IBM solidDB
IBM solidDB Universal Cache

V6.3

链接库访问用户指南

Information Management software

S151-1147-00

IBM solidDB
IBM solidDB Universal Cache

V6.3

链接库访问用户指南

Information Management software

S151-1147-00

声明

在使用本资料及其支持的产品之前，请阅读第 65 页的『声明』中的信息。

本版本适用于 IBM solidDB（产品编号 5724-V17）和 IBM solidDB Universal Cache（产品编号 5724-W91）V6R3 及所有后续发行版和修订版，直到在新版本中另有声明为止。

目录

图 v

表 vii

关于本手册 ix
印刷约定 ix
语法表示法约定 x

1 链接库访问简介 1
链接库访问库 2
基于磁盘的服务器与无盘服务器 3
库的内容 3
与链接库访问配合使用的应用程序类型 4
用于链接库访问的 solidDB 客户机 API 和驱动程序 . 5
solidDB SA API 6
solidDB ODBC API 6
solidDB JDBC API 7
solidDB 控制 API（SSC API） 7

2 创建和运行链接库访问应用程序 9
访问链接库访问库 9
用于远程应用程序的库 9
样本 C 应用程序 10
使用数据同步 10
为链接库访问链接应用程序 11
为链接库访问准备用户应用程序 12
与具有链接库访问的 solidDB 建立本地或远程连接 14
启动和停止 solidDB 链接库访问 15
使用控制 API 函数 SSCStartServer 显式启动 . . 16
使用 ODBC API 函数调用 SQLConnect 来隐式启动 18
使用 SA API 函数调用 SaConnect 来隐式启动 . . 20
关闭 solidDB 链接库访问 20
隐式启动配置参数 21

3 对控制 API 的描述 23
检索任务信息 23
特殊事件的通知函数 23
获取 solidDB 状态和服务器信息 23
控制 API 函数总结 24
控制 API 和等价的 ADMIN COMMAND 25
控制 API 引用 25
函数摘要 25
返回值 26
控制 API 错误代码和消息 27
SSCGetServerHandle 27
摘要 27
注释 27
返回值 28
SSCGetStatusNum 28
摘要 28
注释 28
返回值 28
SSCIsRunning 29
摘要 29
返回值 29
注释 29
SSCIsThisLocalServer 29
摘要 29
返回值 29
注释 29
SSCRegisterThread 30
摘要 30
返回值 30
注释 30
另请参阅 30
SSCSetCipher 30
摘要 30
返回值 32
注释 32
SSCSetNotifier 33
摘要 33
返回值 35
注释 35
示例 35
SSCSetState 36
摘要 36
返回值 37
注释 37
SSCStartDisklessServer 37
摘要 37
SSCStartDisklessServer 参数选项 38
返回值 39
注释 39
示例 39
另请参阅 40
SSCStartServer 40
摘要 40
返回值 41
注释 43
另请参阅 43
SSCStopServer 43
摘要 44
返回值 44
注释 44
另请参阅 45
SSCUnregisterThread 45
摘要 45
返回值 45
注释 45
另请参阅 45

4 使用无盘功能 47
无盘服务器的配置参数 47
无盘服务器中使用的参数 47
不适用于无盘引擎的配置参数 49

5 将 solidDB 链接库访问与 Java 语言配合使用 51
solidDB JDBC 加速器（SJA）概述 51
加速器的工作方式 51
系统需求 53
基本用法 53
安装 53
编译和运行程序 53
建立 JDBC 连接 54
局限性 55
solidDB 服务器控制（SSC）API 55

附录. 链接库访问参数 59
链接库访问参数 59
[Accelerator] 节 59

索引 61

声明 65

图

1. 具有链接库访问的 solidDB 2
2. 链接至 solidDB 4
3. 具有链接库访问的 solidDB – API 7

表

1. 印刷约定. ix
2. 语法表示法约定 x
3. 链接库访问系统库 12
4. 库文件 12
5. SSCStartServer 参数 16
6. SSCStartServer argv 选项 17
7. 控制 API 函数总结 24
8. 控制 API 参数用法类型 25
9. 控制 API 函数的错误代码和消息 27
10. SSCGetStatusNum 参数 28
11. SSCIsRunning 参数 29
12. SCCRegisterThread 参数 30
13. SSCSetCipher 参数 31
14. SSCSetNotifier 函数参数 33
15. SSCSetState 函数参数 36
16. SSCStartDisklessServer 参数 38
17. *argv* 参数的命令行选项. 38
18. SSCStartServer 参数 40
19. SSCStopServer 参数 44
20. SCCUnregisterThread 参数 45
21. 不适用于无盘引擎的配置参数. 49
22. 加速器参数. 59

关于本手册

IBM® solidDB® 链接库访问是一种能够提供更高性能的 solidDB 数据管理解决方案。为了避免产生网络延迟，solidDB 可执行文件和用户应用程序在同一程序空间中建立链接以生成单个可执行文件。通过将网络连接和远程过程调用（RPC）替换为本地函数调用可以显著提高性能。

本指南包含特定于链接库访问的信息。本指南对《IBM solidDB 管理员指南》中包含的信息进行了补充，它包括有关管理和维护 solidDB 的详细信息。

本指南假定您具备 C 编程语言的应用知识、一般的 DBMS 知识、熟悉 SQL 语言并且具备 solidDB 数据管理产品知识（例如，solidDB 内存数据库或者 solidDB 基于磁盘的引擎）。如果您要使用 solidDB Java™ 加速器，那么本指南还假定您具备 Java 的应用知识。

印刷约定

solidDB 文档使用下列印刷约定:

表 1. 印刷约定

格式	适用于
数据库表	此字体用于所有普通文本。
NOT NULL	采用此字体的大写字母指示 SQL 关键字和宏名称。
solid.ini	这些字体指示文件名和路径表达式。
`SET SYNC MASTER YES;` `COMMIT WORK;`	此字体用于程序代码和程序输出。示例 SQL 语句也使用此字体。
run.sh	此字体用于样本命令行。
TRIG_COUNT()	此字体用于函数名。
java.sql.Connection	此字体用于接口名称。
LockHashSize	此字体用于参数名、函数自变量和 Windows® 注册表条目。
argument	此类强调词指示用户或应用程序必须提供的信息。
管理指南	这种样式用于引用其他文档或者同一文档中的章节。新术语和强调的问题也按此样式书写。
文件路径表示	文件路径按 UNIX® 格式提供。斜杠（/）字符表示安装根目录。

表 1. 印刷约定 (续)

格式	适用于
操作系统	如果文档包含有关操作系统之间的差别的内容，那么首先提到的是 UNIX 格式。Microsoft® Windows 格式位于 UNIX 格式之后并括在括号中。其他操作系统将单独列出。对于不同的操作系统还可能有不同的章节进行描述。

语法表示法约定

solidDB 文档使用下列语法表示法约定:

表 2. 语法表示法约定

格式	适用于
`INSERT INTO table_name`	语法描述采用此字体。可替换部分采用*此*字体。
solid.ini	此字体指示文件名和路径表达式。
[ ]	方括号指示可选项；如果是粗体文本，那么必须将方括号包含在语法中。
|	竖线，用于将语法行中的两个互斥选项分隔开。
{ }	大括号用于对语法行中的一组互斥选项进行定界；如果是粗体文本，那么必须将大括号包括在语法中。
...	省略号指示可以多次重复使用自变量。
. . .	由三个点组成的一列表示这是先前代码行的延续。

1 链接库访问简介

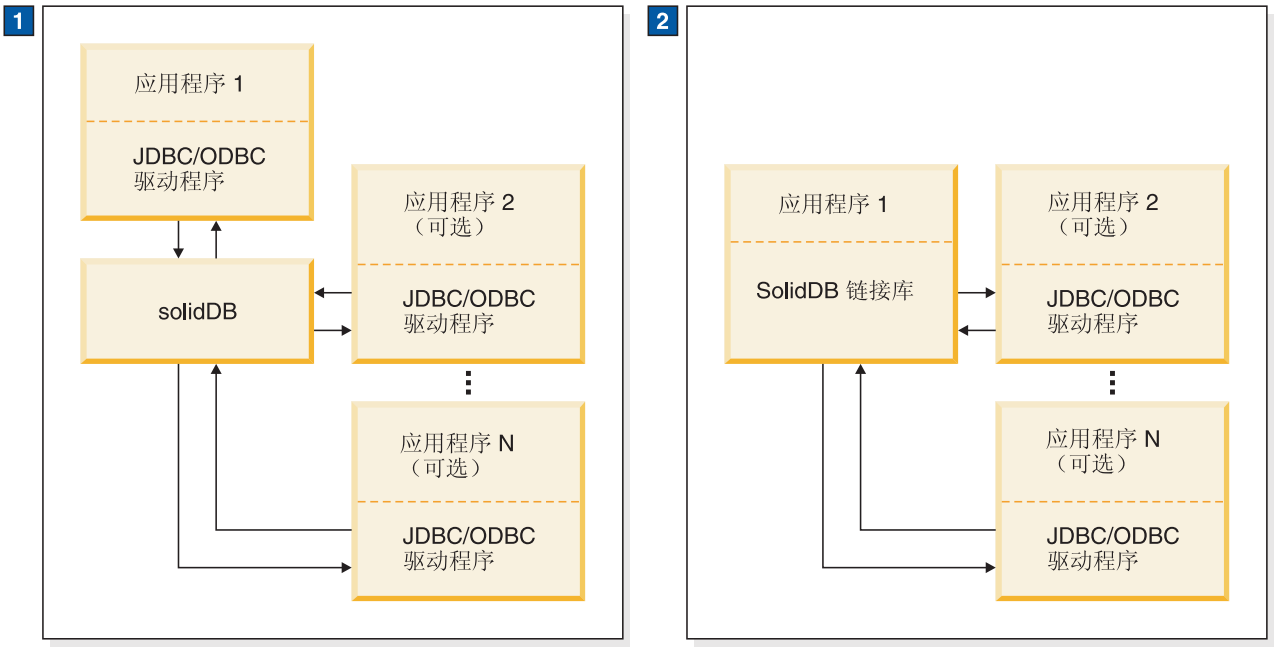
IBM® solidDB 链接库访问是一个函数库，它与 solidDB 提供相同的功能和接口。用户应用程序可以链接至此库。已链接的应用程序通过直接调用函数来实现与服务器通信，从而节省了客户机与服务器之间通过诸如 TCP/IP 等网络协议进行通信时所需要的开销。将应用程序和服务器链接到单个可执行程序中可以获得更高性能。

应用程序不必经过重新编写就可以使用链接库访问库。例如，您不必调用专用函数（用于启动和停止数据库服务器的函数除外）。您的应用程序可以继续使用它经常使用的 ODBC 函数调用。当链接库访问库已链接至应用程序时，将直接对服务器（无需通过网络）调用这些 ODBC 函数。

应用程序还可以访问其他某些链接库访问函数调用，以执行诸如在服务器中调度任务等操作。但是，除非您自己想使用这些函数调用，否则并不是必须使用它们。

在应用程序链接至服务器之后，并不意味着只有已链接的应用程序才能使用服务器。不仅“本地”客户机应用程序（已直接链接至链接库访问库的应用程序）可以访问正在作为链接库访问函数库执行的 solidDB 服务器，“远程”客户机应用程序（通过诸如 TCP/IP 等通信协议连接至服务器的应用程序）也可以访问此服务器。远程客户机认为链接库访问服务器与任何其他 solidDB 服务器相似，而本地客户机认为它是速度更快、更准确并且可控制的 solidDB 服务器。

注: 尽管“远程”应用程序通常与服务器在不同的计算机中运行，但是，如果一个应用程序使用网络通信协议与服务器通信，那么即使此客户机与数据库服务器在同一台计算机上运行，也认为它是一个“远程”应用程序。



1. 在标准 solidDB 数据库配置中，应用程序和服务器是独立的程序。
2. solidDB 链接库是一个联结到应用程序中的子例程库。其他应用程序可能也与该服务器通信。

图 1. 具有链接库访问的 solidDB

上图显示了一个使用链接库访问库的样本 solidDB。

注:

本地应用程序请求是通过 solidDB SA API 或 ODBC API 直接函数调用处理的。对于本地应用程序，链接库访问还提供了一个控制 API，此 API 处理用于控制 solidDB 后台进程和客户机任务的本地请求。您还可以对链接库访问使用 JDBC 调用。请参阅第 51 页的5 章，『将 solidDB 链接库访问与 Java 语言配合使用』。

正如您在上图中所看到的，远程客户机通过链接至客户机应用程序的 ODBC 或 JDBC 驱动程序来进行通信，而本地客户机应用程序不需要任何远程通信驱动程序。

链接库访问库

在标准（非链接库访问）solidDB 配置中，应用程序（“客户机”）和数据库引擎（“服务器”）是独立的进程，它们通过网络协议进行通信。客户机必须链接至通信驱动程序（例如，ODBC 或 JDBC 驱动程序），这些通信驱动程序通过网络与数据库服务器通信。

通过链接库访问，应用程序链接至一个包含完整数据库服务器功能的静态库（例如，UNIX 系统中的 .lib 或 .a 文件）。这就意味着 solidDB 与应用程序在同一个可执行文件中运行，因此不需要通过网络传输数据。链接至链接库访问库的应用程序还可以使用 ODBC API 和 SA API 建立多个连接。这两个 API 都是可重入的，因此允许同时建立来自不同线程的连接。

直接链接至链接库访问库的用户应用程序还可以创建与其他数据库服务器的远程连接。传递给 ODBC API 或 SA API 连接函数的连接字符串用于定义连接类型是本地连接还是远程连接。

有关如何链接应用程序的详细信息，请阅读第 11 页的『为链接库访问链接应用程序』。

当您启动应用程序时，只有此应用程序中的代码才会开始自动运行。服务器代码在很大程度上与您的应用程序代码无关，您必须通过调用函数来显式启动服务器。（在大多数或所有实现中，服务器在与应用程序所使用线程不同的线程上运行。通过调用函数来启动服务器时，将执行服务器代码所需要的任何初始化步骤，必要时还将创建适当的附加线程，并启动在这些线程上运行的服务器。）

基于磁盘的服务器与无盘服务器

链接库访问库包含两个不同的函数调用来启动服务器。其中一个函数调用将启动正常的服务器（也就是基于磁盘的服务器），而另一个函数调用将启动不使用磁盘驱动器的服务器。有关更多信息，请参阅第 47 页的4 章，『使用无盘功能』以及对于 SSCStartServer 和 SSCStartDisklessServer 函数的描述。

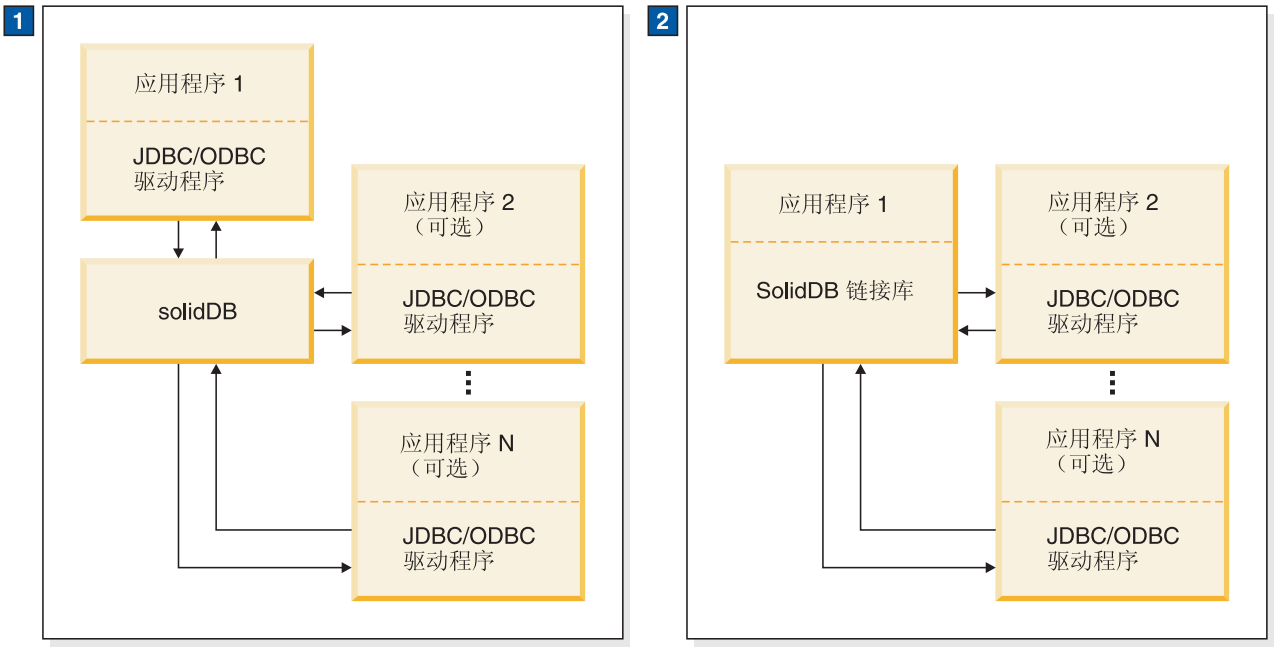
库的内容

链接库访问库包含用于三个独立 API 的函数:

