures Memory Size", SMC_NAME_MEM_KERNEL_STRUCT_SIZE },
```

```
{ "Page Cache Size",              SMC_NAME_MEM_PAGE_CACHE_SIZE },
{ "Procedure Buffer Size",        SMC_NAME_MEM_PROC_BUFFER },
{ "Procedure Header Size",        SMC_NAME_MEM_PROC_HEADER },
{ "Server Structures Size",       SMC_NAME_MEM_SERVER_STRUCT_SIZE },
{ "Most Active Device I/O",       SMC_NAME_MOST_ACT_DEV_IO },
{ "Most Active Device Name",      SMC_NAME_MOST_ACT_DEV_NAME },
{ "Net I/O Bytes",                SMC_NAME_NET_BYTE_IO },
{ "Net Bytes Received",           SMC_NAME_NET_BYTES_RCVD },
{ "Net Bytes Sent",               SMC_NAME_NET_BYTES_SENT },
{ "Net Default Packet Size",      SMC_NAME_NET_DEFAULT_PKT_SIZE },
{ "Net Max Packet Size",          SMC_NAME_NET_MAX_PKT_SIZE },
{ "Net Packet Size Received",     SMC_NAME_NET_PKT_SIZE_RCVD },
{ "Net Packet Size Sent",         SMC_  NAME_NET_PKT_SIZE_SENT },
{ "Net Packets Received",         SMC_NAME_NET_PKTS_RCVD },
{ "Net Packets Sent",             SMC_NAME_NET_PKTS_SENT },
{ "Page Hit Percent",             SMC_NAME_PAGE_HIT_PCT },
{ "Logical Page Reads",           SMC_NAME_PAGE_LOGICAL_READ },
{ "Page I/O",                     SMC_NAME_PAGE_IO },
{ "Physical Page Reads",          SMC_NAME_PAGE_PHYSICAL_READ },
{ "Page Writes",                  SMC_NAME_PAGE_WRITE },
{ "Process State Count",          SMC_NAME_PROC_STATE_CNT },
{ "Timestamp",                    SMC_NAME_TIMESTAMP },
{ "Elapsed Time",                 SMC_NAME_ELAPSED_TIME },
{ "SQL Server Name",              SMC_NAME_SQL_SERVER_NAME },
{ "SQL Server Version",           SMC_NAME_SQL_SERVER_VERSION },
{ "Procedure Elapsed Time",       SMC_NAME_STP_ELAPSED_TIME },
{ "Procedure Hit Percent",        SMC_NAME_STP_HIT_PCT },
{ "Procedure Line Text",          SMC_NAME_STP_LINE_TEXT },
{ "Procedure Execution Count",    SMC_NAME_STP_NUM_TIMES_EXECUTED },
{ "Procedure Logical Reads",      SMC_NAME_STP_LOGICAL_READ },
{ "Procedure Physical Reads",     SMC_NAME_STP_PHYSICAL_READ },
{ "Time Waited on Lock",          SMC_NAME_TIME_WAITED_ON_LOCK },
{ "Transactions",                 SMC_NAME_XACT },
{ "Rows Deleted",                 SMC_NAME_XACT_DELETE },
{ "Rows Inserted Clustered",      SMC_NAME_XACT_CINSERT },
{ "Rows Inserted",                SMC_NAME_XACT_INSERT },
{ "Rows Inserted Nonclustered",   SMC_NAME_XACT_NCINSERT },
{ "Rows Updated",                 SMC_NAME_XACT_UPDATE },
{ "Rows Updated Directly",        SMC_NAME_XACT_UPDATE_DIRECT },
{ (SMC_CHARP)0,                   SMC_NAME_NONE }
};
SMC_INT    idx = 0;
SMC_BOOL   match = FALSE;
while( match == FALSE)
{
  if ( value == dataitem_name_map[ idx ].enum_name )
```

```
      return dataitem_name_map[ idx ].str_name;

    if (dataitem_name_map[ idx ].enum_name == SMC_NAME_NONE )
      return dataitem_name_map[ idx ].str_name;

    idx++;
  }
}


SMC_CHARP
LookupDataItemStat(
  SMC_DATAITEM_STATTYPE   value
  )
{
  typedef struct {
    SMC_CHARP               str_stat;
    SMC_DATAITEM_STATTYPE   enum_stat;
  } DATAITEM_STAT_MAPPER;

