SYBASE iAnywhere

MobiLink
客户端管理

2009 年 2 月

11.0.1 版

版权和商标

目录

关于本手册

主题

本手册介绍如何设置、配置和同步 MobiLink 客户端。MobiLink 客户端可以是 SQL Anywhere 或者 UltraLite 数据库。本手册同时也介绍了 Dbmlsync API，通过它可无缝地将同步集成到 C++ 或 .NET 客户端应用程序中。

目标读者

本手册适用于要向信息系统添加同步的用户。

开始之前

有关 MobiLink 与其它同步和复制技术的比较，请参见“数据交换技术概述” 《SQL Anywhere 11 - 简介》。

关于 SQL Anywhere 文档

完整的 SQL Anywhere 文档以四种形式提供，但所包含信息均相同。

- **HTML 帮助** 联机帮助文档包含完整的 SQL Anywhere 文档，其中包括手册和 SQL Anywhere 工具的上下文相关帮助。

  如果使用 Microsoft Windows 操作系统，则联机帮助文档以 HTML 帮助 (CHM) 格式提供。若要访问此文档，请选择 [开始] » [程序] » [**SQL Anywhere 11**] » [文档] » [联机手册]。

  管理工具使用同一联机文档来实现帮助功能。

- **Eclipse** 在 Unix 平台上以 Eclipse 格式提供完整的联机帮助。要访问文档，请从 SQL Anywhere 11 安装的 *bin32* 或 *bin64* 目录下运行 *sadoc*。

- **DocCommentXchange** DocCommentXchange 是一个用于访问和讨论 SQL Anywhere 文档的社区。

  使用 DocCommentXchange 可以执行以下任务：

  - 查看文档
  - 检查是否有用户对文档各部分所做出的阐明
  - 提供建议和修正意见以在将来的版本中为所有用户改进文档

  访问 http://dcx.sybase.com。

- **PDF** 整套 SQL Anywhere 手册会以一组 Portable Document Format (PDF) 文件的形式提供。您必须有 PDF 阅读器才能查看信息。要下载 Adobe Reader，请访问 http://get.adobe.com/reader/。

  若要在 Microsoft Windows 操作系统上访问 PDF 文档，请选择 [开始] » [程序] » [**SQL Anywhere 11**] » 文档 » [联机手册 - PDF 格式]。

  要在 Unix 操作系统上访问 PDF 文档，请使用 Web 浏览器打开 *install-dir/documentation/zh/pdf/index.html*。

关于文档集中的手册

SQL Anywhere 文档由以下手册组成：

- **SQL Anywhere 11 - 简介** 本手册介绍 SQL Anywhere 11，一个提供数据管理和数据交换技术的综合数据包，通过它可以为服务器环境、台式机环境、移动环境以及远程办公环境快速开发由数据库驱动的应用程序。

- **SQL Anywhere 11 - 更改和升级** 本手册介绍 SQL Anywhere 11 以及该软件以前版本中的新功能。

- **SQL Anywhere 服务器 - 数据库管理** 本手册介绍如何运行、管理及配置 SQL Anywhere 数据库。它介绍了数据库连接、数据库服务器、数据库文件、备份过程、安全性、高可用性、使用复制服务器进行复制以及管理实用程序和选项。

- **SQL Anywhere 服务器 - 编程** 本手册介绍如何使用 C、C++、Java、PHP、Perl、Python 和 .NET 编程语言（例如 Visual Basic 和 Visual C#）建立和部署数据库应用程序。其中介绍了各种编程接口，如 ADO.NET 和 ODBC。
- **SQL Anywhere 服务器 - SQL 参考** 本手册提供了系统过程和目录（系统表和视图）的参考信息。也介绍了 SQL 语言（搜索条件、语法、数据类型和函数）的 SQL Anywhere 实现。
- **SQL Anywhere 服务器 - SQL 的用法** 本手册介绍如何设计和创建数据库；如何导入、导出和修改数据；如何检索数据以及如何建立存储过程和触发器。
- **MobiLink - 入门** 本手册介绍基于会话的关系数据库同步系统 MobiLink。MobiLink 技术支持双向复制并且非常适用于移动计算环境。
- **MobiLink - 客户端管理** 本手册介绍如何设置、配置和同步 MobiLink 客户端。MobiLink 客户端可以是 SQL Anywhere 或者 UltraLite 数据库。本手册同时也介绍了 Dbmlsync API，通过它可无缝地将同步集成到 C++ 或 .NET 客户端应用程序中。
- **MobiLink - 服务器管理** 本手册说明如何设置和管理 MobiLink 应用程序。
- **MobiLink - 服务器启动的同步** 本手册介绍 MobiLink 服务器启动的同步，这种功能允许 MobiLink 服务器启动同步或在远程设备上进行操作。
- **QAnywhere** 本手册介绍 QAnywhere，一个用于移动、无线、台式机和膝上型客户端的消息传递平台。
- **SQL Remote** 本手册介绍用于移动计算的 SQL Remote 数据复制系统，此系统支持使用电子邮件或文件传输等间接链接共享 SQL Anywhere 统一数据库和多个 SQL Anywhere 远程数据库之间的数据。
- **UltraLite - 数据库管理和参考** 本手册介绍适用于小型设备的 UltraLite 数据库系统。
- **UltraLite - C 及 C++ 编程** 本手册介绍 UltraLite C 和 C++ 编程接口。利用 UltraLite，可以开发数据库应用程序，并将它们部署到手持式设备、移动设备或嵌入式设备。
- **UltraLite - M-Business Anywhere 编程** 本手册介绍 UltraLite for M-Business Anywhere。利用 UltraLite for M-Business Anywhere，用户可以开发基于 Web 的数据库应用程序，并将它们部署到运行 Palm OS、Windows Mobile 或 Windows 的手持式设备、移动设备或嵌入式设备。
- **UltraLite - .NET 编程** 本手册介绍 UltraLite.NET。利用 UltraLite.NET，您可以开发数据库应用程序，并将它们部署到计算机、手持式设备、移动设备或嵌入式设备。
- **UltraLiteJ** 本手册介绍 UltraLiteJ。利用 UltraLiteJ，可以在支持 Java 的环境中开发和部署数据库应用程序。UltraLiteJ 支持 BlackBerry 智能手机和 Java SE 环境。UltraLiteJ 基于 iAnywhere UltraLite 数据库产品。
- **错误消息** 本手册提供了 SQL Anywhere 错误消息及其诊断信息的完整列表。

文档约定

本节列出了本文档中使用的约定。

操作系统

SQL Anywhere 可以在各种平台上运行。在大多数情况下，该软件在所有平台上的行为都是相同的，但也有变动或限制。这些变动或限制通常基于基础操作系统（Windows、Unix），很少基于特定变型（AIX、Windows Mobile）或版本。

为了简化对操作系统的提及，本文档按如下方式对支持的操作系统进行分组：

- **Windows** Microsoft Windows 系列包括 Windows Vista 和 Windows XP（主要用于服务器、台式计算机和膝上型计算机），以及 Windows Mobile（用于移动设备）。

  除非另外指定，否则当本文档提及 Windows 时，是指所有基于 Windows 的平台，包括 Windows Mobile。

- **Unix** 除非另外指定，否则当本文档提及 Unix 时，是指所有基于 Unix 的平台，包括 Linux 和 Mac OS X。

目录和文件名

大部分情况下，对目录和文件名的引用在所有支持的平台上都是类似的，只需在不同形式之间进行简单的转换。这时需使用 Windows 约定。在细节更为复杂的情况下，文档显示所有相关形式。

下面是文档编写中用于简化目录和文件名的约定：

- **大写和小写目录名** 在 Windows 和 Unix 上，目录和文件名可以包括大写和小写字母。创建目录和文件时，文件系统会保留字母大小写。

  在 Windows 上，对目录和文件的提及*不区分大小写*。混合使用大小写的目录和文件名很常见，但使用所有小写字母来提及目录和文件的形式也很常见。SQL Anywhere 安装包含诸如 *Bin32* 和 *Documentation* 的目录。