- solidDB 控制 API (SSC API) 库，此库包含用于控制任务调度的函数。
- solidDB ODBC Driver 函数，此函数允许与服务器库直接通信而不需要通过网络。
- solidDB SA API 库，此库对于其他使用链接库访问的功能可能是必需的。例如，此库允许从表中插入、删除和选择记录。

因为应用程序通过所有这三个 API（SSC、SA 和 ODBC）链接至库，所以应用程序可以调用这些 API 的任意组合中的函数。有关每个 API 的详细信息，请阅读第 5 页的『用于链接库访问的 solidDB 客户机 API 和驱动程序』。

注: 远程应用程序也可以访问这三个 API（SSC、SA 和 ODBC）。但是，对于远程应用程序来说，这三个 API 的函数并不是全部在同一个文件中。有关远程应用程序和具有双重角色的应用程序的详细信息，请阅读第 4 页的『与链接库访问配合使用的应用程序类型』。有关用于远程应用程序的 API 文件的信息，请阅读第 5 页的『用于链接库访问的 solidDB 客户机 API 和驱动程序』。



1. 在标准 solidDB 数据库配置中，应用程序和服务器是独立的程序。
2. solidDB 链接库是一个链接到应用程序中的子例程库。其他应用程序可能也与服务器通信。

图 2. 链接至 solidDB

与链接库访问配合使用的应用程序类型

链接库访问应用程序位于服务器本地；服务器和应用程序组合成为单个程序。调用 ODBC 函数时实际上是直接访问服务器，而不是通过 ODBC 驱动程序和通信协议（例如，TCP/IP）来访问服务器。

除了处理来自与链接库访问库相链接的本地应用程序的请求以外，服务器还将处理来自远程应用程序的请求。

远程应用程序并未链接至链接库访问库。它是一个独立的可执行文件，必须通过使用网络连接（例如，TCP/IP）或者其他连接（例如，共享内存）才能与服务器通信。远程应用程序通常（但并不始终都是）与服务器在不同的计算机上运行。但是，单台计算机可以在运行链接库访问本地应用程序的同时将一个或多个远程应用程序作为独立进程来运行。

大多数应用程序是本地应用程序（即，链接至单个可执行文件中的链接库访问库的应用程序）或者远程应用程序（从不链接至链接库访问库的应用程序）。但是，也可以编写一个既可以作为本地应用程序也可以作为远程应用程序的应用程序；它根据其编译和链接方式来转换是作为本地应用程序还是远程应用程序。这样一个*双方式应用程序*在作为本地应用程序或远程应用程序时使用相同的 C 语言应用程序代码；但是作为本地应用程序和远程应用程序时，它将链接至不同的库。

将双方式应用程序与链接库访问配合使用

就链接库访问来说，当双方式应用程序在本地运行时，必须将它链接至本地链接库访问库。但是，当双方式应用程序远程运行时，它必须链接至链接库访问控制 API 存根库（例如，在 Windows 平台上为 solidctrlstub.lib），以便可以在没有链接时错误的情况下编译、链接和执行此双方式应用程序。

“控制 API 存根库”对于远程应用程序是必需的，因为链接库访问自己的控制 API（是在本地链接库访问库中提供的）不能与远程应用程序配合使用。例如，假定您有一个本地应用程序（包含控制 API 函数）已链接至标准 ODBC 库，而您希望远程运行此应用程序。通过链接至控制 API 存根库，就可以避免必须从代码中除去控制 API 函数调用。这样就很容易将链接库访问本地应用程序变成一个正常的远程客户机应用程序。

注:

控制 API 存根库中包含“不执行任何操作”的函数；如果您在远程应用程序中调用这些函数，它们在服务器上将不起作用。

双方式应用程序在下列情况下也很有用:

- 您可能希望在将本地应用程序与链接库访问库进行链接之前对此应用程序进行测试。
- 您可能希望所有用户/进程具有相同的应用程序逻辑，无论它们是本地还是远程用户/进程。

双方式应用程序

假定有两个用户要运行同一个应用程序。User1 在本地运行此应用程序（可以获得更高的性能）。User2 远程运行同一应用程序。

User1（本地用户）将编译并与服务器库（例如，solidac.a）进行链接，并且负责启动和停止服务器以及使用链接库访问的控制 API 来执行其他调度任务。User2（远程用户）将运行同一应用程序，但是要在 User1 启动服务器之后才能连接至服务器。因此，只有 User1 才能控制任务系统。

用于链接库访问的 solidDB 客户机 API 和驱动程序

下面是对可用于与链接库访问配合使用的 API 的简要描述。

注:

这些描述使用了第 4 页的『与链接库访问配合使用的应用程序类型』中所定义的“本地”应用程序和“远程”应用程序这两个术语。

solidDB SA API

SA API 是一个用于 solidDB 数据管理服务的低级别专用 C 语言 API。它包含在链接库访问库中（例如，在 Windows 平台上为 ssolidacxx.dll 或者在 UNIX 平台上为 solidac.a）。链接库访问库包含 SA-API 库，它使用 SA API 函数调用对本地应用程序提供支持。

SA API 库用于 solidDB 产品内部，通过此库可以访问 solidDB 数据库表中的数据。此库包含 90 个函数，它们提供了用于连接数据库和运行基于游标的操作的低级别机制。solidDB SA API 可以显著提高性能。例如，可以使用 SA API 来优化批处理插入操作的性能。

对于远程应用程序，链接库访问库还支持 SA API 函数调用。但是，您必须链接至单独的 SA API 库文件（例如，在 Windows 平台上为 solidimpsa.lib）。

有关 solidDB SA API 的详细信息，请参阅 *solidDB Programmer Guide*.

solidDB ODBC API

solidDB ODBC API 提供了一种符合标准的方法并通过 SQL 来访问本地或远程 solidDB 数据库中的数据。它提供了一些函数来控制数据库连接、执行 SQL 语句、检索结果集、落实事务以及其他数据管理功能。

ODBC API 是 solidDB 数据库的调用级接口（CLI），它符合 ANSI X3H2 SQL CLI，并且包含在链接库访问库中（例如，在 Windows 平台上为 ssolidacxx.dll 或者在 UNIX 平台上为 solidac.a）。

链接库访问支持 ODBC 3.51 标准。链接库访问库包含 solidDB ODBC 3.x，它对需要直接对服务器进行函数调用的本地应用程序提供支持。

对于本地应用程序，链接库访问库支持 ODBC 函数调用。对于远程应用程序（或者要远程运行的双方式应用程序），您必须链接 ODBC 驱动程序以获得相同功能。

如果应用程序是一个双方式应用程序（即，既可以在本地运行也可以远程运行的应用程序），并且它使用链接库访问的控制 API 和 ODBC，那么您将需要两个不同的可执行文件，一个在本地运行，另一个远程运行。当您链接应用程序以使它在本地运行时，将它链接至链接库访问库，此库同时支持 ODBC 函数和控制 API 库。当您链接应用程序以使它远程运行时，必须将它同时链接至 ODBC 驱动程序和控制 API 存根库（例如，在 Windows 平台上为 solidctrlstub.lib）。此存根库实际上没有为远程应用程序提供对服务器的任何控制权；它只是允许您编译和链接应用程序并且不产生有关“未解析符号”的错误。

注:

远程调用双方式应用程序中的 ODBC 函数时，客户机将通过网络来调用服务器。在本地调用已加速的应用程序中的 ODBC 函数时，ODBC 子例程库会直接将本地应用程序连接至服务器（无需通过网络）。

有关 ODBC API 的更多详细信息，请阅读 *solidDB Programmer Guide*.

solidDB JDBC API

只有远程应用程序才使用 JDBC API。作为 JDK 1.2 的核心 API，它定义 Java 类来表示数据库连接、SQL 语句、结果集和数据库元数据等。它允许您发出 SQL 语句并处理结果。JDBC 是用于 Java 数据库访问的主要 API。链接库访问支持 JDBC 1.x 和 2.x。有关更多详细信息，请阅读 *solidDB Programmer Guide*.

solidDB 控制 API（SSC API）

solidDB 控制 API（SSC API）是一个用 C 语言编写的线程安全接口，用来控制服务器在 solidDB 数据库产品中的行为。

控制 API 包含在链接库访问库中（例如，在 Windows 平台上为 ssolidacxx.dll 或者在 UNIX 平台上为 solidac.a）。链接库访问库通过使用控制 API 函数调用本地应用程序提供支持，还提供了一个可用于仅用作远程应用程序的独立库。

如果应用程序将远程运行并且包含控制 API 函数调用，那么您必须链接控制 API 存根库（例如，在 Windows 平台上为 solidctrlstub.lib）。实际上，此库不会向您提供服务器的远程应用程序控制；它仅在不出现具有链接库访问的 solidDB链接时错误的情况下，允许您将其作为远程应用程序编译和链接至您的应用程序。

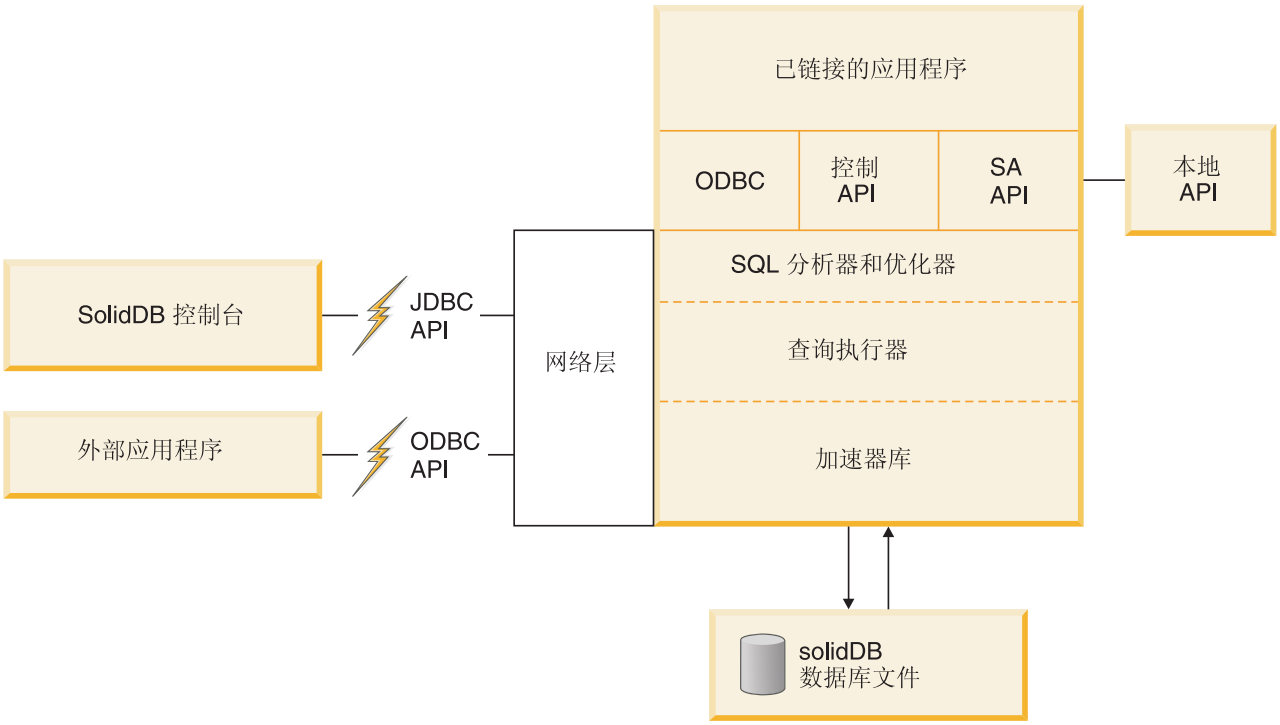


图 3. 具有链接库访问的 solidDB - API

2 创建和运行链接库访问应用程序

本章描述如何创建和运行链接库访问应用程序。它包含下列主题:

- 下载链接库访问库
- 将应用程序链接至链接库访问库
- 创建或使用现有数据库
- 启动和停止具有链接库访问的 solidDB

注:

本章提供了不同于 solidDB （不具有链接库访问）的特定于链接库访问的新增内容、补充内容和链接库访问用法差别。有关 solidDB SQL、solidDB 数据管理工具、常规 solidDB 管理和维护以及数据库错误代码的信息，请参阅《solidDB 管理员指南》。请阅读第 23 页的3 章，『对控制 API 的描述』和 *solidDB Programmer Guide*，以了解有关使用支持链接库访问的 API 来开发应用程序的详细信息。

访问链接库访问库

具有链接库访问的 solidDB 是包含在 solidDB Development Kit 中的一个库文件。

例如，如果您要在 HP-UX 系统中使用 solidDB，那么链接库访问库文件为 solidac.a。有关特定于平台的库列表，请参阅第 11 页的『为链接库访问链接应用程序』。

所有平台的链接库访问库都包含下列各项:

- solidDB 数据管理功能
- 本地用户应用程序的 SA API 头（sa.h）
- 本地用户应用程序的 solidDB 控制 API 接口头（sscapi.h）

有关将用户应用程序链接至链接库访问库的详细信息，请阅读第 11 页的『为链接库访问链接应用程序』。

用于远程应用程序的库

在此链接库访问指南中，“远程”应用程序是指未链接至服务器的任何应用程序，也就是未使用链接库访问库的任何应用程序。因此，虽然与数据库服务器在同一个节点上运行，但是未链接至此数据库服务器的应用程序也被认为是一个“远程”应用程序。远程应用程序通过诸如 TCP/IP 等网络通信协议与服务器进行通信。另一方面，“本地”应用程序都已链接至链接库访问库，并且无需通过使用网络协议就可以直接调用此库中的函数。

因为远程应用程序要使用网络通信协议，所以链接库访问并不会提高远程应用程序的性能。只有本地应用程序（即，直接链接至加速器库的应用程序）才能获得更高的性能。

但是，在某些情况下，远程应用程序可以通过使用 SA API 来获得更高的性能，SA API 允许通过低级别操作来读写数据库。

如果您要使用远程应用程序，那么可能需要将 solidDB SDK 中的下列库链接到您的应用程序中。

- 当应用程序具有“控制 API”函数调用，并且您希望远程运行此应用程序时，请链接至 solidDB 控制 API 存根库（在 Windows 平台上为 solidctrlstub.lib）。（请注意，如果您的应用程序是本地应用程序而不是远程应用程序（即，它直接链接至链接库访问库），那么您不需要 solidctrlstub.lib。）

有关控制 API 存根库（solidctrlstub.lib）的更多详细信息，请阅读第 5 页的『将双方式应用程序与链接库访问配合使用』。

- 如果您要运行远程 solidDB SA API 应用程序（没有链接库访问），那么请链接至 solidDB SA API（在 Windows 平台上为 solidimpsa.lib）。