  DATAITEM_STAT_MAPPER   dataitem_stat_map[] = {
  { "Value for Sample",             SMC_STAT_VALUE_SAMPLE },
  { "Value for Session",            SMC_STAT_VALUE_SESSION },
  { "Rate for Sample",              SMC_STAT_RATE_SAMPLE },
  { "Rate for Session",             SMC_STAT_RATE_SESSION },
  { "Avg for Sample",               SMC_STAT_AVG_SAMPLE },
  { "Avg for Session",              SMC_STAT_AVG_SESSION },
  { (SMC_CHARP)0,    0 }
  };

  SMC_INT    idx = 0;
  SMC_BOOL   match = FALSE;

  while( match == FALSE)
  {
    if ( value == dataitem_stat_map[ idx ].enum_stat )
      return dataitem_stat_map[ idx ].str_stat;

    if (dataitem_stat_map[ idx ].enum_stat == 0 )
      return dataitem_stat_map[ idx ].str_stat;

    idx++;
  }
}
```

```
SMC_CHARP
LookupLockResult(
  SMC_LOCK_RESULT   value
  )
{
  typedef struct {
    SMC_CHARP         str_lock_res;
    SMC_LOCK_RESULT   enum_lock_res;
  } LOCK_RESULT_MAPPER;

  LOCK_RESULT_MAPPER lock_result_map[] = {
  { "granted",               SMC_LOCK_R_GRANTED },
  { "notneeded",             SMC_LOCK_R_NOTNEEDED },
  { "waited",                SMC_LOCK_R_WAITED },
  { "didntwait",             SMC_LOCK_R_DIDNTWAIT },
  { "deadlock",              SMC_LOCK_R_DEADLOCK },
  { "interrupted",           SMC_LOCK_R_INTERRUPTED},
  { (SMC_CHARP)0,  0 }
  };

  SMC_INT    idx = 0;
  SMC_BOOL   match = FALSE;

  while( match == FALSE)
  {
    if ( value == lock_result_map[ idx ].enum_lock_res )
      return lock_result_map[ idx ].str_lock_res;

    if (lock_result_map[ idx ].enum_lock_res == 0 )
      return lock_result_map[ idx ].str_lock_res;

    idx++;
  }
}
SMC_CHARP
LookupLockResultSummary(
  SMC_LOCK_RESULT_SUMMARY   value
  )
{
  typedef struct {
    SMC_CHARP                 str_lock_ressum;
    SMC_LOCK_RESULT_SUMMARY   enum_lock_ressum;
  } LOCK_RESULT_SUMMARY_MAPPER;

  LOCK_RESULT_SUMMARY_MAPPER lock_result_summary_map[] = {
  { "granted",               SMC_LOCK_RS_GRANTED },
```

```
  { "notgranted",              SMC_LOCK_RS_NOTGRANTED },
  { (SMC_CHARP)0,  0 }
  };

  SMC_INT    idx = 0;
  SMC_BOOL   match = FALSE;

  while( match == FALSE)
  {
    if ( value == lock_result_summary_map[ idx ].enum_lock_ressum )
      return lock_result_summary_map[ idx ].str_lock_ressum;

    if (lock_result_summary_map[ idx ].enum_lock_ressum == 0 )
      return lock_result_summary_map[ idx ].str_lock_ressum;

    idx++;
  }
}

SMC_CHARP
LookupLockStatus(
  SMC_LOCK_STATUS   value
  )
{
  typedef struct {
    SMC_CHARP         str_lock_status;
    SMC_LOCK_STATUS   enum_lock_status;
  } LOCK_STATUS_MAPPER;

  LOCK_STATUS_MAPPER lock_status_map[] = {
  { "held_blocking",              SMC_LOCK_S_HELD_BLOCKING },
  { "held_notblocking",           SMC_LOCK_S_HELD_NOTBLOCKING },
  { "requested_blocked",          SMC_LOCK_S_REQUESTED_BLOCKED },
  { "requested_notblocked",       SMC_LOCK_S_REQUESTED_NOTBLOCKED },
  { (SMC_CHARP)0,    0 }
  };