  在 Unix 上，对目录和文件的提及*区分大小写*。混合使用大小写的目录和文件名不常见。大多数的目录和文件名全部使用小写字母。SQL Anywhere 安装包含诸如 *bin32* 和 *documentation* 的目录。

  本文档采用 Windows 形式的目录名。大多数情况下，在 Unix 上可以将大小写混合形式的目录名转换成小写字母的等效目录名。

- **分隔目录和文件名的斜线** 文档使用反斜线作为目录分隔符。例如，PDF 格式的文档位于 *install-dir\Documentation\zh\PDF*（Windows 形式）。

  在 Unix 上，用正斜线替换反斜线。PDF 文档位于 *install-dir/documentation/zh/pdf* 下。

- **可执行文件** 文档使用 Windows 约定显示可执行文件名（带有诸如 *.exe* 或 *.bat* 后缀）。在 Unix 上，可执行文件名没有后缀。

  例如，在 Windows 上，网络数据库服务器是 *dbsrv11.exe*。在 Unix 上是 *dbsrv11*。

- ***install-dir*** 在安装过程中，选择 SQL Anywhere 的安装位置。创建环境变量 SQLANY11，用来表示此位置。文档中以 *install-dir* 表示此位置。

  例如，本文档将此文件表示为 *install-dir*\readme.txt。在 Windows 上，这等同于 *%SQLANY11%\readme.txt*。在 Unix 上，这等同于 *$SQLANY11/readme.txt* 或 *${SQLANY11}/readme.txt*。

有关 *install-dir* 缺省位置的详细信息，请参见“SQLANY11 环境变量”一节 《SQL Anywhere 服务器 - 数据库管理》。

- ***samples-dir*** 在安装过程中，选择 SQL Anywhere 随附的示例的安装位置。创建环境变量 SQLANYSAMP11，用来表示此位置。文档中以 *samples-dir* 表示此位置。

  要在 *samples-dir* 中打开 Windows 资源管理器窗口，请在 [**开始**] 菜单中，选择 [**程序**] » [**SQL Anywhere 11**] » [**示例应用程序和项目**]。

  有关 *samples-dir* 缺省位置的详细信息，请参见“SQLANYSAMP11 环境变量”一节 《SQL Anywhere 服务器 - 数据库管理》。

命令提示符和命令 shell 语法

大多数操作系统都提供一种或多种使用命令 shell 或命令提示符来输入命令和参数的方法。Windows 命令提示符包括 Command Prompt（DOS 提示符）和 4NT。Unix 命令 shell 包括 Korn shell 和 bash。每个 shell 都具有一些功能，其能力不仅仅局限于简单命令。这些功能通过特殊字符来驱动。特殊字符和功能随 shell 的不同而不同。如果没有正确使用这些特殊字符，通常会导致语法错误或意外行为。

本文档以普通形式提供命令行示例。如果这些示例中包含 shell 的特殊字符，则命令需要根据特定 shell 进行修改。修改方法不在本文档所述范围之内，但通常是在包含这些特殊字符的参数两旁加上引号，或是在特殊字符前面使用转义字符。

下面是命令行语法的一些示例，不同的平台可能会有不同的形式：

- **括号和大括号** 有些命令行选项需要一个参数，该参数将以列表形式接受详细的值指定。该列表通常用括号或大括号括起来。本文档使用括号。例如：

  ```
  -x tcpip(host=127.0.0.1)
  ```

  如果括号导致出现语法问题，用大括号替代：

  ```
  -x tcpip{host=127.0.0.1}
  ```

  如果两种形式都将产生语法问题，应按照 shell 的要求，用引号将整个参数括起来：

  ```
  -x "tcpip(host=127.0.0.1)"
  ```

- **引号** 如果必须在参数值中指定引号，该引号可能会与用于括参数的引号的传统用法发生冲突。例如，要指定值中包含双引号的加密密钥，则可能必须用引号括起密钥，然后转义嵌入的引号：

  ```
  -ek "my \"secret\" key"
  ```

  在许多 shell 中，密钥的值为 `my "secret" key`。

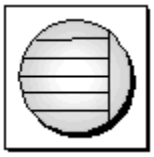
- **环境变量** 本文档介绍设置环境变量。在 Windows shell 中，环境变量使用语法 *%ENVVAR%* 来指定。在 Unix shell 中，环境变量使用语法 *$ENVVAR* 或 *${ENVVAR}* 来指定。

图标

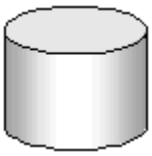
本文档中使用了下列图标。

- 客户端应用程序。

- 数据库服务器，如 Sybase SQL Anywhere。



- 数据库。在某些高水平的图中，可以使用此图标表示数据库和管理该数据库的数据库服务器。



- 复制或同步中间件。用于帮助在数据库之间共享数据。例如 MobiLink 服务器和 SQL Remote 消息代理。

- 编程接口。

联系文档小组

我们欢迎您就本帮助文档提出意见、建议和反馈信息。

要提交意见和建议，请发送电子邮件到 SQL Anywhere 文档小组，地址为 iasdoc@sybase.com。虽然我们不对这些电子邮件进行回复，但您的反馈会帮助我们改进文档，因此我们真诚地欢迎您提出宝贵的意见和建议。

DocCommentXchange

也可以使用 DocCommentXchange 将意见或建议直接置于帮助主题中。DocCommentXchange (DCX) 是一个用于访问和讨论 SQL Anywhere 文档的社区。使用 DocCommentXchange 可以执行以下任务：

- 查看文档
- 检查是否有用户对文档各部分所做出的阐明
- 提供建议和修正意见以在将来的版本中为所有用户改进文档

访问 http://dcx.sybase.com。

查找详细信息并请求技术支持

附加信息和资源可从 Sybase iAnywhere 开发人员社区获得，网址是 http://www.sybase.com/developer/library/sql-anywhere-techcorner。

如果您有问题或是需要帮助，可将邮件发布到下面所列的 Sybase iAnywhere 新闻组。

当您向这些新闻组发布邮件时，请务必提供问题的详细信息，包括 SQL Anywhere 版本的内部版本号。可以通过运行以下命令找到此信息：**dbeng11 -v**。 **dbeng11 -v**.

新闻组位于 *forums.sybase.com* 新闻服务器上。

这些新闻组包括：

- sybase.public.sqlanywhere.general
- sybase.public.sqlanywhere.linux
- sybase.public.sqlanywhere.mobilink
- sybase.public.sqlanywhere.product_futures_discussion
- sybase.public.sqlanywhere.replication
- sybase.public.sqlanywhere.ultralite
- ianywhere.public.sqlanywhere.qanywhere

有关 Web 开发问题，请访问 http://groups.google.com/group/sql-anywhere-web-development。

新闻组免责声明

iAnywhere Solutions 没有义务为其新闻组提供解决方案、信息或建议，除提供系统操作员监控服务和确保新闻组的运行和可用性外，iAnywhere Solutions 也没有义务提供任何其它服务。

如果时间允许，iAnywhere 技术顾问以及其他员工也会对新闻组服务提供帮助。他们是在自愿的基础上提供帮助的，所以可能无法定期提供解决方案和信息。他们可以提供多少帮助取决于他们的工作量。

MobiLink 客户端简介

本节介绍可用于 MobiLink 同步的客户端，并提供所有类型 MobiLink 客户端所共有的信息。

MobiLink 客户端

目录

SQL Anywhere 客户端

要将 SQL Anywhere 数据库用作 MobiLink 客户端，需将同步对象添加到该数据库。需要添加的对象包括发布、MobiLink 用户以及将发布连接到用户的预订。请参见：

- “创建远程数据库”一节第 90 页
- “发布数据”一节第 94 页
- “创建 MobiLink 用户”一节第 101 页
- “创建同步预订”一节第 104 页