如果您要仅使用 ODBC、SA API 或者 JDBC 作为远程应用程序（不使用控制 API 函数调用），那么不需要链接至 solidctrlstub.lib。

样本 C 应用程序

有关加速器控制 API 用法样本（在 C 编程语言中提供），请参阅安装目录下的 samples/aclib、samples/aclib_smartflow 和 samples/control_api。这些 C 语言样本反映了使用 ODBC API 函数来连接至 solidDB 服务器的已链接应用程序。

使用数据同步

如果您不熟悉 solidDB 数据同步，*solidDB Advanced Replication Guide* 中包含一些样本脚本可以帮助您。

在运行样本 C 语言应用程序 acsnet.c 之前（此应用程序位于 samples/aclib_smartflow 目录下），建议您通过至少执行下列其中一项操作来熟悉 solidDB 功能:

- 使用 solidDB（没有链接库访问）来运行 *solidDB Advanced Replication Guide* 中包含的 SQL 脚本。可在 samples/smartflow 下找到这些脚本。
- 在本地使用 solidDB 链接库访问来运行这些 SQL 脚本。作为一个先决条件，您需要按照本章中的指示信息来设置应用程序以启动服务器。有关详细信息，请阅读第 11 页的『为链接库访问链接应用程序』和第 15 页的『启动和停止 solidDB 链接库访问』。

注:

不能使用 SA API 来运行同步命令。

- 运行实现样本文件 aclibstandalone.c（它带有链接库访问库）时将模拟正常的服务器。此样本文件位于 samples/aclib 目录下。

在使用上述任何方法之后，就可以使用 solidDB SQL 编辑器（solsql）来运行 *solidDB Advanced Replication Guide* 的 *Getting Started with Data Synchronization* 一章中的所有步骤。

使用高级复制脚本来设置 ODBC 应用程序

可以构建一个与样本 C 应用程序 acsNet.c 相似的 ODBC 应用程序，用来执行在设置、配置和运行同步环境时所需要的所有语句。可以在 samples/aclib_smartflow 目录下找到 acsNet.c。

要设置样本数据库以将它与 ODBC 客户机应用程序配合使用，可以执行样本脚本 replica3.sql、replica4.sql、replica5.sql 和 replica6.sql，所有这些脚本都可以在 samples/smartflow/eval_setup 目录下找到。这些样本脚本中包含一些 SQL 语句，可用来将新数据写入副本以及控制同步消息的执行。可以通过 solidDB SQL 编辑器（solsql）来独立运行每个脚本。

或者，您可以将 SQL 语句嵌入 C/ODBC 应用程序中，编译此应用程序，然后将它直接链接至链接库访问库。链接至链接库访问库之后，使用样本脚本时就能获得链接库访问体系结构的性能优势。

samples/odbc 目录中的样本程序 embed.c 说明了如何通过使用链接库访问来设置具有 ODBC 客户机应用程序的数据库。您可以将样本脚本中的 SQL 命令（例如，replica3.sql 等）插入 embed.c 应用程序中。

为链接库访问链接应用程序

solidDB 链接库访问是一个必须链接至用户应用程序的库。只要应用程序正在运行，本地应用程序和远程应用程序就可以通过此库来请求 solidDB 数据管理服务。

注:

如果您要编写将使用 solidDB 控制 API 的远程用户应用程序，那么需要将远程应用程序链接至 solidDB 控制 API 存根库（例如，在 Windows 平台上为 solidctrlstub.lib）。如果您要远程使用 solidDB SA API（没有链接库访问），那么您需要链接至单独的 solidDB API 库（在 Windows 上为 solidimpsa.lib）。如果您要仅远程使用 ODBC、SA API 或者 JDBC（不使用控制 API），那么不需要链接至 solidDB 控制 API 存根库。

您一次仅将一个应用程序直接链接至链接库访问库。但是，一旦所链接的应用程序已启动且正在运行，并且服务器也已启动，那么任何网络客户机都可以使用服务器支持的任何协议（例如，TCP/IP、共享内存和命名管道，这取决于操作系统）来连接至此服务器。远程客户机不能使用直接函数调用。

将应用程序链接至具有链接库访问的 solidDB 时，请使用操作系统所需要的下列库之一。请参阅操作系统文档。

表 3. 链接库访问系统库

平台	具有链接库访问库的 solidDB
Windows	solidimpac.lib（这是一个导入库文件，使您可以访问实际的库文件，即 ssolidacxx.dll）
Solaris	solidac.a
HP-UX	solidac.a
Linux®	solidac.a
VxWorks	solidac.a

为链接库访问准备用户应用程序

为了允许应用程序使用具有链接库访问的 solidDB，请务必执行下列操作:

- 链接至链接库访问库而不是驱动程序库。

  如果您要使用远程应用程序，那么可能需要链接至其他库。有关详细信息，请阅读第 9 页的『用于远程应用程序的库』。

- 将连接字符串更改为本地或远程服务器名称。有关详细信息，请阅读第 14 页的『与具有链接库访问的 solidDB 建立本地或远程连接』。

- 如果需要，请添加对 SSCStartServer 和 SSCStopServer 的调用或者其他控制 API 调用。有关详细信息，请阅读第 25 页的『控制 API 引用』。

信号处理程序

信号处理程序用来向应用程序报告发生了意外事件（例如，除数为 0）。不能在用户应用程序中设置信号处理程序，这是因为它们将覆盖由链接库访问设置的信号处理程序。例如，如果用户应用程序为浮点异常设置一个信号处理程序，那么此设置就会覆盖由链接库访问设置的信号处理程序，从而使得服务器无法捕获诸如“除数为 0”等异常情况。

动态链接库

solidDB 同时提供了“静态的”和“动态的”链接库访问库。下面显示了一些主要平台的动态链接库文件的名称。（有关静态库的名称，请参阅第 11 页的『为链接库访问链接应用程序』。）

表 4. 库文件

平台	具有链接库访问库的 solidDB
Windows	ssolidacxx.dll
Solaris	ssolidacxx.so

表 4. 库文件 (续)

平台	具有链接库访问库的 solidDB
HP-UX	ssolidacxx.sl
Linux	ssolidacxx.so

静态和动态库文件中都包含 solidDB 服务器的完整副本（采用库格式）。当您使用静态库文件时（例如，lib/solidac.a），将程序直接链接至此文件，当然，您的代码和库代码都将写入最终获得的可执行文件。如果您链接至动态库文件，那么此库中的代码不会包含在输出文件中，但是输出文件中包含可执行程序。当运行您的程序时，将单独从此动态链接库中装入代码。

除了更改可执行文件的大小之外，链接至静态库文件与链接至动态库文件具有很小的区别。当然，如果您要在计算机上执行单个客户机和单个服务器，那么内存中的代码总量在任何时候都相似。它们的性能也差不多，只不过当您使用动态库时稍微会增加一点开销。

使用动态链接库文件的主要好处在于，如果您在同一台计算机中执行服务器的多个副本，那么可以节省内存。例如，如果您要在单台计算机上执行开发工作，并且您希望此计算机上同时具有高级复制主控服务器和高级复制副本服务器，或者您希望同时具有 HotStandby 主服务器和 HotStandby 辅助服务器，那么您可能更愿意使用动态库，以便内存中不会同时具有此链接库访问库的多个副本。

在 Microsoft Windows 上，solidDB 链接库访问包含其他文件 lib/solidimpac.lib。在 Microsoft Windows 上，如果您想使用动态链接库，那么请不要直接链接至动态链接库 ssolidacxx.dll 本身，而是链接至 solidimpac.lib（这是一个导入库）。这只会将少量代码链接至客户机可执行文件。当实际执行您的客户机程序时，Microsoft Windows 操作系统将自动装入 ssolidacxx.dll 文件，并且您的客户机将能够调用此 .dll 文件中的常用链接库访问函数。当您运行一个引用了此 .dll 文件的程序时，此 .dll 文件必须位于您的装入路径下。

注:

使用动态链接库文件并不意味着您可以将多个“本地”客户机链接至 solidDB 服务器。即使使用动态库方法，您仍然只能具有单个本地客户机；所有其他客户机必须是远程客户机，这意味着它们将通过使用 TCP 或者其他网络协议与 solidDB 服务器进行通信，而不是通过可供本地客户机使用的直接函数调用来与服务器进行通信。

MakeFile 示例

下面是在 Windows 和 Vxworks 中提供库名的一些示例。

Microsoft Windows MakeFile 示例

在下面的 Microsoft Windows makefile 示例中，使用了链接库访问的 solidDB 库名 solidimpac.lib。

```
# compiler
CC        = cl
# compiler flags
CFLAGS        = -I. -DSS_WINDOWS -DSS_WINNT
```

```
# linker flags and directives
SYSLIBS = libcmt.lib kernel32.lib advapi32.lib netapi32.lib wsock32.lib
user32.lib oldnames.lib gdi32.lib
LFLAGS  = ..\solidimpac.lib
OUTFILE  = -Fe

# MyApp building
all: myapp

myapp: myapp.c
  $(CC) $(CFLAGS)  $(OUTFILE)myapp myapp.c /link$(LFLAGS)
/NODEFAULTLIB:libc.lib
```

VxWorks MakeFile 示例

在下面的 VxWorks makefile 示例中，使用了链接库访问的 solidDB 库名 solidac.a。请注意，此示例使用的是反斜杠。如果您的 makefile 程序不支持路径名中使用反斜杠，那么请将反斜杠更改为正斜杠。

```
CC      = ccppc
CFLAGS  = -DSS_UNIX -DSS_VXW -I. -I..\..\include -I$(WIND_BASE)
\target\h \
          -DCPU=PPC603 -DMV2600
LFLAGS  = -nostartfiles -s -r ..\..\lib\solidac.a
OUTFILE = -o

# solidDB with AcceleratorLib samples building

all:    acsNet acsrv

acsNet: acsNet.c
        $(CC) $(CFLAGS) $(OUTFILE)acsNet acsNet.c $(LFLAGS)

acsrv:  acsrv.c
        $(CC) $(CFLAGS) $(OUTFILE)acsrv acsrv.c $(LFLAGS)
```

与具有链接库访问的 solidDB 建立本地或远程连接

一旦应用程序链接至链接库访问库，它就可以使用 ODBC API 或者 SA API 直接与本地服务器建立本地或远程连接。应用程序还可以与其他 solidDB 服务器（包括其他使用链接库访问的服务器）建立远程连接。

建立本地连接

当您建立本地连接时，客户机将通过直接对链接库访问库进行函数调用来调用服务器；而不是通过网络来调用服务器。

在 ODBC API 中，要与本地服务器（即，已链接至应用程序的服务器）建立连接，用户应用程序将使用文字串“localserver”来调用 SQLConnect 函数。请注意，对于本地服务器连接，也可以指定一个空源名“”。您也可以指定本地服务器名称，但是这将导致链接库访问使用“远程”连接（即，通过网络来建立连接，而不是通过直接对链接库访问库进行函数调用来建立连接）。

以下 ODBC API 代码示例使用用户名 dba 和密码 dba 直接连接至本地 solidDB 服务器:

```
rc = SQLConnect(hdbc, "localserver", (SWORD)SQL_NTS, "dba", 3, "dba", 3);
```

或者

```
rc = SQLConnect(hdbc, "", (SWORD)SQL_NTS, "dba", 3, "dba", 3);
```

在 SA API 中，要建立连接，用户应用程序将使用文字串“localserver”（而不是服务器名称）来调用 SaConnect 函数。请注意，对于本地服务器连接，也可以指定一个空源名“”。您也可以指定本地服务器名称，但是这将导致链接库访问使用“远程”连接（即，通过网络来建立连接，而不是通过直接对链接库访问库进行函数调用来建立连接）。

以下 SA API 示例代码使用用户名 dba 和密码 dba 直接连接至 solidDB 服务器:

```
SaConnectT* sc = SaConnect("localserver", "dba", "dba");
```

或者

```
SaConnectT* sc = SaConnect("", "dba", "dba");
```

建立远程连接

当您建立远程连接时，客户机将通过网络来调用服务器，而不是通过直接对链接库访问库进行函数调用来调用服务器。

在 ODBC API 中，要建立远程连接，用户应用程序将使用远程服务器的名称来调用 SQLConnect 函数。以下 ODBC API 代码示例使用用户名 dba 和密码 dba 连接至远程 solidDB 服务器。在此示例中，客户机和服务器使用的网络协议为“tcp”（TCP/IP）。服务器名为“remote_server1”，它侦听的端口为 1313。

```
rc = SQLConnect(hdbc, "tcp remote_server1 1313",
(SWORD)SQL_NTS, "dba", 3, "dba", 3);
```

在 SA API 中，要建立远程连接，用户应用程序将使用远程服务器的名称来调用 SaConnect 函数。在此示例中，客户机和服务器使用的网络协议为“tcp”（TCP/IP）。服务器名为“remote_server1”，它侦听的端口为 1313。

```
SaConnectT* sc = SaConnect("tcp remote_server1 1313", "dba", "dba");
```

启动和停止 solidDB 链接库访问

可以从下列 API 来启动、重新启动和停止 solidDB 服务器:

- 显式: 从本地（已链接的）用户应用程序通过调用控制 API 函数 SSCStartServer 来启动 solidDB，通过调用 SSCStopServer 将它关闭。

  当您启动一个新的、尚未数据库的 solidDB 服务器，您必须显式指定 solidDB 将使用 SSCStartServer() 函数并附带下列参数来创建一个新的数据库:

  -Uusername
  -Ppassword
  -Ccatalogname（缺省数据库目录名）

  有关详细信息，请阅读第 16 页的『使用控制 API 函数 SSCStartServer 显式启动』。

- 隐式: 当在本地首次使用 ODBC API 函数 SQLConnect 或者 SA API 函数 SaConnect 连接至 solidDB 时。在这种情况下，服务器将在通过调用 SQLDisconnect 或 SaDisconnect 函数使最后一个本地连接与 solidDB 断开连接时停止。

  当从应用程序隐式启动 solidDB 引擎/服务器时，它将检查 solidDB 目录中是否已经存在一个数据库。如果找到了数据库文件，那么 solidDB 将自动打开该数据库。如果

未找到数据库文件，那么 solidDB 将报告错误。（solidDB 在隐式启动期间不会创建新的数据库。要创建新的数据库，您必须使用显式启动函数（例如，SSCStartServer）并传递适当的参数。）

有关详细信息，请阅读第 18 页的『使用 ODBC API 函数调用 SQLConnect 来隐式启动』和第 20 页的『使用 SA API 函数调用 SaConnect 来隐式启动』。

注:

1.

在服务器启动时，在将控制权归还给应用程序之前，必要时可以执行恢复操作。因此，如果服务器已成功启动，那么它就可以开始处理应用程序请求。在应用程序进程的持续时间内，可以根据需要来启动或停止服务器。

2.