  SMC_INT    idx = 0;
  SMC_BOOL   match = FALSE;

  while( match == FALSE)
  {
    if ( value == lock_status_map[ idx ].enum_lock_status )
      return lock_status_map[ idx ].str_lock_status;

    if (lock_status_map[ idx ].enum_lock_status == 0 )
```

```
      return lock_status_map[ idx ].str_lock_status;

    idx++;
  }
}


SMC_CHARP
LookupLockType(
  SMC_LOCK_TYPE   value
  )
{
  typedef struct {
    SMC_CHARP       str_lock_type;
    SMC_LOCK_TYPE   enum_lock_type;
  } LOCK_TYPE_MAPPER;

  LOCK_TYPE_MAPPER lock_type_map[] = {
  { "ex_tab",              SMC_LOCK_T_EX_TAB },
  { "sh_tab",              SMC_LOCK_T_SH_TAB },
  { "ex_int",              SMC_LOCK_T_EX_INT },
  { "sh_int",              SMC_LOCK_T_SH_INT },
  { "ex_page",             SMC_LOCK_T_EX_PAGE },
  { "sh_page",             SMC_LOCK_T_SH_PAGE },
  { "upd_page",            SMC_LOCK_T_UP_PAGE },
  { (SMC_CHARP)0,  0 }
  };

  SMC_INT    idx = 0;
  SMC_BOOL   match = FALSE;

  while( match == FALSE)
  {
    if ( value == lock_type_map[ idx ].enum_lock_type )
      return lock_type_map[ idx ].str_lock_type;

    if (lock_type_map[ idx ].enum_lock_type == 0 )
      return lock_type_map[ idx ].str_lock_type;