通过运行名为 dbmlsync 的命令行实用程序可启动同步。此实用程序将连接到远程数据库，然后通常利用远程数据库的事务日志中包含的信息准备上载。（也可执行脚本式上载，但不使用事务日志。）dbmlsync 实用程序可利用存储在同步发布及同步预订中的信息，与 MobiLink 服务器连接并交换数据。

请参见“启动同步”一节第 106 页。

有关 SQL Anywhere 客户端的详细信息，请参见“SQL Anywhere 客户端”第 89 页。

有关 dbmlsync 命令行选项的详细信息，请参见“MobiLink SQL Anywhere 客户端实用程序 (dbmlsync)”第 121 页。

自定义同步

请参见“自定义 dbmlsync 同步”一节第 116 页。

UltraLite 客户端

只要该应用程序包含一个对相应同步函数的调用，UltraLite 应用程序便会自动启用 MobiLink。

UltraLite 应用程序和库将在应用程序端处理同步操作。您在编写 UltraLite 应用程序时几乎不用考虑同步。UltraLite 运行时将跟踪在上次同步之后发生的更改。

在使用 TCP/IP、HTTP、HTTPS 或 ActiveSync 时，只需调用一次同步函数，即可从应用程序启动同步。而 HotSync 的接口稍有不同。从应用程序或 HotSync 启动同步后，MobiLink 服务器与 UltraLite 运行时即会控制同步过程中发生的操作。

另请参见

- UltraLite - 数据库管理和参考
- “UltraLite 客户端” 《UltraLite - 数据库管理和参考》
- “UltraLiteJ 客户端”一节第 6 页
- “使用 UltraLite 的 ActiveSync 和 HotSync” 《UltraLite - 数据库管理和参考》
- “UltraLite 同步参数和网络协议选项” 《UltraLite - 数据库管理和参考》

UltraLiteJ 客户端

UltraLiteJ 向 Java 应用程序提供了 MobiLink 同步客户端以及更改跟踪和状态跟踪功能，以确保能够进行可靠的同步。只要该应用程序包含一个对相应同步函数的调用，UltraLiteJ 应用程序便会自动启用 MobiLink。

UltraLiteJ 应用程序和库在应用程序端处理同步操作。您在编写 UltraLite 应用程序时几乎不用考虑同步。UltraLiteJ 运行时将跟踪在上次同步之后发生的更改。

在使用 HTTP 或 HTTPS 时，只需调用一次同步函数，即可从应用程序启动同步。

MobiLink 文件传输实用程序 (mlfiletransfer) 和 SQL 直通功能不适用于 UltraLiteJ 客户端。

另请参见

- “数据同步”一节 《UltraLiteJ》
- “将 UltraLiteJ 用作 MobiLink 客户端”一节 《UltraLiteJ》
- UltraLite - 数据库管理和参考
- “UltraLite 客户端” 《UltraLite - 数据库管理和参考》
- “UltraLite 同步参数和网络协议选项” 《UltraLite - 数据库管理和参考》

为客户端指定网络协议

MobiLink 服务器使用 -x 命令行选项，为同步客户端指定一个或多个连接到 MobiLink 服务器所需的网络协议。所选的网络协议必须与客户端使用的同步协议相匹配。

mlsrv11 命令行选项的语法是：

mlsrv11 -c "*connection-string***" -x** ***protocol*****(** ***options*** **)**

在下面的示例中，选择了不带任何附加协议选项的 TCP/IP 协议。

```
mlsrv11 -c "dsn=SQL Anywhere 11 Demo" -x tcpip
```

可用如下格式的选项对协议进行配置：

(*keyword=value*;...**)**

例如：

```
mlsrv11 -c "dsn=SQL Anywhere 11 Demo" -x tcpip(
   host=localhost;port=2439)
```

另请参见

有关 MobiLink 网络协议和协议选项的完整的详细信息，可在以下位置找到：

要查找……	**请参见……**
如何为 MobiLink 服务器设置网络选项	“-x 选项”一节 《MobiLink - 服务器管理》
可用于 MobiLink 客户端应用程序的所有网络协议选项	“MobiLink 客户端网络协议选项汇总”一节第 33 页
如何为 SQL Anywhere 客户端设置选项	“CommunicationAddress (adr) 扩展选项”一节第 179 页 “CommunicationType (ctp) 扩展选项”一节第 180 页
如何为 UltraLite 客户端设置选项	“Stream Parameters 同步参数”一节 《UltraLite - 数据库管理和参考》 “Stream Type 同步参数”一节 《UltraLite - 数据库管理和参考》 “UltraLite 同步实用程序 (ulsync)”一节 《UltraLite - 数据库管理和参考》

MobiLink 中的系统表

MobiLink 服务器系统表

将某个数据库设置为用作统一数据库时，MobiLink 服务器所需的 MobiLink 系统表即会创建。

请参见“MobiLink 服务器系统表”《MobiLink - 服务器管理》。

UltraLite 系统表

请参见“UltraLite 系统表”《UltraLite - 数据库管理和参考》。

SQL Anywhere 系统表

不能直接访问 SQL Anywhere 系统表，但可通过系统视图访问它们。请参见“系统视图”一节《SQL Anywhere 服务器 - SQL 参考》。

MobiLink 用户对以下 SQL Anywhere 系统视图特别感兴趣：

- “SYSSYNC 系统视图”一节 《SQL Anywhere 服务器 - SQL 参考》
- “SYSPUBLICATION 系统视图”一节 《SQL Anywhere 服务器 - SQL 参考》
- “SYSSUBSCRIPTION 系统视图”一节 《SQL Anywhere 服务器 - SQL 参考》
- “SYSSYNCSCRIPT 系统视图”一节 《SQL Anywhere 服务器 - SQL 参考》

SQL Anywhere 还提供统一视图，用于查询系统视图以提供您可能需要的信息。请参见“统一视图”一节 《SQL Anywhere 服务器 - SQL 参考》。

MobiLink 用户

目录

MobiLink 用户简介

MobiLink 用户（也称为**同步用户**）是您在连接 MobiLink 服务器时用于进行验证的名称。

为使用户名成为同步系统的一部分：

- 必须在远程数据库上创建 MobiLink 用户名。
- 必须使用 MobiLink 服务器注册 MobiLink 用户名。

MobiLink 用户名和口令与数据库用户名和口令不同。MobiLink 用户名用于验证从远程数据库到 MobiLink 服务器的连接。

还可以使用用户名来控制 MobiLink 服务器的行为。将 **username** 参数用于同步脚本中即可实现这一控制目的。

请参见“在脚本中使用远程 ID 和 MobiLink 用户名”一节第 15 页。

MobiLink 用户名存储在统一数据库 ml_user MobiLink 系统表的名称列中。

在同步系统内，MobiLink 用户名不必唯一。如果安全性不是问题的话，您甚至可以给每个远程数据库指派相同的 MobiLink 用户名。

UltraLite 用户验证

尽管 UltraLite 和 MobiLink 的用户验证模式相互独立，为简单起见您仍可能希望将 UltraLite 用户 ID 的值与 MobiLink 用户名共享。仅当只有单个用户使用 UltraLite 应用程序时才可以这样做。

请参见“UltraLite 用户验证”一节 《UltraLite - 数据库管理和参考》。

创建和注册 MobiLink 用户

MobiLink 用户在远程数据库中创建，在统一数据库中注册。

在远程数据库中创建 MobiLink 用户

若要在远程数据库中添加用户，可采用以下方法：

- 对于 SQL Anywhere 远程数据库，请使用 Sybase Central 或 CREATE SYNCHRONIZATION USER 语句。

  请参见“创建 MobiLink 用户”一节第 101 页。

- 对于 UltraLite 远程数据库，请设置用户名和密码同步参数。

  请参见“User Name 同步参数”一节 《UltraLite - 数据库管理和参考》和“Password 同步参数”一节 《UltraLite - 数据库管理和参考》。