如果您要启动无盘服务器，那么必须使用控制 API 函数 SSCStartDisklessServer 来启动此服务器。

使用控制 API 函数 SSCStartServer 显式启动

要显式启动 solidDB，用户应用程序应调用以下控制 API 函数:

```
SSCStartServer (int argc, char* argv [ ],
SscServerT* h, SscStateT runflags)
```

其中的各个参数为:

表 5. SSCStartServer 参数

参数	描述
argc	命令行参数的数目。
argv	在执行函数调用期间使用的命令行参数组成的数组。argv[0] 参数是仅为用户应用程序的路径和文件名保留的，并且必须提供此参数。有关有效选项，请参阅下面的 SSCStartServer 选项。
h	每个服务器都有一个“句柄”（就是一个指向数据结构的指针）用于标识此服务器并指示有关此服务器的信息的存储位置。当使用其他控制 API 函数来引用此服务器时，此句柄是必需的。将在调用 SSCStartServer 函数时为您提供此服务器的句柄。要获得服务器的句柄，请创建一个类型为 pointer-to-server-handle 的变量（即，创建一个 SSCServerT *，它是一个指向句柄的指针 – 实质上是一个指向指针的指针），并在调用 SSCStartServer 时传递此变量。如果成功创建了服务器，那么 SSCStartServer 函数会将新服务器的句柄（指针）写入您为其传递了地址的变量中。
runflags	此参数的选项为 SSC_STATE_OPEN（允许建立远程连接）和 SSC_STATE_PREFETCH（需要时服务器将执行预取）。预取涉及到内存和/或磁盘高速缓存，磁盘高速缓存提供了对表内容的预读功能。请参阅下面的 runflags 参数条目: `runflags = SSC_STATE_OPEN | SSC_STATE_PREFETCH;`

当您首次启动服务器时，仅当您已经指定了数据库管理员的用户名和密码以及缺省数据库目录名时，solidDB 才会创建新的数据库。有关数据库目录的详细信息，请阅读《IBM solidDB SQL 指南》中的『将 solidDB SQL 用于数据管理』这一章中的『管理数据库对象』一节。

例如:

```
SscServerT h; char* argv[4];
argv[0] = "appname"; /* path and filename of the user app. */
argv[1] = "-UDBA"; /* user name */
argv[2] = "-PDBA"; /* user's password */
argv[3] = "-CDBA"; /* catalog name */
/* Start the server */
rc = SSCStartServer(argc, argv, &h, run_flags);
```

如果您启动的服务器上没有现存的数据库并且未指定数据库目录名，那么 solidDB 将返回一个错误并指出找不到数据库。

注:

如果您已经有一个现存的数据库，那么不需要指定用户名和密码或者目录名。

缺省情况下，将在当前工作目录所在的 solidDB 工作目录中，将数据库作为一个文件来创建（可以使用缺省名称 solid.db，也可以使用您在 solid.ini 文件中指定的名称）。一个只包含系统表和视图的空数据库将占用大约 850 KB 磁盘空间。创建数据库所花的时间取决于您使用的硬件平台。

创建数据库之后，solidDB 将开始侦听网络，以了解是否有远程客户机连接请求。

SSCStartServer argv 参数选项

下面是 argv 参数的命令行选项。请注意，所有选项都区分大小写。

表 6. SSCStartServer argv 选项

选项	描述
-c *dir*	更改工作目录。
-m	监视用户的消息和 SQL 语句。
-n *name*	设置服务器名称。
-U *username*	指定所创建数据库的管理员的用户名。用户名不区分大小写，其长度至少应为两个字符。用户名最长可为 80 个字符。用户名必须以字母或下划线开头。可使用小写字母 a 到 z、大写字母 A 到 Z、下划线字符“_”以及数字 0 到 9。 **注意:** **必须记住您的用户名，以便能够连接至 solidDB。没有缺省用户名；您在创建数据库时输入的用户名是唯一可用于连接至此数据库的用户名。**
-P *password*	指定所创建数据库的管理员的密码。密码不区分大小写，并且其长度至少应为三个字符。密码可以将字母、下划线或数字作为开头。可使用小写字母 a 到 z、大写字母 A 到 Z、下划线字符“_”以及数字 0 到 9。

表 6. SSCStartServer argv 选项 (续)

选项	描述
-C *catalogname*	指定数据库缺省目录的名称；如果您是首次启动服务器，那么此选项是必需的。有关数据库目录的详细信息，请阅读《IBM solidDB SQL 指南》中的『将 solidDB SQL 用于数据管理』这一章中的『管理数据库对象』一节。
-xautoconvert	将数据库格式转换为当前版本并启动服务器进程。
-xforcerecovery	强制进行前滚恢复。
-xignoreerrors	忽略索引错误。
-xtestblocks	测试数据库块。
-xtestindex	测试数据库索引。

启动 SSCStartServer

使用服务器名称、目录名以及管理员的用户名和密码来启动 SSCStartServer:

```
SscStateT runflags = SSC_STATE_OPEN; SscServerT h; char* argv[5];
argv[0] = "appname"; /* path and filename of the user app. */
argv[1] = "-nsolid1"; argv[2] = "-UDBA" argv[3] = "-PDBA";
argv[4] = "-CDBA"; /* Start the server */ rc =
SSCStartServer(argc, argv, &h, run_flags);
```

注:

如果您已经有一个现存的数据库，那么不需要指定用户名和密码或者目录名。

使用 SSCStopServer 来关闭

如果服务器是通过 SSCStartServer 启动的，那么必须通过在嵌入式应用程序中执行以下函数调用来关闭此服务器:

SSCStopServer()

例如:

```
/* Stop the server * /
SSCStopServer (h, TRUE);
```

使用 ODBC API 函数调用 SQLConnect 来隐式启动

首次调用 SQLConnect 函数时，将隐式启动服务器。当用户应用程序调用 SQLDisconnect 函数并且这是最后一个打开的本地连接时，服务器将隐式关闭。请注意，无论当前是否具有远程连接，服务器都将关闭。

注:

当您首次启动服务器时，必须使用 SSCStartServer() 函数并且指定缺省数据库目录以及管理员的用户名和密码来创建 solidDB 数据库。有关描述和示例，请阅读第 16 页的『使用控制 API 函数 SSCStartServer 显式启动』。

下面是一个使用 SQLConnect 和 SQLDisconnect 进行隐式启动和关闭的示例:

```
/* Connection #1 */
rc = SQLConnect (hdbc1, "", SQL_NTS, "dba", SQL_NTS, "dba",
SQL_NTS); //Server Started Here
... odbc calls

/* Disconnect #1 */
SQLDisconnect (hdbc1); //Server Shut Down Here

/* Connection #2 */
rc = SQLConnect (hdbc2, "", SQL_NTS, "dba", SQL_NTS, "dba",
SQL_NTS); //Server Started Here
... odbc calls

/* Disconnect #2 * /
SQLDisconnect (hdbc2); //Server Shut Down Here
```

或者

```
/* Connection #1*/
rc = SQLConnect (hdbc1, "", SQL_NTS, "dba",
SQL_NTS, "dba", SQL_NTS); // Server Started Here

/* Connection #2*/
rc = SQLConnect (hdbc2, "", SQL_NTS, "dba", SQL_NTS, "dba", SQL_NTS);

... odbc calls

/* Disconnect #1 */
SQLDisconnect (hdbc1);
/* Disconnect #2 * /
SQLDisconnect (hdbc2); // Server Shut Down Here
```

注:

如果服务器是通过调用 SSCStartServer 函数启动的，那么 SQLDisconnect 不会隐式关闭。必须通过 SSCStopServer、ADMIN COMMAND 'shutdown' 或其他显式关闭方法来显式关闭服务器。

```
SscStateT runflags = SSC_STATE_OPEN;
SscServerT server;
SQLHDBC hdbc;
SQLHENV henv;
SQLHSTMT hstmt;

/* Start the server */
SSCStartServer (argc, argv, &server, runflags); // Server Started Here

/* Alloc environment */
rc = SQLAllocEnv (&henv);

/* Connect to the database */
rc = SQLAllocConnect (henv, &hdbc);
rc = SQLConnect (hdbc, "", SQL_NTS, "dba", SQL_NTS, "dba", SQL_NTS);

/* Delete all the rows from table foo */
rc = SQLAllocStmt (hdbc, &hstmt):
rc = SQLExecDirect (hsmt, (SQLCHAR *) "DELETE FROM FOO", SQL_NTS);

/* Commit */
rc = SQLTransact (henv, hdbc, SQL_COMMIT);
rc = SQLFreeStmt (hstmt, SQL_DROP);

/* Disconnect */
SQLDisconnect (hdbc);
```

```
SQLFreeConnect (hdbc);

/* Free the environment */
SQLFreeEnv(henv);

/* Stop the server */
SSCStopServer (server, TRUE);  // Server Shut Down Here
```

使用 SA API 函数调用 SaConnect 来隐式启动

首次调用 SaConnect 函数时，将隐式启动服务器。当用户应用程序调用 SaDisconnect 函数并且没有后续连接时，服务器将隐式关闭。

注:

当您首次启动服务器时，必须使用 SSCStartServer() 函数并且指定缺省数据库目录以及用户名和密码来创建 solidDB 数据库。有关描述和示例，请阅读第 16 页的『使用控制 API 函数 SSCStartServer 显式启动』。

下面是一个使用 SaConnect 和 SaDisconnect 进行隐式启动和关闭的示例:

```
/* Open Connection */
SaConnect(...);

Server Started Here
... sa calls

/* Close Connection */
SaDisconnect(...);

Server Shut Down Here
```

注:

如果服务器是通过调用 SSCStartServer 函数启动的，那么只能通过调用 SSCStopServer 函数来关闭此服务器。

关闭 solidDB 链接库访问

只要您具有 SYS_ADMIN_ROLE 特权，就可以从 solidDB 客户机接口（甚至从另一个远程 solidDB 连接）来关闭 solidDB 服务器。

您可以从诸如 solidDB SQL 编辑器（solsql）的应用程序或者从“solidDB 远程控制”工具来按程序执行关闭[1]。

为此，请执行下列步骤:

1. 为了防止与 solidDB 建立新的连接，请通过输入以下命令来关闭数据库:

   ADMIN COMMAND 'close'

2. 通过输入以下命令来退出所有 solidDB 用户:

1. 当使用“solidDB 远程控制”工具来完成步骤 1 到 3 时，仅输入命令名且不带引号（例如，close）。

ADMIN COMMAND 'throwout all'

3.

通过输入以下命令来停止 solidDB:

ADMIN COMMAND 'shutdown'

所有关闭机制都将启动同一例程，此例程会将所有已缓冲的数据写入数据库文件中，释放高速缓存存储器，最终将终止服务器程序。关闭服务器可能要花一点时间，这是因为服务器必须将所有已缓冲的数据从主内存写入磁盘中。

注:

可以使用显式方法（例如，SSCStopServer）来关闭先前使用隐式方法（例如，SQLConnect）启动的服务器。反向操作则不行；例如，不能使用 SQLDisconnect 来停止先前使用 SSCStartServer 启动的服务器。

隐式启动配置参数

仅当建立了本地连接时，solidDB 才会隐式启动服务器。在 solid.ini 配置文件的 [Accelerator] 节中，ImplicitStart 参数在缺省情况下被设置为 Yes。当您使用任何 ODBC 连接必需的 SQLConnect 函数时，此缺省设置将自动启动服务器。SaConnect 函数的行为相似。首次调用此函数时，将隐式启动服务器。

3 对控制 API 的描述

控制 API（又称为 SSC API）是一系列函数，它们提供了简单而有效的方法来控制 solidDB 的任务系统。

检索任务信息

要检索所有活动任务的列表，请使用 SSCGetActiveTaskClass 函数。要检索所有暂挂任务的列表，请使用 SSCGetSuspendedTaskClass 函数。要获取任务类的优先级，请使用 SSCGetTaskClassPrio 函数。

特殊事件的通知函数

链接库访问对优先级任务做出了精确调整。可以使用 SSCSetNotifier() 函数来确定一旦发生了特殊事件，solidDB 就会调用用户定义的指定函数。此函数将检测的特殊事件为:

- solidDB 服务器关闭
- 从索引到存储器树的 Bonsai 合并
- Bonsai 合并最大时间间隔
- 备份请求或执行检查点操作请求
- 空闲服务器状态
- 从主服务器中接收到 netcopy 请求,这是一个要求将主数据库的网络副本发送至辅助服务器的请求。
- netcopy 请求的完成，当使用通过网络复制（netcopy）接收到的新数据库来启动服务器时就会完成 netcopy 请求。

获取 solidDB 状态和服务器信息

可以使用 SSCGetStatusNum 函数来查看 solidDB 数据库服务器的当前状态信息。将显示以下信息:

- 未从 Bonsai 树合并到存储器树的行数

如果 solidDB 服务器正在运行，那么 SSCGetServerHandle 函数将返回此服务器的句柄。

还可以使用 SSCIsRunning 函数来验证服务器是否正在运行，使用 SSCIsThisLocalServer 函数来验证应用程序是已链接至本地链接库访问服务器库（例如，在 Windows 平台上为 ssolidacxx.dll）还是“伪”服务器库（例如，在 Windows 平台上为 solidctrlstub.lib），这两个库用来测试要使用控制 API 的远程应用程序。

控制 API 函数总结

下面是对各个控制 API 函数以及在『控制 API 函数参考』一节中对此函数进行描述的位置的简要总结。

表 7. 控制 API 函数总结

函数	描述	要了解更多详细信息，请参阅:
SSCSetCipher	设置由应用程序提供的加密库。	请参阅 第 30 页的『SSCSetCipher』。
SSCStartServer	启动 solidDB 链接库访问服务器。	请参阅 第 40 页的『SSCStartServer』。
SSCStartDisklessServer	启动 solidDB 链接库访问无盘服务器。	请参阅 第 37 页的『SSCStartDisklessServer』。
SSCSetState	设置 solidDB 服务器的状态（例如，SSC_STATE_OPEN 指示是否允许建立后续连接）。将状态设置为 ~SSC_STATE_OPEN 将同时阻塞本地连接和远程连接。	请参阅 第 36 页的『SSCSetState』。
SSCRegisterThread	为服务器注册一个链接库访问应用程序线程。需要在用户应用程序中的每个线程中注册之后，才能调用任何加速器 API 函数。	请参阅 第 30 页的『SSCRegisterThread』。
SSCUnregisterThread	为服务器注销一个链接库访问应用程序线程。需要在已注册的每个线程中解除注册之后才能终止服务器。	请参阅 第 45 页的『SSCUnregisterThread』。
SSCStopServer	停止 solidDB 服务器。	请参阅 第 43 页的『SSCStopServer』。
SSCSetNotifier	指定用户定义的函数，当发生所指定的事件（例如，合并、备份和关闭等）时，solidDB 就会调用此函数。	请参阅第 33 页的『SSCSetNotifier』。
SSCIsRunning	如果服务器正在运行，那么将返回一个非零值。	请参阅 第 29 页的『SSCIsRunning』。
SSCIsThisLocalServer	指示应用程序是链接至具有链接库访问的 solidDB 服务器还是“伪”库（solidctrlstub），以使用链接库访问的控制 API 来测试 solidDB 远程应用程序。	请参阅 第 29 页的『SSCIsThisLocalServer』。
SSCGetServerHandle	如果服务器正在运行，那么将返回 solidDB 服务器句柄。	请参阅 第 27 页的『SSCGetServerHandle』。
SSCGetStatusNum	获取 solidDB 状态信息。	请参阅 第 28 页的『SSCGetStatusNum』。

控制 API 和等价的 ADMIN COMMAND

控制 API 函数具有等价的 solidDB SQL 扩展 ADMIN COMMAND。在远程站点和本地站点中都可以通过 solidDB 工具（例如，solidDB 远程控制（solcon）和 solidDB SQL 编辑器（solsql））来执行这些命令。

有关控制 API 等价 ADMIN COMMAND 的详细信息，请参阅第 59 页的『链接库访问参数』。

控制 API 引用

下列各页按字母顺序描述了每个控制 API 函数。每个描述都包含用途、摘要、参数、返回值和注释。

函数摘要

此函数的声明摘要为:

```
ReturnType SSC_CALL function(modifier parameter[,...]);
```

ReturnType 会发生变化，但通常是一个指示调用成功或失败的值。在本节的后面部分更详细地描述了返回值。

为了提供可移植性，SSC_CALL 是必需的。SSC_CALL 指定函数的调用约定。在 sscapi.h 文件中为每个平台都适当地定义了调用约定。

各个参数是用斜体显示的，下面对它们进行了描述。

参数描述

在每个函数描述中，都是按表格式来描述各个参数的。表中包含的是参数的一般用法类型（下面进行了描述）和特定函数中参数变量的用途。

参数用法类型

下表显示了控制 API 参数可能采用的用法类型。请注意，如果一个参数用作指针，那么它将包含另一个用法类别，以指定参数变量在调用之后的所有权。

表 8. 控制 API 参数用法类型

用法类型	含义
in	指示此参数是输入。
output	指示此参数是输出。
in out	指示此参数是输入/输出。
use	仅适用于 pointer 参数。这意味着只有在进行函数调用期间才使用此参数。在进行函数调用之后，调用者可以使用此参数来执行它希望的操作。这是最常见的参数传递类型。