    idx++;
  }
```

附 录 B

数据类型和结构

主题	页码
数据类型概述	227

数据类型概述

表 B-1 列出了带有说明的 Monitor Client Library 常量类型，及相应的 C 或 Open Client 数据类型。

表 B-1：数据类型概述

Monitor Client Library 数据类型	说明	相应的 C 或 Open Client 数据类型
SMC_ALARM_ACTION_TYPE	指定触发报警时所采取的操作类型	无
SMC_ALARM_ID	报警标识符	size_t
SMC_ALARM_IDP	报警标识符的指针	size_t*
SMC_BOOL	布尔型	int
SMC_CHAR	字符	char
SMC_CHARP	字符指针	char*
SMC_CHARPP	字符指针的指针	char**
SMC_CCHARP	字符指针常量	CS_CONST char*
SMC_CLOSE_TYPE	指定关闭 Adaptive Server Enterprise Monitor 连接时的选项	无
SMC_COMMAND_ID	命令标识符	size_t
SMC_COMMAND_IDP	命令标识符指针	size_t*
SMC_CONNECT_ID	连接标识符	size_t
SMC_CONNECT_IDP	连接标识符指针	size_t*
SMC_DATETIME	日期和时间	CS_DATETIME
SMC_DATAITEM_NAME	标识 Monitor Client Library 要获取的特定性能数据块	无
SMC_DATAITEM_NAMEP	指向 SMC_DATAITEM_NAME 的指针	无

Monitor Client Library 数据类型	说明	相应的 C 或 Open Client 数据类型
SMC_DATAITEM_STATTYPE	标识 Monitor Client Library 应在数据上执行的规范化 （如果有的话）	无
SMC_DATAITEM_STRUCT	标识 Monitor Client Library 要获取的数据	无
SMC_DATAITEM_STRUCTP	指向 SMC_DATAITEM_STRUCT 的指针	无
SMC_DATAITEM_TYPE	标识 Monitor Client Library 获取的数据的数据类型	无
SMC_DATAITEM_TYPEP	指向 SMC_DATAITEM_TYPE 的指针	无
SMC_DOUBLE	双精度浮点数	double
SMC_DOUBLEP	双精度指针	double*
SMC_ERR_SEVERITY	指示错误的严重程度	无
SMC_FILTER_ID	过滤器标识符	size_t
SMC_FILTER_IDP	指向过滤器标识符的指针	size_t*
SMC_FILTER_TYPE	指定用 smc_create_filter 创建的过滤器类型	无
SMC_HS_ESTIM_OPT	指定在历史性能数据回放中，如果不能从可用的记录数据中可靠地计算出数据，是否允许估计数据。	无
SMC_HS_MISSDATA_OPT	指定在历史性能数据回放中，对无可用数据的时间段，是否返回采样数据。	无
SMC_HS_PLAYBACK_OPT	指定是否规范化和 / 或汇总历史性能数据。	无
SMC_HS_SESS_DELETE_OPT	指定是否删除与 Historical Server 会话相关联的数据文件	无
SMC_HS_SESS_ERR_OPT	指定错误发生后，是否继续记录会话	无
SMC_HS_SESS_PROT_LEVEL	指定记录会话中的数据是否可被其他用户访问	无
SMC_HS_SESS_SCRIPT_OPT	指定是否应创建脚本，以创建与记录会话中视图相对应的表	无
SMC_HS_TARGET_OPT	指定是否将历史性能数据回放发送到客户端应用程序，或用于创建新会话	无
SMC_INFO_TYPE	指定在对 smc_get_command_info 的调用中请求的信息类型	无
SMC_INT	整型	int
SMC_INTP	整型指针	int*
SMC_LOCK_RESULT	标识锁请求的可能结果	无
SMC_LOCK_RESULT_SUMMARY	标识锁请求结果的两个主要类别	无
SMC_LOCK_STATUS	标识锁或锁请求的可能状态	无
SMC_LOCK_TYPE	标识锁的粒度和互斥性	无
SMC_LONG	长整型	long
SMC_LONGP	长整型指针	long*

Monitor Client Library 数据类型	说明	相应的 C 或 Open Client 数据类型
SMC_OBJ_TYPE	标识 Adaptive Server 数据库中的对象类型	无
SMC_PROC_STATE	标识 Adaptive Server 进程的可能状态	无
SMC_PROP_ACTION	指定在 smc_connect_props 调用中要采取的操作	无
SMC_PROP_TYPE	指定 smc_connect_props 调用的对象属性	无
SMC_RETURN_CODE	指示 Monitor Client Library 操作是否成功，若失败，发生什么样的错误	无
SMC_SERVER_MODE	指定 Adaptive Server Enterprise Monitor 连接应获取活动性能数据，还是处理历史数据	无
SMC_SESSION_ID	会话标识符	size_t
SMC_SESSION_IDP	会话标识符指针	size_t*
SMC_SIZET	无符号整型数	size_t
SMC_SIZETP	无符号整型数指针	size_t*
SMC_SOURCE	指示检测错误的软件层	无