向统一数据库添加 MobiLink 用户名

在远程数据库中创建用户名之后，可以使用以下方法中的任意一种在统一数据库中注册这些用户名：

- 使用 mluser 实用程序。

  请参见“MobiLink 用户验证实用程序 (mluser)”一节 《MobiLink - 服务器管理》。

- 使用 Sybase Central。

- 为 authenticate_user 事件或 authenticate_user_hashed 事件实现一个脚本。调用这些脚本中的任何一个时，MobiLink 服务器都会自动添加验证成功的用户。

  请参见“authenticate_user 连接事件”一节 《MobiLink - 服务器管理》
  或“authenticate_user_hashed 连接事件”一节 《MobiLink - 服务器管理》。

- 指定 mlsrv11 的 -zu+ 命令行选项。在这种情况下，任何尚未添加到统一数据库中的现有 MobiLink 用户都会在第一次进行同步时添加。此选项在开发时有用，但不建议在已部署的应用程序中使用。

  请参见“-zu 选项”一节 《MobiLink - 服务器管理》。

为用户提供初始口令

每个用户的口令都与用户名一起存储在 ml_user 表中。可以使用 Sybase Central 或 mluser 命令行实用程序来提供初始口令。

Sybase Central 可以很方便地添加单个用户和口令。而 mluser 实用程序则适用于成批添加。

如果创建了无口令的用户，则 MobiLink 将不会验证该用户，而且在连接和同步时也不需要口令。

♦ 为用户提供初始 MobiLink 口令（Sybase Central 管理模式）

1. 使用 MobiLink 插件从 Sybase Central 连接到统一数据库。
2. 选择 [**模式**] » [**管理**]。
3. 单击 [**用户**]。
4. 选择 [**文件**] » [**新建**] » [**用户**]。
5. 按照 [**创建用户向导**] 中的说明进行操作。

♦ 提供初始 MobiLink 口令（命令行）：

1. 创建一个文件，文件中每行包含一个用户名和口令，用空格分开。
2. 打开命令提示符，然后运行 mluser 命令行实用程序。例如：

   ```
   mluser -c "dsn=my_dsn" -f password-file
   ```

   在此命令行中，-c 选项指定了一个到统一数据库的 ODBC 连接。-f 选项指定了包含用户名和口令的文件。

   请参见“MobiLink 用户验证实用程序 (mluser)”一节 《MobiLink - 服务器管理》。

新用户的同步

通常，每个 MobiLink 客户端必须提供有效的 MobiLink 用户名和口令以连接到 MobiLink 服务器。

如果在启动 MobiLink 服务器时设置了 -zu+ 选项，服务器会接受并响应来自未注册用户的同步请求。接收到来自 ml_user 表之外其他用户的请求时，该请求会被处理，而该用户也会添加到 ml_user 表中。

使用 -zu+ 时，如果 MobiLink 客户端使用当前 ml_user 表中不存在的用户名进行同步，缺省情况下，MobiLink 将采取以下操作：

- **新用户，无口令** 如果用户未提供口令，缺省情况下该用户名将以空口令添加到 ml_user 表中。此用户无需提供口令就可以进行同步。
- **新用户，口令** 如果用户提供了口令，则该用户名和口令都将被添加到 ml_user 表中，这样新用户名将在 MobiLink 系统中成为可识别的用户名。以后，此用户必须指定相同的口令才能同步。
- **新用户，新口令** 新用户可在新口令字段或口令字段中提供信息。无论何种情况，新口令的设置都将覆盖旧口令的设置，并使用新口令将新用户添加到 MobiLink 系统中。以后，此用户必须指定相同的口令才能同步。

请参见“-zu 选项”一节 《MobiLink - 服务器管理》。

未知用户阻止同步

缺省情况下，MobiLink 服务器仅识别 ml_user 表中注册的用户。此缺省设置有两个优点。第一，降低了未经授权访问 MobiLink 服务器的风险。第二，防止授权用户意外地使用错误用户名或拼错的用户名进行连接。应该避免此类意外，因为这些意外可能使 MobiLink 系统产生不可预知的行为。

提示最终用户输入口令

最终用户每次从 MobiLink 客户端同步时必须提供 MobiLink 用户名和口令，除非您在 MobiLink 服务器上选择禁用用户验证。

♦ **提示最终用户输入 MobiLink 口令：**

- UltraLite 和 SQL Anywhere 客户端提供用户名和口令的机制是不同的。
  - **UltraLite** 同步时，UltraLite 客户端必须在同步结构的口令字段中提供有效值。对于内部 MobiLink 同步而言，口令要与 MobiLink 系统表 ml_user 中的值匹配才算有效。

    在同步之前应用程序应提示最终用户输入 MobiLink 用户名和口令。

    请参见“UltraLite 同步参数和网络协议选项” 《UltraLite - 数据库管理和参考》。
  - **SQL Anywhere** 用户可以在 dbmlsync 命令行中提供有效口令。如果未在命令行中提供有效口令，dbmlsync 连接窗口会提示用户提供有效口令。后一种方式更加安全，因为命令行对同一台计算机上运行的其它进程是可见的。

    如果验证失败，则将提示用户重新输入用户名和口令。

请参见“-c 选项”一节第 134 页。

更改口令

MobiLink 提供了一种机制供最终用户更改口令。UltraLite 和 SQL Anywhere 客户端的接口是不同的。

♦ **提示最终用户输入 MobiLink 口令：**

- UltraLite 和 SQL Anywhere 客户端提供用户名和口令的机制不同。
  - **SQL Anywhere** 在 dbmlsync 命令行中提供有效的现有口令和新口令，如果您不使用命令行参数，则需要在 dbmlsync 连接窗口中提供。

    请参见“-mp 选项”一节第 150 页和“-mn 选项”一节第 149 页。
  - **UltraLite** 同步时，应用程序必须在同步结构的 password 字段中提供现有口令，在 new_password 字段中提供新口令。

    请参见“Password 同步参数”一节 《UltraLite - 数据库管理和参考》和“New Password 同步参数”一节 《UltraLite - 数据库管理和参考》。

初始口令可以在统一服务器中或进行第一次尝试执行同步时设置。请参见“为用户提供初始口令”一节第 11 页和“新用户的同步”一节第 12 页。

一旦指派了口令，您就不能从客户端将口令重置为空字符串。

远程 ID

远程 ID 是远程数据库在 MobiLink 同步系统中的唯一标识。

当创建 SQL Anywhere 或 UltraLite 数据库时，其远程 ID 的值为空值。当数据库与 MobiLink 同步时，MobiLink 会检查是否具有空值的远程 ID。如果找到，则会将 GUID 指派为远程 ID。经过设置后，在手工更改前，数据库会一直维护相同的远程 ID。

对于 SQL Anywhere 远程数据库，MobiLink 服务器通过远程 ID 和预订跟踪同步进程。对于 UltraLite 数据库，MobiLink 服务器通过远程 ID 和发布跟踪同步进程。此信息存储在 ml_subscription 系统表中。远程 ID 也会在每次同步时记录在 MobiLink 服务器日志中。

如果在两个或多个并行同步中使用相同的远程 ID，MobiLink 服务器会发出一个错误消息。

另请参见

- “ml_subscription”一节 《MobiLink - 服务器管理》

设置 MobiLink 远程 ID

虽然系统将远程 ID 创建为 GUID，但您可以将它更改为更具意义的名称。对于 SQL Anywhere 和 UltraLite 数据库，远程 ID 会作为名为 ml_remote_id 的属性存储在数据库中。

有关 SQL Anywhere 客户端的信息，请参见“设置远程 ID”一节第 91 页。

有关 UltraLite 客户端的信息，请参见“UltraLite ml_remote_id 选项”一节 《UltraLite - 数据库管理和参考》。

将启动数据库部署到多个位置时，最安全的方法是部署远程 ID 为 NULL 的数据库。如果已通过同步将数据库进行了预填充，可先将远程 ID 的设置恢复为 NULL，然后再进行部署。此方法确保了远程 ID 的唯一性，因为远程数据库首次进行同步时系统会为其指派唯一的远程 ID。也可以将远程 ID 作为一个远程设置步骤来进行设置，但它必须是唯一的。