表 8. 控制 API 参数用法类型 (续)

用法类型	含义
take	仅适用于 pointer 参数。这意味着函数采用了此参数值。在进行函数调用之后，调用者不能引用此参数。当不再需要此参数时，此函数或者在此函数中创建的对象负责释放此参数。

hold	仅适用于 pointer 参数。这意味着即使在进行函数调用之后，函数仍将保留参数值。在进行函数调用之后，调用者可以继续引用参数值并且负责释放参数。
	
	警告:
	
	因为用户和服务器共享此参数，所以在服务器结束使用此参数之前请不要将它释放。通常，对于一个被挂起的对象，可以在释放挂起它的对象之后再释放此对象。例如:
	`conn = SaConnect("", "dba", "dba");`
	`/* Connection is held until cursor is freed */`
	`scur = SaCursorCreate(conn, "mytable");`
	`...`
	`SaCursorFree(scur);`
	`/* After we free the cursor, it is safe to free */`
	`/* the connection (or, as in this case, call a */`
	`/* server function that frees the connection). */`
	`SaDisconnect(conn);`

返回值

每个函数描述都指示此函数是否返回了值以及所返回值的类型。

SscTaskSetT

当函数返回一个类型为 SscTaskSetT 的值时，此定义用作一个位掩码。SScTaskSetT 是在 sscapi.h 中定义的，它可以为下列值:

```
SSC_TASK_NONE
SSC_TASK_CHECKPOINT
SSC_TASK_BACKUP
SSC_TASK_MERGE
SSC_TASK_LOCALUSERS
SSC_TASK_REMOTEUSERS
SSC_TASK_SYNC_HISTCLEAN
SSC_TASK_SYNC_MESSAGE
SSC_TASK_HOTSTANDBY
SSC_TASK_HOTSTANDBY_CATCHUP
SSC_TASK_ALL（上述所有任务）
```

请注意，HotStandby“netcopy”和 HotStandby“copy”操作是由“SSC_TASK_BACKUP”任务执行的；没有单独的“SSC_TASK_NETCOPY”任务。

控制 API 错误代码和消息

控制 API 函数可能会返回下列错误代码和消息:

表 9. 控制 API 函数的错误代码和消息

错误代码/消息	描述
SSC_SUCCESS	操作成功。
SSC_ERROR	一般错误。
SSC_ABORT	操作异常终止。
SSC_FINISHED	如果已经执行了所有任务，那么 SSCAdvanceTasks 将返回此消息。
SSC_CONT	如果还需要执行其他任务，那么 SSCAdvanceTasks 将返回此消息。
SSC_CONNECTIONS_EXIST	存在开放式连接。
SSC_UNFINISHED_TASKS	存在未完成的任务。
SSC_INFO_SERVER_RUNNING	服务器已经在运行。
SSC_INVALID_HANDLE	给定了无效的本地服务器句柄。此服务器与通过 SSCStartServer 启动的服务器不匹配。
SSC_INVALID_LICENSE	找不到许可证或者找到了无效许可证文件。
SSC_NODATABASEFILE	找不到数据库文件。
SSC_SERVER_NOTRUNNING	服务器未运行。
SSC_SERVER_INNETCOPYMODE	服务器采用 netcopy 方式（仅适用于高可用性/HotStandby）。

这些常量（SSC_SUCCESS 等）是在 sscapi.h 文件中定义的。

SSCGetServerHandle

如果服务器正在运行，那么 SSCGetServerHandle 将返回 solidDB 服务器句柄。

摘要

```
SscServerT SSC_CALL SSCGetServerHandle(void)
```

注释

此函数没有相应的 solidDB SQL 扩展 ADMIN COMMAND。

返回值

- 如果服务器未运行，那么将返回 NULL。
- 如果服务器正在运行，那么将返回服务器句柄。

SSCGetStatusNum

SSCGetStatusNum 将获取 solidDB 的状态信息。

摘要

```
SscRetT SSC_CALL SSCGetStatusNum(SscServerT h, SscStatusT stat,
      long * num)
```

SSCGetStatusNum 函数接受下列参数:

表 10. SSCGetStatusNum 参数

参数	用法类型	描述
h	in, use	服务器的句柄。
stat	in	指定用于检索的状态标识:
num	out	如果成功调用了此函数，那么当它返回时，此参数的值将设置为未合并的写入数，或者设置为服务器线程数，这取决于所请求的信息。

注释

此函数没有相应的 solidDB SQL 扩展 ADMIN COMMAND。

如果您调用 SSCGetStatusNum 并为 stat 参数传递一个不识别的值，那么此函数将返回 SSC_SUCCESS。

返回值

- SSC_SUCCESS - 操作成功。如果您为 stat 参数传递了一个无效值，那么也会返回此值。
- SSC_ERROR - 操作失败。
- SSC_SERVER_INNETCOPYMODE - 服务器采用 netcopy 方式（仅适用于 HotStandby）。
-

SSC_SERVER_NOTRUNNING – 服务器未运行。

SSCIsRunning

如果服务器正在运行，那么 SSCIsRunning 将返回非零值。

摘要

```
int SSC_CALL SSCIsRunning(SscServerT h)
```

SSCIsRunning 函数接受下列参数:

表 11. SSCIsRunning 参数

参数	用法类型	描述
h	in, use	服务器的句柄

返回值

- 0 – 服务器未运行。
- 非零值 – 服务器正在运行。

注释

此函数没有相应的 solidDB SQL 扩展 ADMIN COMMAND。

SSCIsThisLocalServer

SSCIsThisLocalServer 指示应用程序是已链接至 solidDB 服务器还是“伪”库（solidctrlstub）。solidctrlstub 库使开发者可以使用控制 API 来测试 solidDB 远程应用程序，而不需要与链接库访问库进行链接和修改源代码。

摘要

```
int SSC_CALL SSCIsThisLocalServer(void)
```

返回值

- 0 – 应用程序并未链接至 solidDB 服务器。
- 1 – 应用程序已链接至 solidDB 服务器。

注释

此函数没有相应的 solidDB SQL 扩展 ADMIN COMMAND。

SSCRegisterThread

SSCRegisterThread 用于为服务器注册 solidDB 应用程序线程。每个使用控制 API、ODBC API 或 SA API 的线程都必须注册。必须在线程调用 SSCRegisterThread 函数之后，才能使用任何其他链接库访问 API 函数。

如果应用程序只有一个线程（主线程），也就是说，应用程序本身未创建任何线程，那么不需要注册。

在线程终止之前，它必须通过调用 SSCUnregisterThread 函数将它自己注销。

摘要

```
SscRetT SSC_CALL SSCRegisterThread(SscServerT h)
```

SCCRegisterThread 函数接受下列参数:

表 12. SCCRegisterThread 参数

参数	用法类型	描述
h	In, Use	服务器的句柄

返回值

- SSC_SUCCESS
- SSC_INVALID_HANDLE

注释

此函数没有相应的 solidDB SQL 扩展 ADMIN COMMAND。

另请参阅

SSCUnregisterThread

SSCSetCipher

SSCSetCipher 函数设置由应用程序提供的密码以及加密/解密函数。当 SSCStartServer 中使用了相关的数据库加密命令行参数时，就会自动使用所提供的密码。

摘要

```
void SSC_CALL SSCSetCipher(
    void* cipher,
    char* (SSC_CALL *encrypt)(void *cipher, int page_no, char *page,
        int n, size_t pagesize),
    int (SSC_CALL *decrypt)(void *cipher, int page_no, char *page,
        int n, size_t pagesize));
```

表 13. SSCSetCipher 参数

参数	用法类型	描述
cipher	in	指向特定于应用程序的密码对象（例如，加密密码）的指针。solidDB 服务器不使用此指针或相反解释此指针。它只是将它传递给由应用程序提供的加密/解密函数。
encrypt	in	指向由应用程序提供的加密函数的指针。当服务器必须对数据库文件或日志文件页进行加密时，就会从服务器中调用此函数。此函数的参数为: • cipher – 指向由应用程序提供的密码对象的指针。 • page_no – 由服务器提供的页号。加密算法可以安全地忽略它，或者将它用作加密密钥的一部分。 • n – 要加密的页数。 • pagesize – 要加密的页的大小。 • page – 指向要加密的数据缓冲区的指针。数据缓冲区的大小为: *n*pagesize* 函数必须返回指向要写入文件的已加密数据缓冲区（大小为 *n*pagesize*）的指针。 服务器不会释放指向已加密数据缓冲区的指针。 加密函数可以采用任何方式来覆盖或处理作为“page”参数传递给函数的数据缓冲区。例如，加密函数可以“在原地”对数据进行加密并返回“页”指针。

表 13. SSCSetCipher 参数 (续)

参数	用法类型	描述
decrypt	in	指向由应用程序提供的解密函数的指针。当服务器已读取已加密的数据库或日志文件的一部分并且需要对它进行解密时，就会从服务器中调用此函数。此函数的参数为: • cipher – 指向由应用程序提供的密码对象的指针。 • page_no – 由服务器提供的页号。解密算法可以安全地忽略它，或者将它用作解密密钥的一部分。 • n – 要解密的页数。 • pagesize – 要解密的页的大小。 • page – 指向要解密的数据缓冲区的指针。数据缓冲区的大小为: *n*pagesize*

返回值

SSCSetCipher 函数不会返回任何值。应当在使用 SSCStartServer 函数来启动链接库访问服务器之前调用此函数。

注释

如果解密函数已成功地对页进行解密，那么它应当返回一个非零值；如果由于某个原因造成解密失败，那么它应当返回 0。在后面这种情况下，服务器将紧急关闭，因为它无法继续运行。此函数应当将已加密的数据返回在由“page”参数指定的同一缓冲区中。

使用链接库访问加密 API

以下代码说明了 AcceleratorLib 加密 API 的用法。采用的加密方法很普通，也就是常用的 XOR 迷惑。

```
char* SS_CALLBACK encrypt(void *cipher, int page_no, char *page, int np,
size_t pagesize)
{
        size_t n = np*pagesize;
        int *key = cipher;
        size_t i;

        for (i=0; i<n; i++) {
            page[i] ^= (i+*key);
        }
        return page;
```

```
}


bool SS_CALLBACK decrypt(void *cipher, int page_no, char *page, int np,
size_t pagesize)
{
        size_t n = np*pagesize;
        int *key = cipher;
        size_t i;

        for (i=0; i<n; i++) {
            page[i] ^= (i+*key);
        }

        return TRUE;
}
...

int main(int argc, char** argv)
{
    int key = 17;
...
        SSCSetCipher(&key, encrypt, decrypt);
        sscret = SSCStartServer(argsc, args, &h, SSC_STATE_OPEN);
        ....
        SSCStopServer(h, FALSE);
...

}
```

SSCSetNotifier

SSCSetNotifier 设置链接库访问服务器在启动和停止时将调用的回调函数。此函数没有相应的 ADMIN COMMAND。

摘要

```
SscRetT SSC_CALL SSCSetNotifier(SscServerT h, SscNotFunT what,
    notify_fun handler, void* userdata
)
```

SSCSetNotifier 函数接受下列参数:

表 14. SSCSetNotifier 函数参数

参数	用法类型	描述
h	in	服务器的句柄。

表 14. SSCSetNotifier 函数参数 (续)

参数	用法类型	描述
what	in	指定要通知的事件。选项为: • SSC_NOTIFY_EMERGENCY_EXIT 在使用 SSCStartServer() 激活了服务器之后，如果此服务器崩溃，就会调用此函数。必须在发出 SSCStartServer() 之前发出通知函数调用 SSCSetNotifier() • SSC_NOTIFY_SHUTDOWN 服务器关闭时调用此函数。 • SSC_NOTIFY_SHUTDOWN_REQUEST 当服务器接收到关闭请求时调用此函数，如果用户定义的函数接受请求，那么服务器可能会关闭。可以通过从被通知的函数返回 SSC_ABORT 或者为此请求返回 SSC_CONT 来拒绝关闭服务器。 • SSC_NOTIFY_ROWSTOMERGE 当 bonsai 索引树中有需要合并到存储服务器的数据时，就要调用此函数。 • SSC_NOTIFY_MERGE_REQUEST 当超过了 solid.ini 配置文件中的 MergeInterval 参数设置并且必须启动合并时，就要调用此函数。 • SSC_NOTIFY_BACKUP_REQUEST 当请求了备份时调用此函数。可以通过从被通知的函数返回 SSC_ABORT 来拒绝备份。 • SSC_NOTIFY_CHECKPOINT_REQUEST 当请求了执行检查点操作时调用此函数。可以通过从被通知的函数返回 SSC_ABORT 来拒绝执行检查点操作。 • SSC_NOTIFY_IDLE 当服务器切换到空闲状态时调用此函数。 • SSC_NOTIFY_NETCOPY_REQUEST 此回调函数仅适用于 HotStandby 组件。当从主服务器中接收到 netcopy 请求时调用此函数，netcopy 请求是要求将主数据库的网络副本发送至辅助服务器的请求。有关 netcopy 命令的详细信息，请参阅《solidDB 高可用性用户指南》。 • SSC_NOTIFY_NETCOPY_FINISHED 此回调函数仅适用于 HotStandby 组件。当完成了 netcopy 请求时调用此函数。完成了 netcopy 请求时，将使用通过网络复制（netcopy）接收到的新数据库来启动服务器，并且调用 SSC_NOTIFY_FINISHED 以便让应用程序知道服务器再次可用。
notify_fun_handler	in, hold	要调用的用户函数。
userdata	in, hold	要传递给 notify 函数的用户数据。 请务必阅读第 25 页的『参数描述』中有关释放用法类型为 *hold* 的参数的警告。

返回值

- SSC_SUCCESS – 接受了来自服务器的请求。

  仅适用于 HotStandby:

  如果 SSC_NOTIFY_NETCOPY_FINISHED 返回 SSC_SUCCESS，那么所有其他应用程序连接都将终止并且将服务器设置为“netcopy 侦听方式”。在此方式下，服务器将接受来自主服务器的连接，并且辅助服务器唯一可以执行的操作就是接收来自 HotStandby netcopy 命令的数据。有关“netcopy 侦听方式”的更多详细信息，请阅读《solidDB 高可用性用户指南》。（请注意，“netcopy 侦听方式”在以前又称为“备份侦听方式”。）

- SSC_ABORT – 拒绝了来自服务器的请求。

  仅适用于 HotStandby:

  如果 SSC_NOTIFY_NETCOPY_REQUEST 返回了 SSC_ABORT，那么表示 netcopy 未启动并且会将错误代码（SRV_ERR_OPERATIONREFUSED）返回到主服务器中。

- SSC_INNETCOPYMODE – 服务器采用 netcopy 方式（仅适用于 HotStandby）。

  SSC_SERVER_NOTRUNNING – 服务器未运行。

注释

此函数没有相应的 solidDB SQL 扩展 ADMIN COMMAND。

释放一个用法类型为 *hold* 的参数时应谨慎。请阅读有关 hold 第 25 页的『参数描述』的警告。

用户定义的 notifier 函数不应调用任何 SA、SSC 或 ODBC 函数。

创建用户定义的 notifier 函数时，必须符合以下原型:

```
int SSC_CALL mynotifyfun(SscServerT h, SscNotFunT what ,void* userdata);
```

一旦您使用了 SSC_CALL 来显式定义用户函数的约定，那么请使用 SSCSetNotifier 函数来注册此函数，以便在发生所指定的事件时调用此函数；例如:

```
SscRetT SSCSetNotifier(h, SSC_NOTIFY_IDLE, mynotifyfun, NULL);
```

示例

关闭时调用函数

假定用户将创建 user_own_shutdownrequest 函数，每次请求关闭时就会调用此函数:

```
int SSC_CALL user_own_shutdownrequest(SscServerT h, SscNotFunT what, void
    *userdata);
{
  if (shutdown not needed) {
```

```
    return SSC_ABORT;
  }
  return SSC_CONT; /*Proceed with shutdown*/
}
```