SMC_VALUE_UNION	包含数据的结构	无
SMC_VALUE_UNIONP	指向 SMC_VALUE_UNION 的指针	无
SMC_VIEW_ID	视图标识符	size_t
SMC_VIEW_IDP	视图标识符指针	size_t*
SMC_VOID	无类型	无类型
SMC_VOIDP	空指针	void*

此附录的余下部分，描述了 C 或 Open-Client Client Library 中没有的个别数据类型。

枚举：SMC_ALARM_ACTION_TYPE

确定触发报警时所采取的操作类型的枚举：

表 B-2：报警操作类型

SMC_ALARM_A_EXEC_PROC
SMC_ALARM_A_LOG_TO_FILE
SMC_ALARM_A_NOTIFY

枚举：SMC_CLOSE_TYPE

用于确定关闭命令范围的枚举：

表 B-3：关闭类型

SMC_CLOSE_REQUEST

枚举：SMC_DATAITEM_NAME

与 smc_create_view 联合使用，指定性能数据的枚举。有关可用数据项的列表，请参见第 2 章 “数据项和统计类型”。

枚举：SMC_DATAITEM_STATTYPE

与 smc_create_view 联合使用，确定性能数据的统计类型和累计间隔的枚举。

表 B-4：数据项统计类型

SMC_STAT_VALUE_SAMPLE
SMC_STAT_VALUE_SESSION
SMC_STAT_RATE_SAMPLE
SMC_STAT_RATE_SESSION
SMC_STAT_AVG_SAMPLE
SMC_STAT_AVG_SESSION

结构：SMC_DATAITEM_STRUCT

与 smc_create_view 联合使用，确定性能数据的结构。

typedef struct SMC_DATAITEM_STRUCT{	
SMC_DATAITEM_NAME	*dataItemName*
SMC_DATAITEM_STATTYPE	*dataItemStatType*
} SMC_DATAITEM_STRUCT;	

枚举：SMC_DATAITEM_TYPE

与 smc_get_dataitem_type 联合使用，确定性能数据结果的物理类型的枚举：

表 B-5：数据项类型

SMC_DI_TYPE_NONE
SMC_DI_TYPE_CHARP
SMC_DI_TYPE_DATIM
SMC_DI_TYPE_DOUBLE
SMC_DI_TYPE_ENUMS
SMC_DI_TYPE_INT
SMC_DI_TYPE_LONG

枚举：SMC_ERR_SEVERITY

与 smc_get_command_info 联合使用，确定错误、警告或信息提示严重性的枚举。

表 B-6：错误严重性

SMC_ERR_SEV_INFO
SMC_ERR_SEV_WARN
SMC_ERR_SEV_FATAL

枚举：SMC_FILTER_TYPE

标识过滤器类型的枚举：

表 B-7：过滤器类型

SMC_FILT_T_EQ
SMC_FILT_T_NEQ
SMC_FILT_T_GE
SMC_FILT_T_LE
SMC_FILT_T_GE_AND_LE
SMC_FILT_T_TOP_N

枚举：SMC_HS_ESTIM_OPT

指定回放会话期间是否允许估计某些数据的枚举。

表 B-8：Historical Server 错误操作

SMC_HS_ESTIM_ALLOW
SMC_HS_ESTIM_DISALLOW

枚举：SMC_HS_MISSDATA_OPT

指定在回放会话期间，若给定采样没有要回放的性能数据， Historical Server 应采取何种操作的枚举：

表 B-9：Historical Server 丢失数据选项

SMC_HS_MISSDATA_SHOW
SMC_HS_MISSDATA_SKIP

枚举：SMC_HS_PLAYBACK_OPT

指定是否规范化和 / 或汇总回放会话的数据的枚举。

表 B-10：Historical Server 保护级别

SMC_HS_PBTYPE_ENTIRE
SMC_HS_PBTYPE_ACTUAL
SMC_HS_PBTYPE_INTERVAL
SMC_HS_PBTYPE_RAW

枚举：SMC_HS_SESS_DELETE_OPT

指定是否删除与 Historical Server 连接相关数据文件的枚举。

表 B-11：Historical Server 文件删除选项

SMC_HS_SESS_DELETE_FILES
SMC_HS_SESS_RETAIN_FILES

枚举：SMC_HS_SESS_ERR_OPT

指定若记录会话遇到非致命错误时， Historical Server 应采用何种操作的枚举：

表 B-12：Historical Server 错误选项

SMC_HS_SESS_ERR_CONT
SMC_HS_SESS_ERR_HALT

枚举：SMC_HS_SESS_PROT_LEVEL

指定访问 Historical Server 记录的性能数据的保护级别的枚举：

表 B-13：Historical Server 保护级别

SMC_HS_SESS_PROT_PRIVATE
SMC_HS_SESS_PROT_PUBLIC

枚举：SMC_HS_SESS_SCRIPT_OPT

指定脚本 （若有）类型的枚举，脚本由 Historical Server 创建，以帮助用户处理记录会话的性能数据：

表 B-14：Historical Server 脚本选项

SMC_HS_SESS_SCRIPT_SYBASE
SMC_HS_SESS_SCRIPT_NONE

枚举：SMC_HS_TARGET_OPT