示例

在对每个远程数据库对应一个用户的 MobiLink 设置进行定义时，为了简化管理任务，最好对每个远程数据库上的所有三个 MobiLink 标识符使用相同的编号。例如，在 SQL Anywhere 远程数据库中，您可按如下方式设置它们：

```
-- Set the MobiLink user name:
  CREATE SYNCHRONIZATION USER "1" ... ;

-- Set the partition number for DEFAULT GLOBAL AUTOINCREMENT:
  SET OPTION PUBLIC.GLOBAL_DATABASE_ID = '1';

-- Set the MobiLink remote ID:
  SET OPTION PUBLIC.ml_remote_id = '1';
```


在脚本中使用远程 ID 和 MobiLink 用户名

MobiLink 用户可以标识一个人员，用于进行验证。而远程 ID 唯一地标识一个 MobiLink 远程数据库。

在许多同步脚本中，可以选择通过远程 ID（使用命名参数 s.remote_id）或者通过 MobiLink 用户名（使用 s.username）标识远程数据库。使用远程 ID 有一些好处，特别是在 UltraLite 中。

当部署在远程数据库和 MobiLink 用户之间具有一对一关系，且 MobiLink 用户名唯一地标识远程数据库时，您可以忽略远程 ID。这种情况下，MobiLink 事件脚本可引用 username 参数，此参数值即为用于进行验证的 MobiLink 用户名。

如果 MobiLink 用户要同步不同数据库中的数据，但每个远程数据库中的数据相同，则同步脚本可以引用 MobiLink 用户名。但如果 MobiLink 用户要同步不同数据库中不同的几组数据，则同步脚本应引用远程 ID。

在 UltraLite 数据库中，即使上一次上载的状态为未知，不同用户也可同步相同的数据库，这是因为 MobiLink 服务器通过远程 ID 跟踪同步进程。这种情况下，您不能再在下载脚本中引用 MobiLink 用户名来进行基于时间戳的下载，因为其他各用户的某些行可能会丢失且永远不能被下载。为防止出现这种情况，您需要在统一数据库中实现一个映射表，每个使用相同远程数据库的用户在表中都占一行。将基于当前同步的远程 ID 的映射表和统一表结合使用后，可以确保能下载所有用户的所有数据。

您也可以使用不同的脚本版本将不同的数据同步到不同的远程数据库。请参见“脚本版本”一节《MobiLink - 服务器管理》。

选择用户验证机制

用户验证是保护数据的安全系统的一部分。

MobiLink 为您提供了选择用户验证机制的功能。您不必使用单一的安装机制，MobiLink 允许您在安装过程中为不同的脚本版本选择不同的验证机制，以实现灵活性。

- **无 MobiLink 用户验证** 如果数据无需口令保护，可在安装中选择不使用任何用户验证。在这种情况下，MobiLink 用户名仍必须包含在 ml_user 表中，但 hashed_password 列为空值。
- **内部 MobiLink 用户验证** MobiLink 使用存储在 MobiLink 系统表 ml_user 中的用户名和口令执行验证。

  内部机制将在下面的章节中解释。
- **自定义验证** 可以使用 MobiLink 脚本 authenticate_user 来将内置 MobiLink 用户验证系统替换为自己的验证系统。例如，如果统一数据库管理系统允许，您可用数据库用户验证代替 MobiLink 系统。

  请参见“自定义用户验证”一节第 20 页。

有关 MobiLink 的其它安全相关功能及其相关产品的信息，请参见：

- “加密 MobiLink 客户端/服务器通信”一节 《SQL Anywhere 服务器 - 数据库管理》
- UltraLite 客户端： “保护 UltraLite 数据库”一节 《UltraLite - 数据库管理和参考》
- SQL Anywhere 客户端： “保护数据的安全” 《SQL Anywhere 服务器 - 数据库管理》

用户验证体系结构

MobiLink 用户验证系统基于用户名和口令来实现。您可以让 MobiLink 服务器使用内置的机制来校验用户名和口令，也可以选择实现自定义的用户验证机制。

在内置的验证系统中，用户名和口令都存储在统一数据库的 MobiLink 系统表 ml_user 中。口令以散列形式存储，因此 MobiLink 服务器以外的其它应用程序无法读取 ml_user 表，也无法重新构造出口令的初始形式。可以通过以下方法将用户名和口令添加到统一数据库中：使用 Sybase Central、使用 mluser 实用程序或在启动 MobiLink 服务器时指定 -zu+。

请参见“创建和注册 MobiLink 用户”一节第 10 页。

MobiLink 客户端在连接 MobiLink 服务器时会提供以下值：

- **用户名** MobiLink 用户名，必需信息。要进行同步，用户名必须存储在 ml_user 系统表中，否则您必须用 -zu+ 选项启动 MobiLink 服务器，以便将新用户添加到 ml_user 表中。
- **口令** MobiLink 口令。仅当用户未知或 MobiLink 系统表 ml_user 中相应的口令为空值时是可选信息。
- **新口令** 新的 MobiLink 口令，可选信息。MobiLink 用户可通过设置此值来更改他们的口令。

自定义验证

或者，您也可以选择使用自己的用户验证机制。

请参见“自定义用户验证”一节第 20 页。

验证过程

以下是对验证过程中所发生事件的顺序的说明。

1. 远程应用程序使用远程 ID、MobiLink 用户名以及口令和新口令（可选）来启动同步请求。MobiLink 服务器会启动一个新事务并触发 begin_connection_autocommit 事件和 begin_connection 事件。
2. MobiLink 会验证远程 ID 当前是否未在进行同步并将 authentication_status 预设为 4000。
3. 如果定义了 authenticate_user 脚本，则将发生以下情况：
   a. 如果用 SQL 编写 authenticate_user 脚本，则将以 authentication_status 预设值 4000、所提供的 MobiLink 用户名以及可选的口令和新口令来调用此脚本。

      如果用 Java 或 .NET 编写 authenticate_user 脚本并返回 SQL 语句，则将以 authentication_status 预设值 4000、所提供的 MobiLink 用户名以及可选的口令和新口令来调用此 SQL 语句。
   b. 如果 authenticate_user 脚本在执行过程中抛出异常或出现错误，同步进程将停止。

   authenticate_user 脚本或返回的 SQL 语句必须是对带有两到四个参数的存储过程的调用。authentication_status 预设值会作为第一个参数传递且可以由存储过程进行更新。第一个参数返回的值是 authenticate_user 脚本中的 authentication_status。
4. 如果存在 authenticate_user_hashed 脚本，将发生以下情况：
   a. 如果提供了口令，将为该口令计算散列值。如果提供了新口令，也将为其计算散列值。
   b. 将以 authentication_status 的当前值（如果 authenticate_user 脚本不存在则为预设 authentication_status，否则为 authenticate_user 脚本中返回的 authentication_status）和散列口令来调用 authenticate_user_hashed 脚本。此行为与第 3 步相同。第一个参数的返回值用作 authenticate_user_hashed 脚本的 authentication_status。
5. MobiLink 服务器会采用 authenticate_user 脚本和 authenticate_user_hashed 脚本（如果这两个脚本存在）所返回 auth_user 状态中的较大值，如果这两个脚本均不存在，则采用预设 authentication_status。
6. MobiLink 服务器在 ml_user 表中查询您提供的 MobiLink 用户名。
   a. 如果调用了自定义脚本 authenticate_user 或 authenticate_user_hashed，但您提供的 MobiLink 用户名未在 ml_user 表中而且 authentication_status 有效（1000 或 2000），则 MobiLink 用户名会被添加到 MobiLink 系统表 ml_user 中。如果 authentication_status 无效，将不会更新 ml_user，并且会发生错误。
   b. 如果未调用自定义脚本而且您提供的 MobiLink 用户名也不在 ml_user 表中，那么假如您用 -zu+ 选项启动 MobiLink 服务器，您所提供的 MobiLink 用户名会被添加到 ml_user 中。否则，将发生错误，authentication_status 会设置为无效。
   c. 如果调用了自定义脚本且您提供的 MobiLink 用户名在 ml_user 表中，则不会发生任何情况。
   d. 如果**未**调用自定义脚本，而您所提供的 MobiLink 用户名在 ml_user 表中，则将根据 ml_user 表中的值来检查口令。如果口令与 ml_user 表中该 MobiLink 用户的口令相匹配，authentication_status 会设置为有效。否则 authentication_status 会设置为无效。