然后，可以按如下所示调用 SSCSetNotifier 函数，以指定在服务器关闭之前调用 user_own_shutdownrequest。

```
SSCSetNotifier(handle, SSC_NOTIFY_SHUTDOWN, user_own_shutdownrequest, NULL);
```

注:

如果 user_own_shutdownrequest 函数返回 SSC_ABORT，那么不允许关闭服务器；如果该函数返回 SSC_CONT，那么可以关闭服务器。

SSCSetState

SSCSetState 设置链接库访问服务器的状态。这使您可以控制服务器是否接受后续连接以及服务器是否使用预取。

如果服务器设置为“打开”，那么服务器将接受连接。如果服务器设置为“关闭”，那么它将不接受任何后续连接（无论是本地连接还是远程连接）；但是，允许已经建立的所有连接都继续运行。

打开预取就会告知服务器要进行“预读”，以访存可能马上就要引用的数据。进行预取将需要更多内存或磁盘高速缓存空间。打开预取时，性能通常会更高。关闭预取时，需要的内存更少。如果您的查询需要对服务器进行大量顺序扫描，那么打开预取就非常有用。例如，如果您使用报告或聚集函数来获取整个数据库（或者它的很大一部分）的值，那么预取可能对您有帮助。如果您的所有查询只涉及到一条或多条记录，那么预取通常就没太大用处。由于预取将耗用内存，因此在可用内存很少的系统中，预取实际上可能会降低性能。

下列准则可以帮助您决定何时使用预取。

如果您有大量的可用内存（或者磁盘高速缓存空间），并且您的查询需要执行大量顺序扫描，那么就可以使用预取。

如果您的可用内存很少，并且您的查询通常一次只读取一条不相关的记录，那么请不要使用预取。

摘要

```
SscRetT SSC_CALL SSCSetState(SscServerT h,SscStateT runflags)
```

SSCSetState 函数接受下列参数:

表 15. SSCSetState 函数参数

参数	用法类型	描述
h	in, use	服务器的句柄。

表 15. SSCSetState 函数参数 (续)

参数	用法类型	描述
runflags	in	选项可以由 SSC_STATE_OPEN 标志（它意味着允许建立新的远程连接）和 SSC_STATE_PREFETCH 标志（它意味着用户允许服务器在必要时执行预取）组合而成。下面是可能存在的组合的一个示例: • 服务器设置为“打开”: state = state | SSC_STATE_OPEN; • 服务器设置为“关闭”: state = state & ~SSC_STATE_OPEN; • 打开预取: state = state | SSC_STATE_PREFETCH; • 关闭预取: state = state & ~SSC_STATE_PREFETCH;

返回值

- SSC_SUCCESS – 操作成功。
- SSC_ERROR – 操作失败。
- SSC_SERVER_INNETCOPYMODE – 服务器采用 netcopy 方式（仅适用于 HotStandby）。
- SSC_SERVER_NOTRUNNING – 服务器未运行。

注释

此函数具有相应的 solidDB SQL 扩展 ADMIN COMMAND。此命令为:

ADMIN COMMAND ’close’;

SSCStartDisklessServer

SSCStartDisklessServer 将启动一个使用链接库访问的无盘服务器。

摘要

```
SscRetT SSC_CALL SSCStartDisklessServer (int argc, char* argv[ ],
    SscServerT * h, SscStateT runflags, char* lic_string, char* ini_string);
```

SSCStartDisklessServer 函数接受下列参数:

表 16. SSCStartDisklessServer 参数

参数	用法类型	描述
argc	in	命令行参数的数目。
argv	in, use	在执行函数调用期间使用的命令行参数组成的数组。参数 argv[0] 是仅为用户应用程序的路径和文件名保留的，它必须存在。有关有效参数的列表，请参阅下面所列示的 SSCStartDisklessServer 参数选项。
h	out	将返回已启动的服务器的句柄。当通过其他控制 API 函数来引用服务器时需要此句柄。
runflags	in	此参数的唯一选项是: SSC_STATE_OPEN – 允许建立远程连接。 runflags = SSC_STATE_OPEN
lic_string	in	指定包含 solidDB 许可证文件的字符串。
ini_string	in	指定包含 solidDB 配置文件的字符串。

SSCStartDisklessServer 参数选项

下面是 *argv* 参数的命令行选项。

表 17. argv 参数的命令行选项

选项	描述
-h	显示帮助。
-n *name*	设置服务器名称。
-U *username*	指定数据的用户名。用户名不区分大小写。用户名长度至少应为两个字符。用户名最长可为 80 个字符。用户名必须以字母或下划线开头。可使用小写字母 a 到 z、大写字母 A 到 Z、下划线字符“_”以及数字 0 到 9。 **注:** 必须记住您的用户名，以便能够连接至 solidDB。没有缺省用户名；您在创建数据库时输入的用户名是唯一可用于连接至此数据库的用户名。
-P *password*	指定数据的给定密码。密码不区分大小写。密码长度至少应为三个字符。密码可以将字母、下划线或数字作为开头。可使用小写字母 a 到 z、大写字母 A 到 Z、下划线字符“_”以及数字 0 到 9。

表 17. argv 参数的命令行选项 (续)

选项	描述
-C *catalogname*	指定数据的目录名；如果您是首次启动服务器，那么此选项是必需的。指定此参数时，请务必使用大写字母 C。有关目录的详细信息，请阅读《solidDB 管理员指南》的『将 solidDB SQL 用于数据管理』这一章中的『管理数据库对象』一节。
-x ignoreerrors	忽略索引错误。

返回值

- SSC_SUCCESS – 服务器已启动。
- SSC_ERROR – 服务器未能启动。
- SSC_SERVER_INNETCOPYMODE – 服务器采用 netcopy 方式（仅适用于 HotStandby）。
- SSC_INFO_SERVER_RUNNING – 服务器已经在运行。
- SSC_INVALID_HANDLE – 给定了无效的本地服务器句柄。
- SSC_INVALID_LICENSE – 找不到许可证或者找到了无效许可证文件。

注释

缺省情况下，状态设置为 SSC_STATE_OPEN。

此函数没有相应的 solidDB SQL 扩展 ADMIN COMMAND。

示例

SSCStartDisklessServer

```
SscStateT runflags = SSC_STATE_OPEN;
SscServerT h;
char* argv[4]; /* pointers to four parameter strings */
int argc = 4;
char* lic = get_lic(); /* get the license */
char* ini = get_ini(); /* get the solid.ini */
SscRetT rc;
argv[0] = "appname"; /* path and filename of the user app. */
argv[1] = "-Udba"; /* user name */
argv[2] = "-Pdba"; /* user's password */
```

```
argv[3] = "-Cdba"; /* catalog name */
/* Start the diskless server */
rc = SSCStartDisklessServer(argc, argv, &h, runflags, lic, ini);
```

注:

在此示例中，get_ini() 和 get_lic() 是用户必须编写的函数。每个函数必须返回一个包含 solid.ini 文件文本或者 solid.lic 许可证文件的字符串。

如果您未指定目录名，那么 solidDB 将返回错误。

另请参阅

SSCStopServer

另请参阅第 47 页的4 章，『使用无盘功能』。

SSCStartServer

SSCStartServer 将启动链接库访问。在多线程环境中，服务器与客户机运行于不同线程中。在应用程序的持续时间内，应用程序可以根据需要来启动或停止服务器子例程。

请注意，第三个参数是一个“out”参数。如果已成功启动服务器，那么 SSCStartServer 例程会将此参数设置为指向此服务器的句柄。

注:

如果您要启动无盘服务器，那么必须使用控制 API 函数 SSCStartDisklessServer 来启动此服务器。请阅读第 37 页的『SSCStartDisklessServer』。

摘要

```
SscRetT SSC_CALL SSCStartServer(int argc, char* argv[], SscServerT* h
    SscStateT runflags)
```

SSCStartServer 函数接受下列参数:

表 18. SSCStartServer 参数

参数	用法类型	描述
argc	in	命令行参数的数目。
argv	in, use	命令行参数的数组。有关有效参数的列表，请参阅『SSCStartServer』。
h	out	将返回已启动的服务器的句柄。当通过其他控制 API 函数来引用服务器时需要此句柄。

表 18. SSCStartServer 参数 (续)

参数	用法类型	描述
runflags	in	选项可以是下列其中一项或者两项: • SSC_STATE_OPEN – 允许建立远程连接。 • SSC_STATE_PREFETCH – 服务器将在必要时执行预取。 例如: `runflags = SSC_STATE_OPEN & SSC_STATE_PREFETCH`

返回值

- SSC_SUCCESS – 服务器已启动。
- SSC_ERROR – 服务器未能启动。
- SSC_ABORT
- SSC_BROKENNETCOPY – 由于未完成 netcopy 而导致数据库被破坏。
- SSC_FINISHED
- SSC_CONT
- SSC_CONNECTIONS_EXIST
- SSC_UNFINISHED_TASKS
- SSC_INVALID_HANDLE – 给定了无效的本地服务器句柄。
- SSC_INVALID_LICENSE – 找不到许可证或者找到了无效许可证文件。

- SSC_NODATABASEFILE – 找不到数据库文件。
- SSC_SERVER_NOTRUNNING
- SSC_INFO_SERVER_RUNNING – 服务器已经在运行。
- SSC_SERVER_INNETCOPYMODE – 服务器采用 netcopy 方式（仅适用于 HotStandby）。
- SSC_DBOPENFAIL – 打开数据库失败。
- SSC_DBCONNFAIL – 连接至数据库失败。
- SSC_DBTESTFAIL – 测试数据库失败。
- SSC_DBFIXFAIL – 修复数据库失败。
- SSC_MUSTCONVERT – 必须转换数据库。
- SSC_DBEXIST – 数据库已存在。
- SSC_DBNOTCREATED – 未创建数据库。
- SSC_DBCREATEFAIL – 创建数据库失败。
- SSC_COMINITFAIL – 通信初始化失败。
- SSC_COMLISTENFAIL – 通信侦听失败。
- SSC_SERVICEFAIL – 服务操作失败。
- SSC_ILLARGUMENT – 命令行参数非法。

- SSC_CHDIRFAIL – 更改目录失败。
- SSC_INFILEOPENFAIL – 打开输入文件失败。
- SSC_OUTFILEOPENFAIL – 打开输出文件失败。
- SSC_SRVCONNFAIL – 服务器连接失败。
- SSC_INITERROR – 操作初始化失败。
- SSC_CORRUPTED_DBFILE – 断言或其他致命错误。
- SSC_CORRUPTED_LOGFILE – 断言或其他致命错误。

注释

缺省情况下，状态设置为 SSC_STATE_OPEN。

此函数没有相应的 solidDB SQL 扩展 ADMIN COMMAND。

当您启动新的 solidDB 服务器时，必须显式指定 solidDB 会通过将 SSCStartServer() 函数与 -U *username* -P *password* -C *catalogname*（缺省数据库目录名）参数配合使用来创建新的数据库。有关详细信息，请阅读第 16 页的『使用控制 API 函数 SSCStartServer 显式启动』。

如果您要重新启动数据库服务器（即，一个数据库已经存在于目录中），那么 SSCStartServer 将使用现有数据库。

SSCStartServer 函数可能会衍生多个线程来运行服务器任务。服务器任务包括处理本地和远程客户机请求以及运行各种后台任务（例如，执行检查点操作和合并等等）。

另请参阅

SSCStopServer

SSCStopServer

SSCStopServer 将停止链接库访问服务器。

请注意，可以使用显式方法（例如，SSCStopServer）来关闭先前使用隐式方法（例如，SQLConnect）启动的服务器。反向操作则不行；例如，不能使用 SQLDisconnect 来停止先前使用 SSCStartServer 启动的服务器。

每当应用程序运行时，并不限制应用程序只能启动和停止服务器一次。在服务器已停止之后，应用程序可以使用 SSCStartServer 来重新启动此服务器。

摘要

```
SscRetT SSC_CALL SSCStopServer(SscServerT h, bool force)
```

SSCStopServer 函数接受下列参数:

表 19. SSCStopServer 参数

参数	用法类型	描述
h	in, use	服务器的句柄
force	in	选项为: • TRUE – 在所有情况下都停止服务器。 • FALSE – 如果没有开放式连接就停止服务器。否则，就无法停止服务器。

返回值

- SSC_SUCCESS – 服务器已停止。
- SSC_CONNECTIONS_EXIT – 存在开放式连接。
- SSC_UNFINISHED_TASKS – 正在执行的任务。
- SSC_ABORT
- SSC_ERROR

注释

远程用户可以使用 ADMIN COMMAND 'shutdown' 来停止 solidDB。有关详细信息，请参阅第 59 页的『链接库访问参数』。

如果与数据库或现有用户建立了开放式连接，那么 FALSE 选项将不允许关闭。此选项等价于 solidDB SQL 扩展 ADMIN COMMAND 'shutdown'。

带有 &~SSC_STATE_OPEN 选项的 SSCSetState() 将阻止与 solidDB 建立新连接。

另请参阅

SSCStartServer

SSCSetState

SSCUnregisterThread

SSCUnregisterThread 将注销服务器的 solidDB 应用程序线程。必须由已注册的每个线程自己通过 SSCRegisterThread 函数来调用 SSCUnregisterThread 函数。将在线程终止之前调用此函数。

摘要

```
SscRetT SSC_CALL SSCUnregisterThread(SscServerT h)
```

SCCUnregisterThread 函数接受下列参数:

表 20. SCCUnregisterThread 参数

参数	用法类型	描述
h	in, use	服务器的句柄

返回值

- SSC_SUCCESS
- SSC_INVALID_HANDLE

注释

SSC_CALL 需要显式定义用户函数的调用约定。它是在每个平台的相应 sscapi.h 文件中定义的。

此函数没有相应的 solidDB SQL 扩展 ADMIN COMMAND。

另请参阅

SSCRegisterThread

4 使用无盘功能

solidDB 链接库访问允许您创建一个在没有任何磁盘存储空间的情况下运行的数据库引擎。这在没有硬盘的嵌入式系统（例如，网络路由器或交换机中的线卡）中很有用。

可以按以下两种主要方式来运行无盘服务器：1) 独立运行；2) 作为高级复制系统中的副本服务器运行。在每种情况下，您都将通过使用链接库访问函数调用 SSCStartDisklessServer() 来启动服务器。

无盘服务器独立运行

如果独立运行无盘服务器，那么它在启动时当然就无法读取数据，关闭时也无法写入数据。这就意味着此服务器每次启动时都没有先前的任何数据。

而且，由于此服务器无法将数据写入磁盘，因此如果它因发生电源故障之类的错误而异常关闭，那么此服务器中的所有数据都将丢失并且无法恢复。可以通过使用 solidDB HotStandby 组件创建一个“热备用”机器（它包含数据的副本）以降低丢失数据的风险。有关此热备用功能的更多信息，请参阅《solidDB 高可用性用户指南》。

作为高级复制系统的一部分的无盘服务器

无盘服务器可以是高级复制系统中的一个副本服务器。在这种情况下，副本服务器可以将数据发送至主控服务器，并且可以从该主控服务器中下载数据。因此，尽管副本服务器没有自己的磁盘存储器或者其他永久存储器，它也可以使它的某些数据或所有数据持久保存在高级复制系统中。

无盘服务器的配置参数

本节描述用于实现和维护无盘服务器的参数设置。

无盘服务器中使用的参数

配置文件的下列各节包含一些参数，这些参数对于无盘服务器具有特定设置。

[索引文件]节

下面是会影响索引文件的配置参数。

FileSpec_[1...N] 参数

FileSpec 参数描述数据库文件的名称和最大大小。要定义主内存引擎的最大大小（按字节计），FileSpec 参数接受下列自变量：

- 数据库文件名 – 由于无盘服务器不会创建物理数据库文件，因此未使用此参数；但是，必须为此自变量提供一个哑元值。
-

最大文件大小 – 此设置是必需的。您需要指定足够大的文件大小（按字节计），以存储无盘服务器中的所有数据。请注意，最大文件大小必须小于高速缓存大小，高速缓存大小是使用 CacheSize 参数来设置的。