指定回放会话是否为应用程序返回数据，或回放是否在 Historical Server 上创建新会话的枚举：

表 B-15：Historical Server 脚本选项

SMC_HS_TARGET_CLIENT
SMC_HS_TARGET_FILE

枚举：SMC_HS_TARGET_OPT

指定回放会话内数据目标的枚举：

表 B-16：Historical Server 回放目标选项

SMC_HS_TARGET_CLIENT
SMC_HS_TARGET_FILE

枚举：SMC_INFO_TYPE

确定可用于从回调函数进行查询的各种数据的枚举 （使用 smc_get_command_info）：

表 B-17：信息类型

SMC_INFO_ALARM_ACTION_DATA
SMC_INFO_ALARM_ALARMID
SMC_INFO_ALARM_CURRENT_VALUE
SMC_INFO_ALARM_DATAITEM
SMC_INFO_ALARM_ROW
SMC_INFO_ALARM_THRESHOLD_VALUE
SMC_INFO_ALARM_TIMESTAMP
SMC_INFO_ALARM_VALUE_DATATYPE
SMC_INFO_ALARM_VIEWID
SMC_INFO_ERR_MAPSEVERITY
SMC_INFO_ERR_MSG
SMC_INFO_ERR_NUM
SMC_INFO_ERR_SEVERITY
SMC_INFO_ERR_SOURCE
SMC_INFO_ERR_STATE

枚举：SMC_LOCK_RESULT

标识锁请求结果的枚举：

表 B-18：锁结果类型

SMC_LOCK_R_GRANTED
SMC_LOCK_R_NOTNEEDED
SMC_LOCK_R_WAITED
SMC_LOCK_R_DIDNTWAIT
SMC_LOCK_R_DEADLOCK
SMC_LOCK_R_INTERRUPTED

枚举：SMC_LOCK_RESULT_SUMMARY

确定是否授予锁请求的枚举：

表 B-19：锁结果摘要类型

SMC_LOCK_RS_GRANTED
SMC_LOCK_RS_NOTGRANTED

枚举：SMC_LOCK_STATUS

标识锁状态的枚举：

表 B-20：锁状态类型

SMC_LOCK_S_HELD_BLOCKING
SMC_LOCK_S_HELD_NOTBLOCKING
SMC_LOCK_S_REQUESTED_BLOCKED
SMC_LOCK_S_REQUESTED_NOTBLOCKED

枚举：SMC_LOCK_TYPE

标识锁类型的枚举：

表 B-21：锁类型

SMC_LOCK_T_EX_TAB
SMC_LOCK_T_SH_TAB
SMC_LOCK_T_EX_INT
SMC_LOCK_T_SH_INT
SMC_LOCK_T_EX_PAGE
SMC_LOCK_T_SH_PAGE
SMC_LOCK_T_UP_PAGE

枚举：SMC_OBJ_TYPE

标识对象类型的枚举：

表 B-22：对象类型

SMC_OBJ_T_STP
SMC_OBJ_T_TBL

枚举：SMC_PROC_STATE

标识进程状态的枚举：

表 B-23：进程状态

SMC_PROC_STATE_ALARM_SLEEP
SMC_PROC_STATE_BACKGROUND
SMC_PROC_STATE_BAD_STATUS
SMC_PROC_STATE_INFECTED
SMC_PROC_STATE_LOCK_SLEEP
SMC_PROC_STATE_RECV_SLEEP
SMC_PROC_STATE_RUNNABLE
SMC_PROC_STATE_RUNNING
SMC_PROC_STATE_SEND_SLEEP
SMC_PROC_STATE_SLEEPING

SMC_PROC_STATE_STOPPED
SMC_PROC_STATE_TERMINATING
SMC_PROC_STATE_YIELDING
SMC_PROC_STATE_REMOTE_IO
SMC_PROC_STATE_SYNC_SLEEP

枚举：SMC_PROP_ACTION

用于确定 smc_connect_props 函数调用所需操作的枚举：

表 B-24：连接属性操作

SMC_PROP_ACT_SET
SMC_PROP_ACT_GET
SMC_PROP_ACT_CLEAR

枚举：SMC_PROP_TYPE

用于确定在 smc_connect_props 调用中对其操作的属性的 枚举：

表 B-25：连接属性

SMC_PROP_APPNAME
SMC_PROP_ERROR_CALLBACK
SMC_PROP_IFILE
SMC_PROP_LOGIN_TIMEOUT
SMC_PROP_PACKETSIZE
SMC_PROP_PASSWORD
SMC_PROP_SERVERMODE
SMC_PROP_SERVERNAME
SMC_PROP_TIMEOUT
SMC_PROP_USERDATA
SMC_PROP_USERNAME

枚举：SMC_RETURN_CODE

标识返回码类型的枚举：

表 B-26：返回码

SMC_RET_SUCCESS
SMC_RET_FAILURE
SMC_RET_INSUFFICIENT_MEMORY
SMC_RET_CONNECTION_ERROR
SMC_RET_UNABLE_TO_CONNECT_TO_SMS
SMC_RET_UNABLE_TO_CONNECT_TO_SS
SMC_RET_MISSING_RESULT_TABLE
SMC_RET_INVALID_USER_PASSWD
SMC_RET_INVALID_PARAMETER