7. 如果该 authentication_status 有效而且没有调用 authenticate_user 和 authenticate_user_hashed 脚本中的任何一个，那么假如您在 ml_user 表中为此 MobiLink 用户提供了新口令，则口令会被更改为您所提供的那个口令。
8. 如果定义了 authenticate_parameters 脚本且 authentication_status 有效（1000 或 2000），则将发生以下情况：
   a. 参数将传递给 authenticate_parameters 脚本。
   b. 如果 authenticate_parameters 脚本返回的 authentication_status 值大于当前的 authentication_status 值，则新的 authentication_status 值将覆盖旧值。
9. 如果 authentication_status 无效，将中止同步。
10. 如果定义了 modify_user 脚本，则将调用它，以用此脚本返回的新 MobiLink 用户名来替换您提供的 MobiLink 用户名。
11. 无论 authentication_status 如何，MobiLink 服务器总是会在 MobiLink 用户验证后提交事务。如果 authentication_status 有效（1000 或 2000），则同步继续。如果 authentication_status 无效，将中止同步。

自定义用户验证

您可以选择内置 MobiLink 机制以外的其它用户验证机制。以下是使用自定义用户验证机制的一些理由：

- 要包含与现有数据库用户验证模式或外部验证机制的集成。
- 要提供内置 MobiLink 机制所没有的自定义功能，如最小口令长度或口令到期时间。

有三种自定义验证工具：

- mlsrv11 -zu+ 选项
- authenticate_user 脚本或 authenticate_user_hashed 脚本
- authenticate_parameters 脚本

mlsrv11 -zu+ 选项可用来控制用户的自动添加。例如，指定 -zu+ 可以将所有无法识别的 MobiLink 用户名在它们首次进行同步时添加到 ml_user 表中。-zu+ 选项仅用于进行内置的 MobiLink 验证。

The authenticate_user、authenticate_user_hashed 和 authenticate_parameters 脚本都可以覆盖缺省的 MobiLink 用户验证机制。任何经过成功验证的用户都会被自动添加到 ml_user 表中。

可使用 authenticate_user 脚本创建用户 ID 和口令的自定义验证。如果此脚本存在，将执行此脚本，而不进行内部口令比较。该脚本必须返回错误代码以指示验证是成功还是失败。

随同 MobiLink 一起安装的还有几个 authenticate_user 事件的预定义脚本。这些都使您可以更轻松地使用 LDAP、POP3 和 IMAP 服务器进行验证。请参见“向外部服务器验证”一节第 21 页。

使用 authenticate_parameters 创建的自定义验证依赖于用户 ID 和口令之外的值。

另请参见

- “-zu 选项”一节 《MobiLink - 服务器管理》
- “authenticate_user 连接事件”一节 《MobiLink - 服务器管理》
- “authenticate_user_hashed 连接事件”一节 《MobiLink - 服务器管理》
- “authenticate_parameters 连接事件”一节 《MobiLink - 服务器管理》

Java 和 .NET 用户验证

用户验证是 Java 和 .NET 同步逻辑的自然应用，因为 Java 和 .NET 类允许访问计算环境中的其它用户名和口令源，例如应用程序服务器。

samples-dir\MobiLink\JavaAuthentication 目录中提供了一个简单的示例。*samples-dir\MobiLink\JavaAuthentication\CustEmpScripts.java* 中的示例代码实现了一个简单的用户验证系统。在第一次同步中有一个 MobiLink 用户名被添加到 login_added 表。在后续同步中有一行被添加到 login_audit 表。此例中，将用户 ID 添加到 login_added 表之前不进行测试。（有关 *samples-dir* 的信息，请参见“示例目录”一节 《SQL Anywhere 服务器 - 数据库管理》。）

有关解释用户验证的 .NET 示例，请参见“.NET 同步示例”一节 《MobiLink - 服务器管理》。

向外部服务器验证

预定义的 Java 同步脚本与 MobiLink 一起提供，它们使您可以更方便地使用 authenticate_user 事件向外部服务器进行验证。预定义脚本可用于以下验证服务器：

- 使用 JavaMail 1.2 API 的 POP3 或 IMAP 服务器
- 使用 Java 命名和目录接口 (JNDI) 的 LDAP 服务器

如何使用这些脚本取决于 MobiLink 用户名是否直接映射到外部验证系统中的用户 ID。

注意
您也可以使用 [验证] 选项卡，以 Sybase Central [模型] 模式设置向外部服务器的验证。请参见“MobiLink 模型” 《MobiLink - 入门》。

如果 MobiLink 用户名直接映射到用户 ID

MobiLink 用户名直接映射到验证系统中的有效用户 ID 是一种简单的情况，此时可以直接使用预定义脚本来响应 authenticate_user 连接事件。验证代码将根据存储在 ml_property 表中的属性对自身进行初始化。

♦ 在 authenticate_user 中直接使用预定义脚本

1. 将预定义的 Java 同步脚本添加到 ml_scripts MobiLink 系统表中。可以使用存储过程或在 Sybase Central 中完成此操作。
   - 要使用 ml_add_java_connection_script 存储过程，请运行以下命令：

     ```
     call ml_add_java_connection_script(
       'MyVersion',
       'authenticate_user',
       'ianywhere.ml.authentication.ServerType.authenticate' )
     ```

     其中 *MyVersion* 为脚本版本的名称，而 *ServerType* 为 **LDAP**、**POP3** 或 **IMAP**。
   - 要使用 Sybase Central 中的 [**添加连接脚本向导**]，请选择 [**authenticate_user**] 作为脚本类型，并在代码编辑器中输入以下内容：

     ```
     ianywhere.ml.authentication.ServerType.authenticate
     ```

     其中 *ServerType* 为 **LDAP**、**POP3** 或 **IMAP**。

     请参见“ml_add_java_connection_script 系统过程”一节 《MobiLink - 服务器管理》。
2. 为该验证服务器添加属性。

   对需要设置的每种属性使用 ml_add_property 存储过程：

   ```
   call ml_add_property(
     'ScriptVersion',
     'MyVersion',
     'property_name',
     'property_value' )
   ```

   其中，*MyVersion* 是脚本版本的名称，*property_name* 由验证服务器决定，*property_value* 是适用于您的应用程序的值。对想要设置的每种属性重复此调用。

请参见“外部验证程序属性”一节第 22 页和“ml_add_property 系统过程”一节 《MobiLink - 服务器管理》。

如果 MobiLink 用户名不直接映射到用户 ID

如果 MobiLink 用户名与您的用户 ID 不同，则必须间接调用代码，并且必须从 ml_user 值中抽取或映射用户 ID。可通过编写 Java 类来完成此操作。

请参见“使用 Java 语言编写同步脚本” 《MobiLink - 服务器管理》。

下面是一个简单的示例。本示例省去了 extractUserID 方法中的代码，因为该代码取决于 ml_user 值映射到 userid 的方式。所有操作都是用验证类的 "authenticate" 方法完成的。

```
package com.mycompany.mycode;

import ianywhere.ml.authentication.*;
import ianywhere.ml.script.*;

public class MLEvents
{
    private DBConnectionContext _context;
    private POP3 _pop3;

    public MLEvents( DBConnectionContext context )
    {
        _context = context;
        _pop3 = new POP3( context );
    }

    public void authenticateUser(
      InOutInteger status,
      String userID,
      String password,
      String newPassword )
    {
        String realUserID = extractUserID( userID );
        _pop3.authenticate( status, realUserID, password, newPassword );
    }

    private String extractUserID( String userID )
    {
        // code here to map ml_user to a "real" POP3 user
    }
}
```