FileSpec 参数的缺省值为 solidr.db，5000000 个字节。例如:

```
FileSpec_1=SOLIDR.db 5000000
```

注:

如果您指定多个文件，那么最大文件大小设置必须是所有 FileSpec 参数设置的总和。

当然，最大大小受到可用物理内存的限制，这是因为无盘机器没有磁盘用作虚拟内存的交换空间。请注意，在某些平台上，可用于应用程序的物理内存量可能小于机器中的物理内存量。例如，在某些版本的 32 位 Linux 系统中，可用于应用程序的内存量只有理论地址空间（4GB）的一半或者四分之一，这是因为 Linux 会将地址的一个或两个最高有效位保留给它自己的内存管理器使用。

如果内存中的数据量超过了最大文件大小，那么会显示错误消息 11003:

```
文件写入失败，超过了配置大小
```

CacheSize

CacheSize 参数定义服务器为缓冲区高速缓存分配的主内存量（按字节计）。例如:

```
CacheSize=10000000
```

此值的设置取决于无盘服务器的下列条件:

- 对基于磁盘的表，高速缓存大小（按字节计）至少应该比使用 FileSpec 参数设置的最大文件大小（即，数据量）大 20%，原因是此数据保存在缓冲区高速缓存中。20% 的开销是一个估计值，可能会根据数据库的使用情况而变化。例如:

  ```
  [IndexFile]
  FileSpec_1=solid.db 10MB
  CacheSize=12MB
  ```

- 即使未使用基于磁盘的表（数据库是使用内存表创建的），也需要使用高速缓存来存放系统表。在这种情况下，高速缓存大小最少应为 1 到 2 MB。系统表占用的空间取决于数据库对象的数目和复杂程度以及是否使用了高级复制。

- 高速缓存大小必须小于可用于运行无盘服务器的物理内存量。

  可以按如下所示估计无盘服务器使用的内存总量。（请注意，所有这些内存的总量必须小于可用的物理内存总量，这就意味着高速缓存大小必须远远小于可用于服务器的物理内存总量: ）

  ```
  CacheSize + 5MB
  + (100K * 用户数 * 每个用户具有的活动语句数)
  + 内存表空间
  + (要发送至辅助服务器的 HSB 操作数) [1][2]
  ```

[1] 此等式中的条件仅适用于 HotStandby 用户。HSB 主服务器需要一些内存用来存储要发送至辅助服务器的 HotStandby 操作。在主服务器与无盘辅助服务器之间临时发生网络故障期间，主服务器可以继续接受来自应用程序的事务。当服务器之间恢复了网络连接之后，就会将主服务器中的更新发送至辅助服务器。（HotStandby 使用事务日志来存储这些操作。当然，无盘服务器无法将事务日志写入磁盘，因此必须将信息存储在内存中。）此内存与高速缓存是分开的。

[2] 对于此等式中的条件，最大限制目前是 1 MB 或者 512 项操作，以这两者中的较小者为准。与基于磁盘的服务器不同，并不允许事务日志一直增大到用尽可用空间为止。

所需内存的准确数量还取决于其他因素（例如，对服务器执行的查询的性质）。由于操作系统等将占用一些物理内存，因此可用于服务器的内存量通常小于物理内存总量。

[Com] 节

下面是一些将影响主控服务器与无盘副本服务器（如果您使用无盘服务器作为高级复制副本服务器）之间通信的配置参数。

Listen 参数 [Com]

这是无盘服务器在开始侦听网络时使用的协议和名称。其缺省值取决于操作系统。请参阅《solidDB 管理员指南》中的『管理网络连接』。

不适用于无盘引擎的配置参数

对于无盘服务器，下列配置文件参数（按节分组）已被禁用或者不起作用。这些参数将影响不适用于无盘引擎的行为。

表 21. 不适用于无盘引擎的配置参数

参数	描述
[General] 节	
CheckpointInterval	此参数已被禁用，这是因为检查点不应用于无盘服务器。
[IndexFile] 节	
ReadAhead	因为不会从数据库文件中物理读取，因此该参数不起作用
PreFlushPercent	因为不会物理写入数据库文件，因此该参数不起作用
[Logging] 节	
LogEnabled	此参数已被禁用，这是因为始终将对无盘服务器禁用事务记录。 **注:** 无盘方式仅支持事务回滚。通常在某些故障中断了已部分完成的事务时使用事务回滚。无盘方式不支持进行前滚恢复。

5 将 solidDB 链接库访问与 Java 语言配合使用

注:

本章假定您已经熟悉了前面各章中的内容。如果因为您只对 Java/JDBC 感兴趣（对 C/ODBC 不感兴趣）而直接跳到本章学习，那么您将错过很多对于理解本章很有用的内容。

solidDB JDBC 加速器（SJA）概述

与 C/ODBC 程序相似，Java/JDBC 程序可以使用 solidDB 链接库访问来获取更高的性能以及对服务器的更强控制能力。SJA 使 Java 应用程序能够启动本地 solidDB 服务器，此本地服务器将从一个称为“ssolidacxx”的动态库中装入 Java 虚拟机上下文。然后，Java 应用程序将能够连接至 solidDB 服务器并使用 solidDB DBMS 通过标准 JDBC API 提供的服务。

由于客户机应用程序直接链接至服务器库，调用服务器函数时不存在网络（RPC）调用开销，因此客户机应用程序将获得更高的性能。由于应用程序可以调用 solidDB 服务器控制（SSC）库中的函数（方法）来执行诸如为某些类型的任务指定优先级的操作，因此应用程序将具有更高的控制能力。例如，应用程序可以为它自己指定高优先级，而为远程客户机应用程序指定低优先级。

仅当服务器与客户机互相链接起来之后，才能使用 solidDB JDBC 加速器（SJA）；因此，如果 Java 应用程序和 solidDB 服务器要在不同的主机中运行，那么就不能使用 SJA。

当然，只有“本地”客户机（即，已链接至链接库访问库的客户机）不需要通过网络建立连接，从而获得链接库访问提供的更高性能。其他客户机程序也可以使用服务器，但是它们必须通过网络才能与服务器建立连接，因此将它们视作“远程”程序，即使它们与 solidDB 服务器在同一台计算机上运行也是如此。您只能具有一个“本地”客户机；其余都是“远程”客户机。远程程序可以是 C 程序和 Java 程序的混合体。

编写本地客户机时使用的语言并不会限制可以用来编写远程客户机的语言。例如，如果您使用 JDBC 加速器，那么远程客户机程序可以使用 C 和/或 Java。

加速器的工作方式

与 C 语言程序一样，要使用链接库访问的 Java/JDBC 程序必须链接至 solidDB 链接库访问库（ssolidacxx）。此库包含整个 solidDB 服务器，只不过它是采用可调用库形式，而不是采用独立的可执行程序形式。与 Java/JDBC 配合使用的 ssolidacxx 与先前章节中说明的 ssolidacxx 相同；没有单独用于 Java 和 C 客户机的版本。通过链接至此库，客户机程序就减少了通过网络进行 RPC（远程过程调用）所产生的开销。

当您将链接库访问与 Java/JDBC 配合使用时，请将下列各项链接到单个可执行进程中:

- solidDB 链接库访问库,

- Java 语言客户机程序，以及
- JVM。

可执行进程中的各“层”按从上到下的顺序排列依次为:

- 本地 Java/JDBC 客户机应用程序
- JVM（Java 虚拟机）
- solidDB 加速器库（ssolidacxx）

客户机中的 Java 命令由 JVM 执行。如果命令是一个 JDBC 函数调用，那么 JVM 将调用 ssolidacxx 中的相应函数。函数调用是“直接”进行的，无需通过网络（通过 RPC）来调用。使用 JNI（Java 本机接口）来执行调用。请注意，您不需要知道这些低级别的详细信息。您不需要自己编写任何 JNI 代码；您只需要调用远程客户机程序将调用的 JDBC 函数。

从 Java 加速器中访问 solidDB 数据库与通过 RPC 访问 solidDB 数据库几乎完全相同，它们之间只有一点不同：为了访问数据库服务，使用 Java 加速器的应用程序必须首先启动 solidDB 链接库访问服务器。这是由一个称为 SolidServerControl（SSC）的专用 API 完成的。可通过 SSC API 调用来启动和停止 solidDB DBMS。实际的数据库连接是由正常的 JDBC API 完成的。SolidServerControl API 和 solidDB 的 JDBC 驱动程序都可以在一个名为 SolidDriver2.0.jar 的 .jar 文件中找到。

在本地 solidDB 服务器启动之后，将把它从一个称为 ssolidacxx 的动态库装入到 Java 虚拟机上下文中。然后，Java 应用程序将能够连接至 solidDB 服务器并使用 solidDB DBMS 通过标准 JDBC API 提供的服务。

每个使用 solidDB Java 加速器的本地客户机程序都遵循相同的基本三步骤模式:

1. 使用 SolidServerControl 启动加速器服务器
2. 使用正常的 JDBC API 来访问数据库
3. 完成数据库处理之后，再次使用 SolidServerControl 来停止加速器服务器

用于访问 solidDB 加速器服务器的 SolidServerControl 类已被嵌入在 solid.ssc 程序包的 solidDB JDBC 驱动程序文件中。solidDB JDBC 驱动程序 JAR 文件（SolidDriver2.0.jar）中包含下列程序包:

-

solid.jdbc.* solidDB JDBC 驱动程序类

- solid.ssc.* solidDB 服务器控制类（专用接口）

solidDB 服务器控制（solid.ssc）程序包中的类为:

- SolidServerControl（用于使用 Java 来启动和停止 solidDB 服务器）
- SolidServerControlInitializationError（用于报告错误）

有关 SolidServerControl（SSC）类接口的详细信息，请参阅第 55 页的『solidDB 服务器控制（SSC）API』。

要从 Java 应用程序来启动 solidDB 服务器，必须在应用程序的开头将 SolidServerControl 类实例化并使用正确的参数来调用 startServer 方法（在下面提供了示例）。在启动服务器之后，您就可以与服务器建立 JDBC 连接了。

系统需求

您需要具备下列各项才能使用 solidDB Java 加速器:

- solidDB 链接库访问库本身。这是一个名为 ssolidacxx 的文件。其文件名扩展名随平台的不同而改变；下面列示了一些常见名称和平台:
  - Microsoft Windows: ssolidacxx.dll 和导入库 solidimpac.lib
  - Solaris 和 Linux: ssolidacxx.so
  - HP-UX: ssolidacxx.sl
- 使用 solidDB 服务器和链接库访问时应具备的有效许可证文件
- solidDB JDBC2 驱动程序文件（SolidDriver2.0.jar）
- 用于您所使用的平台的 solidDB 通信库（这些通信库是在您安装 solidDB Development Kit 时正常安装的）。
- Java Development Kit (JDK) 1.4.2 或更高版本

基本用法

安装

如果您已经安装了 Java Development Kit，那么不需要执行任何其他安装。 安装了 solidDB 之后，它将包含在使用 solidDB Java 加速器时所需要的库。

注: 可能需要将 PATH 和 CLASSPATH 环境变量设置为适当的值，以便您能访问 Java 编译器。

编译和运行程序

为了使服务器成功启动，您必须至少具有使用 solidDB 和链接库访问所需要的有效许可证。

ssolidacxx 动态链接库必须位于系统搜索路径中。按如下所示继续执行操作:

1. 设置路径（以 Microsoft Windows 命令提示符为例）

   set PATH=<path to your ssolidacxx DLL>;%PATH%

   请确保您提供的路径中也有一个包含 solidDB 通信库的目录。
2. 将 PATH 环境变量设置为包含 JDK 的 HOTSPOT 运行时环境（SJA 仅在热点 JRE 中进行了测试）。例如,

   set PATH=<your JDK directory>\jre\bin\hotspot;%PATH%
3. 将本章末尾包含的示例文件保存到一个名为 SJASample.java 的文件中，并使用以下命令来编译此文件:

   javac -classpath <IBM solidDB JDBC driver directory>/SolidDriver2.0.jar;. \
   SJASample.java
4. 使用与以下相似的命令行来运行应用程序:

   java -Djava.library.path=<path to ssolidacxx DLL> \ -classpath <IBM soliDB JDBC driver directory>/SolidDriver2.0.jar;. \ <your application name>

   例如，在 Microsoft Windows 上，如果您已将服务器安装在 C:\soliddb 下并且您希望运行 SJASample 程序，那么命令行看起来应为如下所示:

   java -Djava.library.path=C:\soliddb\bin -classpath C:\soliddb\jdbc\SolidDriver2.0.jar;. SJASample

   （在 Microsoft Windows 上，ssolidacxx.dll 动态库位于 solidDB 根安装目录的 bin 子目录中。）

   与在示例类 SJASample 中一样，必须使用 SolidServerControl 的 startServer 方法为 solidDB 服务器至少传递下列参数:

   ```
   -c<directory containing solidDB license file>
   -U<username>
   -P<password>
   -C<catalog>
   ```

   请注意，同时使用了大写字母“C”和小写字母“c”，它们代表不同的含义。

   假定当前工作目录中具有所有的必需文件（ssolidacxx 库、通信库、JDBC 驱动程序和 solid.lic），那么可使用与以下命令行相似的命令行来启动 SJASample:

   java -Djava.library.path=. -classpath SolidDriver2.0.jar;. <your application>

   如果一切都按预期运行，那么现在应该有一个 solidDB 加速器服务器已启动并且正在运行。

建立 JDBC 连接

solidDB Java 加速器支持本地数据库连接以及基于 RPC 的连接。

为了建立本地 JDBC 连接（而不是基于 RPC 的 JDBC 连接），需要指定您要在端口 0 使用“localserver”的 JDBC 驱动程序。因此，如果您要通过使用诸如 JDBC 类 DriverManager 来建立数据库连接，那么请使用以下语句来进行连接（在非常靠后的示例代码 SJASample 中也提供了此语句）：

```
DriverManager.getConnection("jdbc:solid://localserver:0", myLogin, myPwd);
```

正如您所知道的，DriverManager 使用 URL *"jdbc:solid://localserver:0"* 与本地服务器建立连接。如果为 getConnection 子例程提供了另一个 URL，那么驱动程序可能将尝试使用 RPC 建立连接。

因此，当您建立 Java 加速器连接时，请记住 URL –

```
jdbc:solid://localserver:0
```

注:

如果您要使用多个将访问 Java 应用程序中的 solidDB 链接库访问服务器的线程（`java.lang.Thread` 对象），那么在使用某个线程来启动与 JDBC 相关的任何活动之前，必须向 solidDB 链接库访问服务器单独注册此线程。线程的注册是通过在此线程的上下文中调用 SolidServerControl API 的 registerThread 方法来完成的。必须为使用 solidDB 的 JDBC 驱动程序的每个用户线程（主线程除外）显式完成线程注册。

用户还必须显式注销已向 solidDB 链接库访问服务器注册的每个线程。要注销线程，请调用 SolidServerControl API 的 unregisterThread 函数。