SMC_RET_INVALID_CACHE
SMC_RET_INVALID_DCID
SMC_RET_INVALID_COMMAND
SMC_RET_INVALID_VIEWID
SMC_RET_INVALID_DINAME
SMC_RET_INVALID_DISTAT
SMC_RET_INVALID_DI_STRUCT
SMC_RET_DI_STAT_MISMATCH
SMC_RET_INVALID_DI_COMBO
SMC_RET_INVALID_DATATYPE
SMC_RET_INVALID_VALUE_COUNT
SMC_RET_INVALID_FILTER_VALUE
SMC_RET_INVALID_FILTER_RANGE
SMC_RET_DATAITEM_CONTAINS_FILTER
SMC_RET_INVALID_COMPOSITE_FILTER
SMC_RET_INVALID_SVR_MODE
SMC_RET_MISSING_DATAITEM
SMC_RET_INVALID_FILTERID
SMC_RET_INVALID_ALARMID
SMC_RET_INVALID_ALARM_VALUE
SMC_RET_INVALID_DINAME_FOR_ALARM
SMC_RET_INVALID_API_FUNC_SEQUENCE
SMC_RET_INVALID_API_FUNCTION
SMC_RET_INVALID_PROPERTY
SMC_RET_INVALID_INFOTYPE

SMC_RET_CONNECT_NOT_CLOSED
SMC_RET_ARITHMETIC_OVERFLOW
SMC_RET_LOGIN_LACKS_SA_ROLE
SMC_RET_INTERNAL_ERROR

枚举：SMC_SERVER_MODE

标识 Adaptive Server Enterprise Monitor 连接类型的枚举：

表 B-27：服务器模式类型

SMC_SERVER_M_LIVE
SMC_SERVER_M_HISTORICAL

枚举：SMC_SOURCE

与 ErrorCallback 联合使用，确定错误、警告或信息提示的来源的枚举。

表 B-28：错误源

SMC_SRC_UNKNOWN
SMC_SRC_HS
SMC_SRC_SMC
SMC_SRC_CT
SMC_SRC_SS
SMC_SRC_SMS

联合：SMC_VALUE_UNION

与 smc_connect_props、 smc_get_command_info 和 smc_get_dataitem_value 联合使用，设置并检索结果的联合。

typedef union SMC_VALUE_UNION {	
SMC_INT	*intValue*
SMC_LONG	*longValue*
SMC_DOUBLE	*doubleValue*
SMC_SIZET	*sizetValue*
SMC_CHARP	*stringValue*
SMC_VOIDP	*voidpValue*
SMC_DATETIME	*datetimeValue*
} SMC_VALUE_UNION;	

附 录 C

向后兼容性

Monitor Client Library 版本 11.5 和更高版本替换了几个 API 函数。新 API 函数和回调函数的功能更强、扩展性更好。考虑到向后兼容性，已经将被替换的 API 函数和回调函数保存在函数库中。

旧函数和替换函数

表 C-1 将 Monitor Client Library 的旧函数与其替换函数对应起来：

表 C-1：旧函数和替换函数

旧函数	替换函数
smc_change_error_handler	smc_connect_props
smc_connect	smc_connect_alloc
	smc_connect_props
	smc_connect_ex
smc_create_alarm	smc_create_alarm_ex
smc_disconnect	smc_close
	smc_connect_drop
smc_refresh	smc_refresh_ex

旧函数和替换函数之间的最主要语法区别在于回调函数的参数。在早期版本中， SMC_CALLBACK、 SMC_ALARM_CALLBACK 和 SMC_ERR_CALLBACK 用于指定回调函数。这些回调函数类型已被 SMC_GEN_CALLBACK 取代。

注释 与旧函数 smc_refresh 不同，刷新函数 smc_refresh_ex 不使用任何回调函数。

除更改回调函数类型以外，还用一系列灵活性和可控性更强的函数替换了 smc_connect 和 smc_disconnect。

自 Adaptive Server 版本 11.5 开始提供的新函数

表 C-2 列出了这些函数。

表 C-2：新函数

smc_create_playback_session
smc_get_command_info
smc_initiate_playback
smc_terminate_playback
smc_terminate_recording

注释 新函数与旧函数不能一起使用。

函数和回调兼容性规则

使用以下规则决定哪些函数和回调可以一起使用：

- 如果正在使用新函数或替换函数，请不要使用旧函数。
- 如果正使用旧函数，请使用旧的错误回调函数类型。
- 如果正使用替换函数或新函数，请使用 11.1 版本的错误回调函数类型。
- 可以将未修改的函数与所有其它类型的函数一起使用。

附 录 D

疑难解答信息和错误消息

故障排除

来自 Adaptive Server 的不明错误消息