在本示例中，需要用 DBConnectContext 对象初始化 POP3 对象，以便 POP3 对象可以找到其初始化属性。如果不用这种方法进行初始化，则必须通过代码设置属性。例如，

```
POP3 pop3 = new POP3();
pop3.setServerName( "smtp.sybase.com" );
pop3.setServerPort( 25 );
```

这适用于所有的验证类，尽管属性会依类的不同而不同。

外部验证程序属性

MobiLink 会在任何可能的情况下（特别是 LDAP 情况下）提供合理的缺省值。尽管不同情况可以设置的属性会有所不同，但以下是一些基本的属性。

POP3 验证程序

mail.pop3.host	服务器的主机名
mail.pop3.port	端口号（在使用缺省端口 110 时可省略）

请参见 http://java.sun.com/products/javamail/javadocs/com/sun/mail/pop3/package-summary.html。

IMAP 验证程序

mail.imap.host	服务器的主机名
mail.imap.port	端口号（在使用缺省端口 143 时可省略）

请参见 http://java.sun.com/products/javamail/javadocs/com/sun/mail/imap/package-summary.html。

LDAP 验证程序

java.naming.provider.url	LDAP 服务器的 URL，如 `ldap://ops-yourLocation/dn=sybase,dn=com`

有关详细信息，请参见 JNDI 文档。

MobiLink 客户端实用程序

目录

MobiLink 客户端实用程序简介

有两个 MobiLink 客户端实用程序：

- “ActiveSync 提供程序的安装实用程序 (mlasinst)”一节第 27 页
- “MobiLink 文件传输实用程序 (mlfiletransfer)”一节第 29 页

此外，请参见：

- UltraLite 实用程序： “UltraLite 实用程序” 《UltraLite - 数据库管理和参考》
- MobiLink 服务器实用程序： “MobiLink 实用程序” 《MobiLink - 服务器管理》
- 其它 SQL Anywhere 实用程序： “数据库管理实用程序” 《SQL Anywhere 服务器 - 数据库管理》

ActiveSync 提供程序的安装实用程序 (mlasinst)

安装 ActiveSync（在 Windows Vista 上称作 Windows 移动设备中心）的 MobiLink 提供程序，或者在 Windows Mobile 设备上注册并安装 UltraLite 应用程序。

语法

mlasinst [*options*] [[*src*] *dst name class* [*args*]]

选项	说明
-d	初始禁用应用程序。
-k *path*	指定台式机提供程序 *mlasdesk.dll* 的位置。 缺省情况下，此文件位于 *install-dir\bin32* 下。 最终用户（一般没有 SQL Anywhere 的完全安装）可能需要在安装 MobiLink ActiveSync 提供程序时指定 -k。
-l *filename*	指定活动日志文件的名称。
-n	注册应用程序而不是将其复制到设备上。 除了安装 MobiLink ActiveSync 提供程序外，此选项还将注册应用程序但不会将其复制到设备。如果应用程序包含多个文件（例如，使用 UltraLite 运行时库 DLL 而不是静态库编译该应用程序时），或者如果有其它可将应用程序复制到设备的方法，则可使用此选项。
-t *n*	以秒为单位，指定台式机提供程序在超时前应等待来自客户端响应的时间（缺省值为 30 秒）。
-u	卸载 ActiveSync 的 MobiLink 提供程序。 此选项注销已注册与 MobiLink ActiveSync 提供程序一起使用的所有应用程序，然后卸载 MobiLink ActiveSync 提供程序。此操作并不从台式计算机或设备上删除文件。如果设备未连接到台式机，则报告错误。
-v *path*	指定设备提供程序 *mlasdev.dll* 的位置。缺省情况下，可在特定于平台的目录 *install-dir\CE* 中查找此文件。 最终用户（一般没有 SQL Anywhere 的完全安装）可能需要在安装 MobiLink ActiveSync 提供程序时指定 -v。

其它参数	说明
src	指定将应用程序复制到设备上时所需的源文件名和路径。仅当注册应用程序并将其复制到设备中时，才会提供此参数。如果使用 -n 选项，则不会提供此参数。

其它参数	说明
dst	指定位于设备上的应用程序的目标文件名和路径。
name	指定 ActiveSync 引用此应用程序时使用的名称。
class	指定应用程序的注册 Windows 类名。
args	指定 ActiveSync 启动应用程序时传送到应用程序的命令行参数。

注释

该实用程序为 ActiveSync 安装 MobiLink 提供程序。提供程序中包含一个在台式机上运行的组件 (*mlasdesk.dll*) 和一个部署到 Windows Mobile 设备的组件 (*mlasdev.dll*)。Mlasinst 实用程序创建一个指向桌面提供程序当前位置的注册表条目，并将该设备提供程序复制到该设备。

如果提供了其它参数，*mlasinst* 实用程序还可用于将 UltraLite 应用程序注册并安装到 Windows Mobile 设备上。或者，您可以使用 ActiveSync 软件注册和安装 UltraLite 应用程序。

受许可要求的限制，您需要为最终用户提供此应用程序以及桌面组件和设备组件，以便他们可以准备要与 ActiveSync 一同使用的应用程序的副本。

必须连接到远程设备，以便安装 ActiveSync 提供程序。

有关使用 ActiveSync 提供程序的安装实用程序的完整说明，请参见：

- SQL Anywhere： “安装 ActiveSync 的 MobiLink 提供程序”一节第 111 页
- UltraLite: “Windows Mobile 上的 ActiveSync”一节 《UltraLite - 数据库管理和参考》

示例

以下命令使用缺省参数为 ActiveSync 安装 MobiLink 提供程序。该命令不注册应用程序。必须将设备连接到桌面，才能安装成功。

```
mlasinst
```

以下命令为 ActiveSync 卸载 MobiLink 提供程序。必须将设备连接到桌面，才能卸载成功：

```
mlasinst -u
```

以下命令可为 ActiveSync 安装 MobiLink 提供程序（如果尚未安装），并注册应用程序 *myapp.exe*。它还会将 *c:\My Files\myapp.exe* 文件复制到设备上的 *\Program Files\myapp.exe*。-p -x 参数是 ActiveSync 启动 *myapp.exe* 时使用的命令行选项。必须将此命令输入到一行中：

```
mlasinst "C:\My Files\myapp.exe" "\Program Files\myapp.exe"
   "My Application" MYAPP -p -x
```

另请参见

- “使用 ActiveSync 同步”一节第 110 页
- “UltraLite 同步参数和网络协议选项” 《UltraLite - 数据库管理和参考》

MobiLink 文件传输实用程序 (mlfiletransfer)

通过 MobiLink 下载文件。

语法

mlfiletransfer [*options*] *file*

选项	说明
-ap *param1*, ...	MobiLink 验证参数。请参见“验证参数”一节 《MobiLink - 服务器管理》。
-dp *path*	下载文件将要存储于其中的本地路径。缺省情况下，将下载文件存储在根目录中（在 Windows Mobile 上）或当前目录中（在其它平台上）。
-df *filename*	已下载文件的本地名称。如果要让文件在客户端和服务器上使用不同的名称，则使用此选项。缺省情况下，使用服务器名。
-f	强制下载，即使本地文件已是最新的。放弃以前的所有部分下载。
-g	显示下载进度。
-p *password*	MobiLink 用户名的口令。
-q	安静模式。不显示消息。
-r	启用下载恢复。如果进行了此设置，则实用程序将恢复因通信错误或用户将其取消而中断的先前的部分下载。当服务器上的文件比部分文件新时，将放弃部分文件。-f 选项将覆盖此选项。
-u *username*	MobiLink 用户名。该选项是必需的。
-v *version*	脚本版本。该选项是必需的。
-x *protocol*(*options*)	*protocol* 可以是 **tcpip**、**tls**、**http** 或 **https** 之一。该选项是必需的。 可使用的协议选项取决于协议。有关每个协议的选项列表，请参见“MobiLink 客户端网络协议选项汇总”一节 第 33 页。