局限性

- solidDB ’admin commands’ 在 Java 加速器上下文中不起作用。
- 如果在虚拟机上下文外部（例如，在本机方法调用中）发生了故障，那么 Java 的行为将不一致。如果在 solidDB 服务器本机代码中应当对某个对象作出断言（甚至使它崩溃），那么 Java 将退出（当它注意到发生了意外异常时）或者彻底挂起。在后面这种情况下，您可能必须手动终止悬挂的 Java 进程。
- 要将内存消耗降低到最小，建议用户显式丢弃所有的分配语句；即，必须通过调用 close() 方法显式释放所有的分配 JDBC 语句对象。
- 如果从多个 Java 线程访问相同的语句对象，那么服务器可能会崩溃。 所以您必须对于每个需要使用 JDBC 的线程打开不同的 JDBC 连接（和语句）。

solidDB 服务器控制（SSC）API

下面是 SolidServerControl 类的完整公共接口。有关使用此类中的某些方法的程序示例，请参阅 samples/accelerator_java/SJASample.java 文件。

```
/**
* See solidDB  Linked Library Access User Guide
* for the following constants
*/
public final static int SSC_SUCCESS = 0;
public final static int SSC_ERROR = 1;
public final static int SSC_ABORT = 2;
public final static int SSC_FINISHED = 3;
public final static int SSC_CONT = 4;
public final static int SSC_CONNECTIONS_EXIST = 5;
public final static int SSC_UNFINISHED_TASKS = 6;
public final static int SSC_INVALID_HANDLE = 7;
```

```
        public final static int SSC_INVALID_LICENSE = 8;
        public final static int SSC_NODATABASEFILE = 9;
        public final static int SSC_SERVER_NOTRUNNING = 10;
        public final static int SSC_INFO_SERVER_RUNNING = 11;
        public final static int SSC_SERVER_INNETCOPYMODE = 12;

        public final static int SSC_STATE_OPEN     = (1 << 0);
        public final static int SSC_STATE_PREFETCH = (1 << 1);


/**
 * Initiates a SolidServerControl class. Output is not directed to any
 * PrintStream.
 *
 * @return        SolidServerControl instance
 */
public static  SolidServerControl instance()
     throws SolidServerInitializationError;


/**
 * Initiates a SolidServerControl class. Output is being directed
 * to a PrintStream object given in parameter 'os'.
 *
 * @param os      the PrintStream for output
 * @return        SolidServerControl instance
 *
 */
public static SolidServerControl instance( PrintStream os )
     throws SolidServerInitializationError;

/**
 * setOutStream method sets the output to the given PrintStream
 *
 * @param os      the PrintStream for output
 */
public void setOutStream( PrintStream os );


/**
 * getOutStream returns the stream used for output in class
 * SolidServerControl
 *
 * @return        returns the outputstream of this object
 */
public PrintStream getOutStream();


/**
 * startServer starts the solidSB Linked Library Access server
 *
 * @param argv       parameter vector for the accelerator server
 *                   ( be sure to give the working directory containing
 *                   solidDB license file (f.ex. -c\tmp) first, in front
       *                 of other parameters. ) See solidDB Linked Library
 *                   Access User Guide for details of parameters that can
 *                   be passed to the Linked Library Access server.
 *
 * @param runflags   Options for this parameter are SSC_STATE_OPEN
 *                   (remote connections are allowed) and
 *                   SSC_STATE_PREFETCH (server will do a "prefetch"
 *                   if needed). Prefetch refers to the memory
 *                   and/or disk cache that provides read-ahead
 *                   capability for table content. Following is
 *                   a runflags parameter entry:
 *                    runflags |= SSC_STATE_OPEN & SSC_STATE_PREFETCH
 *
```

```
 * @return   the return value from the server :
 *              SSC_SUCCESS
 *              SSC_ERROR
 *              SSC_INVALID_LICENSE - No license or invalid license file found.
 *              SSC_NODATABASEFILE  - No database file found.
 */
public long startServer( String[] argv, long runflags );


/**
* stopServer stops the solidDB  Linked Library Access server
 *
 * @param runflags   Runflags for stopping solidDB  Linked Library Access server.
       *                 See solidDB Linked Library Access User Guide for more
 *                  details.
 *
 * @return   the return value from the server
 *              SSC_SUCCESS if server is stopped.
 *              SSC_CONNECTIONS_EXIT if there are open connections.
 *              SSC_UNFINISHED_TASKS if there are still tasks that are
 *                                   executing.
 *              SSC_SERVER_NOTRUNNING if the server is not running.
 */
public long stopServer( int runflags );


/**
 * returns the state of the server, i.e. is the server running or not
 *
 * @return SSC_STATE_OPEN if server is up and running
 */
public int getState();




/**
* registerThread registers this user thread to solidDB Linked Library Access server
*
*
* @return           the return value from the server
*                    SSC_SUCCESS            Registration succeeded.
*                    SSC_ERROR              Registration failed.
*                    SSC_INVALID_HANDLE     Invalid local server handle
given.
*                    SSC_SERVER_NOTRUNNING  Server is not running.
*/
public long registerThread();



/**
* unregisterThread unregisters this user thread from the
* solidDB Linked Library Access server
*
*
* @return           the return value from the server
*                    SSC_SUCCESS            Registration succeeded.
*                    SSC_ERROR              Registration failed.
*                    SSC_INVALID_HANDLE     Invalid local server handle given.
*                    SSC_SERVER_NOTRUNNING  Server is not running.
*/
public long unregisterThread();
```

附录. 链接库访问参数

链接库访问参数

本附录提供了用于链接库访问的所有参数的列表。链接库访问参数出现在 solidDB 配置文件（solid.ini）的 [Accelerator] 节中。

有关所有其他 solidDB 参数的描述，请参阅《IBM solidDB 管理员指南》中的相应附录。

请注意，您可以使用下列方法来更改 solidDB 参数:

- 在 solidDB solsql 中输入 ADMIN COMMAND 'parameter' 命令。
- 手动编辑 solid.ini 配置文件。

请注意，在下一次启动服务器之后，使用上述方法对 solid.ini 文件所作的任何更改才会生效。

[Accelerator] 节

表 22. 加速器参数

[Accelerator]	描述	出厂值
ImplicitStart	如果将此参数设置为 yes，那么一旦在用户应用程序中调用了 ODBC API 函数 SQLConnect，此参数就会自动启动 solidDB。如果设置为 no，那么必须通过调用控制 API 函数 SSCStartServer 来显式启动 solidDB。	yes

索引

[B]

备份侦听方式 35
本地应用程序
  已定义 4

[C]

参数
  FileSpec 47

[F]

访问“链接库访问”
  已定义 9
服务器信息
  检索 23

[G]

关闭
  链接库访问 20
管理无盘服务器
  定义 solidDB 配置文件选项 47

[K]

客户机 API 和驱动程序 5
控制 API
  等价的 ADMIN COMMAND 25
  调度函数总结 24
  SSCGetActiveTaskClass（函数） 23
  SSCGetServerHandle（函数） 23
  SSCGetStatusNum（函数） 23
  SSCGetTaskClassState（函数） 23
  SSCIsRunning（函数） 23
  SSCIsThisLocalServer（函数） 23
  SSCSetNotifier（函数） 23
库
  链接库访问 11
  链接库访问的内容 9
  用于远程用户应用程序 9
  solidimpac 12

[L]

连接
  为链接库访问建立连接 14
  ODBC 远程，不启动服务器 21
链接库访问
  访问 9
  关闭 20
  库 11
  链接应用程序 11
  启动 15
  已描述 1
  组件 1
链接应用程序
  为链接库访问链接 11

[M]

密码
  条件 17, 38

[N]

内存
  无盘服务器使用的内存总量 48
  CacheSize（适用于无盘服务器） 48

[Q]

启动的 solidDB
  具有链接库访问 15
驱动程序和客户机 API 5

[R]

任务信息
  检索 23

[S]

事件
  通知函数 23
数据库
  大小 16
数据库 [索引文件]节 47
双方式应用程序
  已定义 4

[T]

同步
  使用 10

[W]

无盘
　无盘引擎的参数设置　47

[Y]

隐式启动　21
应用程序
　为链接库访问准备　12
用户名
　缺省值　17, 38
远程应用程序
　已定义　4

[Z]

状态信息
　检索　23

C

C 应用程序
　样本　10
CacheSize 参数
　为无盘服务器配置　48

F

FileSpec
　（参数）　47
FileSpec_1 参数
　为无盘服务器配置　47

I

IBM 公司提供的配置文件
　配置　47
ImplicitStart（参数）　21, 59

L

Linux
　内存限制　47
Listen 参数
　为无盘服务器配置　49

M

makefile 示例　13

N

netcopy 侦听方式　35

O

ODBC 应用程序
　使用高级复制脚本来构建　11

S

SaConnect
　隐式启动　20
solidctrlstub　5, 6, 7, 9, 24
solidDB 服务器控制（SSC）API　55
　SSC API　55
solidDB 控制 API 控制 API
　已定义　7
solidDB 配置文件
　参数设置　47
　CacheSize（参数）　48
　FileSpec（参数）　47
　Listen（参数）　49
solidDB JDBC API
　已定义　7
solidDB ODBC API
　已定义　6
solidDB SA
　已定义　6
solidimpac　12
SQLConnect
　隐式启动　18
sscapi.h　26, 27
SSCGetServerHandle
　函数描述　27
SSCGetStatusNum
　函数描述　28
SSCIsRunning
　函数描述　29
SSCIsThisLocalServer
　函数描述　29
SSCRegisterThread
　函数描述　30
SSCServerT　16
SSCSetCipher
　函数描述　30
SSCSetNotifier
　函数描述　33
SSCSetState
　函数描述　36
SSCStartDisklessServer
　函数描述　37
SSCStartServer
　函数描述　40
　显式启动　16
SSCStopServer
　关闭　18
　函数描述　43
SscTaskSetT　26

SSCUnregisterThread
函数描述 45
SSC_ABORT 27
SSC_CALL 25
SSC_CONNECTIONS_EXIST 27
SSC_CONT 27
SSC_ERROR 27
SSC_FINISHED 27
SSC_INFO_SERVER_RUNNING 27
SSC_INVALID_HANDLE 27
SSC_INVALID_LICENSE 27
SSC_NODATABASEFILE 27
SSC_SERVER_INNETCOPYMODE 27
SSC_SERVER_NOTRUNNING 27
SSC_STATE_OPEN 37, 38, 41
SSC_STATE_PREFETCH 37, 41
SSC_SUCCESS 27
SSC_TASK_ALL 26
SSC_TASK_BACKUP 26
SSC_TASK_CHECKPOINT 26
SSC_TASK_HOTSTANDBY 26
SSC_TASK_HOTSTANDBY_CATCHUP 26
SSC_TASK_LOCALUSERS 26
SSC_TASK_MERGE 26
SSC_TASK_NONE 26
SSC_TASK_REMOTEUSERS 26
SSC_TASK_SYNC_HISTCLEAN 26
SSC_TASK_SYNC_MESSAGE 26
SSC_UNFINISHED_TASKS 27

[特别字符]

[索引文件]节
为无盘服务器配置 47
[Com] 节
为无盘服务器配置 49

声明

除非经过 Solid Information Technology Ltd. 或者 International Business Machines Corporation 书面授权，否则不能以任何方式使用本产品中的任何部分。

本产品受美国专利 6144941、7136912、6970876、7139775、6978396 和 7266702 的保护。

为此产品指定的美国出口管制分类编号是 ECCN=5D992b。

本信息是为在美国提供的产品和服务编写的。

IBM 可能在其他国家或地区不提供本文档中讨论的产品、服务或功能特性。有关您当前所在区域的产品和服务的信息，请向您当地的 IBM 代表咨询。任何对 IBM 产品、程序或服务的引用并非意在明示或暗示只能使用 IBM 的产品、程序或服务。只要不侵犯 IBM 的知识产权，任何同等功能的产品、程序或服务，都可以代替 IBM 产品、程序或服务。但是，评估和验证任何非 IBM 产品、程序或服务，则由用户自行负责。

IBM 公司可能已拥有或正在申请与本文档内容有关的各项专利。提供本文档并未授予用户使用这些专利的任何许可。您可以用书面方式将许可查询寄往:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

有关双字节（DBCS）信息的许可查询，请与您所在国家或地区的 IBM 知识产权部门联系，或用书面方式将查询寄往:

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106-0032, Japan

本条款不适用英国或任何这样的条款与当地法律不一致的国家或地区：INTERNATIONAL BUSINESS MACHINES CORPORATION“按现状”提供本出版物，不附有任何种类的（无论是明示的还是暗含的）保证，包括但不限于暗含的有关非侵权、适销和适用于某种特定用途的保证。某些国家或地区在某些交易中不允许免除明示或暗含的保证。因此本条款可能不适用于您。

本信息中可能包含技术方面不够准确的地方或印刷错误。此处的信息将定期更改；这些更改将编入本资料的新版本中。IBM 可以随时对本资料中描述的产品和/或程序进行改进和/或更改，而不另行通知。

本信息中对非 IBM Web 站点的任何引用都只是为了方便起见才提供的，不以任何方式充当对那些 Web 站点的保证。那些 Web 站点中的资料不是 IBM 产品资料的一部分，使用那些 Web 站点带来的风险将由您自行承担。

IBM 可以按它认为适当的任何方式使用或分发您所提供的任何信息而无须对您承担任何责任。

本程序的被许可方如果要了解有关程序的信息以达到如下目的: （i）允许在独立创建的程序和其他程序（包括本程序）之间进行信息交换，以及（ii）允许对已经交换的信息进行相互使用，请与下列地址联系:

IBM Canada Limited
Office of the Lab Director
8200 Warden Avenue
Markham, Ontario
L6G 1C7
CANADA

只要遵守适当的条件和条款，包括某些情形下的一定数量的付费，都可获得这方面的信息。

本资料中描述的许可程序及其所有可用的许可资料均由 IBM 依据 IBM 客户协议、IBM 国际软件许可协议或任何同等协议中的条款提供。

此处包含的任何性能数据都是在受控环境中测得的。因此，在其他操作环境中获得的数据可能会有明显的不同。有些测量可能是在开发级的系统上进行的，因此不保证与一般可用系统上进行的测量结果相同。此外，有些测量是通过推算而估计的，实际结果可能会有差异。本文档的用户应当验证其特定环境的适用数据。

涉及非 IBM 产品的信息可从这些产品的供应商、其出版说明或其他可公开获得的资料中获取。IBM 没有对这些产品进行测试，也无法确认其性能的精确性、兼容性或任何其他关于非 IBM 产品的声明。有关非 IBM 产品性能的问题应当向这些产品的供应商提出。

所有关于 IBM 未来方向或意向的声明都可随时更改或收回，而不另行通知，它们仅仅表示了目标和意愿而已。

本信息包含在日常业务操作中使用的数据和报告的示例。为了尽可能完整地说明这些示例，示例中可能会包括个人、公司、品牌和产品的名称。所有这些名字都是虚构的，若现实生活中实际业务企业使用的名字和地址与此相似，纯属巧合。

版权许可:

本信息包括源语言形式的样本应用程序，这些样本说明不同操作平台上的编程方法。如果是为按照在编写样本程序的操作平台上的应用程序编程接口（API）进行应用程序的开发、使用、经销或分发为目的，您可以任何形式对这些样本程序进行复制、修改、分发，而无须向 IBM 付费。这些示例并未在所有条件下作全面测试。因此，IBM 不能担保或暗示这些程序的可靠性、可维护性或功能。

凡这些实例程序的每份拷贝或其任何部分或任何衍生产品，都必须包括如下版权声明:

©（贵公司的名称）（年）。此部分代码是根据 IBM Corp. 公司的样本程序衍生出来的。

商标

IBM、IBM 徽标、ibm.com®、Solid®、solidDB、InfoSphere™、DB2®、Informix® 和 WebSphere® 是 International Business Machines Corporation 在美国和/或其他国家或地区的商标或注册商标。如果这些商标和其他 IBM 注册商标在本资料中第一次出现时标有商标符号（® 或 ™），那么这些符号表示它们是发布本资料时归 IBM 所有的经过美国政府注册的商标或普通法商标。这些商标也可能是在其他国家或地区的注册商标或普通法商标。在 Web 上的版权和商标信息（www.ibm.com/legal/copytrade.shtml）处提供了 IBM 商标的最新列表。

Java 和所有基于 Java 的商标和徽标是 Sun Microsystems, Inc. 在美国和/或其他国家或地区的商标。

Linux 是 Linus Torvalds 在美国和/或其他国家或地区的注册商标。

Microsoft and Windows 和 Microsoft Corporation 在美国和/或其他国家或地区的注册商标。

UNIX 是 The Open Group 在美国和其他国家或地区的注册商标。

其他公司、产品或服务名称可能是其他公司的商标或服务标记。

中国印刷

S151-1147-00