如果创建的视图需要某个数据库的信息，而该数据库需要恢复，您将从 Adaptive Server 收到错误消息，而不是从 Monitor Client Library 收到简明的错误消息。

视图刷新失败

- 如果在某人创建数据库的同时，你试图去刷新一个视图，刷新可能会失败。
- 如果 Adaptive Server 上的一个或多个数据库处于单用户模式，刷新视图可能会失败。

对象 ID 为负数

如果使用 SMC_NAME_OBJ_ID 数据项创建视图，您可能会发现对象 ID 为负数。负的对象 ID 如实反映了 Adaptive Server 分配的 ID。

Monitor Server 会报告*所有的*活动，包括 Adaptive Server 创建的用来执行复杂查询的临时表上的活动。Adaptive Server 分配给临时表的对象 ID 可为正，也可为负。 Adaptive Server 分配的对象 ID 是报告的内容。

在显示 SMC_NAME_OBJ_NAME 的视图中，将用字符串 **TempObject** 表示临时表的报告结果。

错误消息

Monitor Client Library 是 Open Server 应用程序，它使用 Open Client Library 与 Adaptive Server 和 Monitor Server 进行通信。其中的任何组件都能检测并报告错误条件。 Monitor Client Library 也能检测并报告错误条件，并将错误条件记入客户端和 / 或报告给客户端。

可能会报告下列生成、链接和编译错误消息。这些错误消息按字母顺序在此列出。

Communication failure:check if server is running

运行 testmon.exe 程序时，下列任一情况都会导致报告这一错误：

- *example.h* 文件中的服务器名称不对。
- *sql.ini* 文件丢失。
- *sql.ini* 文件中有错误的网络连接信息。
- Adaptive Server 现在没有运行。
- Historical Server 未运行。
- *example.h* 文件中的用户名设置错误。
- *example.h* 中用户名对应的口令设置错误。

Configuration failure:possibly missing *interfaces* file or bad login parameters

运行 testmon.exe 程序时，下列任一情况都会导致报告这一错误：

- *example.h* 文件中的服务器名称不对。
- *sql.ini* 文件丢失。
- *sql.ini* 文件中有错误的网络连接信息。
- Adaptive Server 现在没有运行。
- Historical Server 未运行。
- *example.h* 文件中的用户名设置错误。
- *example.h* 中用户名对应的口令设置错误。

Don’t know how to build *example.h*

生成 testmon.exe 时，下列任一情况都会导致报告这一编译错误：

- 项目必须重新生成所有的从属文件。
- 项目的 *include* 文件路径需要包含文件名的位置信息。
- 缺省位置为 *%SYBASE%\OCS-15_0\INCLUDE* 和 *%SYBASE%\OCS-15_0\SAMPLE\MONCLT\TESTMON*。

error L2029: ‘SMC_CONNECT’ : unresolved external

生成 testmon.exe 时，以下情况会导致报告此链接错误：

- 必须包含 *smcapi32.lib* 作为要链接的库之一。缺省情况下，它位于 *%SYBASE%\OCS-15_0\LIB* 中。

error L2029: ‘SMC_CREATE_VIEW’ : unresolved external

生成 testmon.exe 时，以下情况会导致报告此链接错误：

- 包含 *smcapi32.lib* 作为要链接的库之一。缺省情况下，它位于 *%SYBASE%\OCS-15_0\LIB* 中。

fatal error C1083: Cannot open include file: ‘cstypes.h’: No such file or directory

生成 testmon.exe 时，下列任一情况都会导致报告这一编译错误：

- 项目必须重新生成所有的从属文件。
- 项目的 *include* 文件路径需要包含文件名的位置信息。
- 缺省位置为 *%SYBASE%\OCS-15_0\INCLUDE* 和 *%SYBASE%\OCS-15_0\SAMPLE\MONCLT\TESTMON*。

fatal error C1083: Cannot open include file: ‘mcpublic.h’: No such file or directory

生成 testmon.exe 时，以下情况会导致报告这一编译错误：

- 必须编辑项目的预处理程序 *include* 路径，使它设置正确。它应包含 *%SYBASE%\OCS-15_0\INCLUDE*。

LINK:fatal error L4051:smcapi32.lib :cannot find library

生成 testmon.exe 时，以下情况会导致报告此链接错误：

- 项目的库文件路径必须包含 *smcapi32.lib* 的位置，该文件应该位于 *%SYBASE%\OCS-15_0\LIB* 中。

索引

符号

英文

B

C

D

F

G

H

J

K

L

M

P

Q

S

T

X

Y

Z