选项	说明
file	服务器上指定要传输的文件。不包括路径，因为文件的位置由 mlsrv11 -ftr 选项（在启动 MobiLink 服务器时必须使用该选项）确定。MobiLink 将在 -ftr 目录的用户名子目录中查找该文件；如果在该处未找到，则会在 -ftr 目录中查找。如果文件不在这两个位置，则 MobiLink 将生成错误。 请参见“-ftr 选项”一节 《MobiLink - 服务器管理》。

注释

在以下情况下，此实用程序可用于下载文件：首次创建远程数据库，需要对远程设备上的软件进行升级，等等。

要使用此实用程序，必须使用 -ftr 选项启动 MobiLink 服务器。-ftr 选项为将要传输的文件创建一个根目录，并为每个注册的 MobiLink 用户创建一个子目录。

UltraLite 用户也可在 UltraLite 运行时使用 MLFileTransfer 方法。请参见“使用 MobiLink 文件传输”一节 《UltraLite - 数据库管理和参考》。

另请参见

- “-ftr 选项”一节 《MobiLink - 服务器管理》
- “authenticate_file_transfer 连接事件”一节 《MobiLink - 服务器管理》

示例

以下命令会将 MobiLink 服务器连接到 CustDB 示例数据库。`-ftr %SystemRoot%\system32` 选项指示 MobiLink 服务器针对请求的文件启动对 *Windows\system32* 目录的监视。在此示例中，MobiLink 服务器首先在 *C:\Windows\system32\mobilink-username* 目录中查找此文件。如果文件不存在，它会在 *C:\Windows\system32* 目录中查找。一般来说，您不会希望 MobiLink 服务器监视 *Windows\system32* 文件夹的文件。此示例使用 *Windows\system32* 目录，以便可以传输位于此处的 Notepad 实用程序。

```
mlsrv11 -c "dsn=SQL Anywhere 11 CustDB" -zu+ -ftr %SystemRoot%\system32
```

以下命令将运行 mlfiletransfer 实用程序。它会使 MobiLink 服务器将 *notepad.exe* 下载到您的本地目录中。

```
MLFileTransfer -u 1 -v "custdb 10.0" -x tcpip notepad.exe
```


MobiLink 客户端网络协议选项

目录

MobiLink 客户端网络协议选项汇总

本节介绍了将 MobiLink 客户端连接到 MobiLink 服务器时可以使用的一些网络协议选项。有几个 MobiLink 客户端实用程序使用 MobiLink 客户端网络协议选项：

若要在使用以下组件时使用客户端网络协议选项	请参见……
dbmlsync	“CommunicationAddress (adr) 扩展选项”一节第 179 页
UltraLite	“Stream Parameters 同步参数”一节 《UltraLite - 数据库管理和参考》或“UltraLite 同步实用程序 (ulsync)”一节 《UltraLite - 数据库管理和参考》中的 -x 选项
UltraLiteJ	“UltraLiteJ 同步流的网络协议选项”一节 《UltraLiteJ》
重定向器	“配置重定向器属性（适用于支持服务器组的重定向器）”一节 《MobiLink - 服务器管理》或“配置重定向器属性（适用于不支持服务器组的重定向器）”一节 《MobiLink - 服务器管理》中的 ML 指令
MobiLink 监控器	“启动 MobiLink 监控器”一节 《MobiLink - 服务器管理》
MobiLink 文件传输	“MobiLink 文件传输实用程序 (mlfiletransfer)”一节第 29 页
MobiLink 监听器	“用于 Windows 设备的监听器实用程序”一节 《MobiLink - 服务器启动的同步》中的 -x
QAnywhere 代理	“-x 选项”一节 《QAnywhere》

所选的网络协议必须与客户端使用的同步协议相匹配。有关如何为 MobiLink 服务器设置连接选项的信息，请参见“-x 选项”一节 《MobiLink - 服务器管理》。

协议选项

- **TCP/IP 协议选项** 如果指定 **tcpip** 选项，则还可以选择指定以下协议选项：

TCP/IP 协议选项	有关详细信息，请参见……
buffer_size=*bytes*	“buffer_size”一节第 38 页
client_port=*nnnnn*[-*mmmmm*]	“client_port”一节第 44 页
e2ee_type={**rsa**|**ecc**}	“e2ee_type”一节第 47 页

TCP/IP 协议选项	有关详细信息，请参见……
e2ee_public_key=*file*	“e2ee_public_key”一节第 48 页
compression={**zlib**|**none**}	“Compression”一节第 45 页
host=*hostname*	“host”一节第 51 页
network_leave_open={**off**|**on**}	“network_leave_open”一节第 57 页
network_name=*name*	“network_name”一节第 58 页
port=*portnumber*	“port”一节第 61 页
timeout=*seconds*	“timeout”一节第 65 页
zlib_download_window_size=*window-bits*	“zlib_download_window_size”一节第 73 页
zlib_upload_window_size=*window-bits*	“zlib_download_window_size”一节第 74 页

- **具有安全性的 TCP/IP 协议** 如果指定 **tls** 选项（该协议是具有 TLS 安全性的 TCP/IP 协议），则可选择指定以下协议选项：

TLS 协议选项	有关详细信息，请参见...
buffer_size=*bytes*	“buffer_size”一节第 38 页
certificate_company=*company_name*	“certificate_company”一节第 39 页
certificate_name=*name*	“certificate_name”一节第 41 页
certificate_unit=*company_unit*	“certificate_unit”一节第 43 页
client_port=*nnnnn*[-*mmmmm*]	“client_port”一节第 44 页
compression={**zlib**|**none**}	“Compression”一节第 45 页
e2ee_type={**rsa**|**ecc**}	“e2ee_type”一节第 47 页
e2ee_public_key=*file*	“e2ee_public_key”一节第 48 页
fips={**y**|**n**}	“fips”一节第 49 页
host=*hostname*	“host”一节第 51 页
network_leave_open={**off**|**on**}	“network_leave_open”一节第 57 页
network_name=*name*	“network_name”一节第 58 页

TLS 协议选项	有关详细信息，请参见...
port=*portnumber*	“port”一节第 61 页
timeout=*seconds*	“timeout”一节第 65 页
tls_type={**rsa**|**ecc**}	“tls_type”一节第 66 页
trusted_certificates=*filename*	“trusted_certificates”一节第 68 页
zlib_download_window_size=*window-bits*	“zlib_download_window_size”一节第 73 页
zlib_upload_window_size=*window-bits*	“zlib_download_window_size”一节第 74 页

- **HTTP 协议** 如果指定 **http** 选项，则还可以选择指定以下协议选项：

HTTP 协议选项	有关详细信息，请参见...
buffer_size=*number*	“buffer_size”一节第 38 页
client_port=*nnnnn*[-*mmmmm*]	“client_port”一节第 44 页
compression={**zlib**|**none**}	“Compression”一节第 45 页
custom_header=*header*	“custom_header”一节第 46 页
e2ee_type={**rsa**|**ecc**}	“e2ee_type”一节第 47 页
e2ee_public_key=*file*	“e2ee_public_key”一节第 48 页
http_password=*password*	“http_password”一节第 52 页
http_proxy_password=*password*	“http_proxy_password”一节第 53 页
http_proxy_userid=*userid*	“http_proxy_userid”一节第 54 页
http_userid=*userid*	“http_userid”一节第 55 页
host=*hostname*	“host”一节第 51 页
network_leave_open={**off**|**on**}	“network_leave_open”一节第 57 页
network_name=*name*	“network_name”一节第 58 页
persistent={**off**|**on**}	“persistent”一节第 60 页
port=*portnumber*	“port”一节第 61 页
proxy_host=*proxy-hostname-or-ip*	“proxy_host”一节第 62 页

HTTP 协议选项	有关详细信息，请参见...
proxy_port=*proxy-portnumber*	“proxy_port”一节第 63 页
set_cookie=*cookie-name*=*cookie-value*	“set_cookie”一节第 64 页
timeout=*seconds*	“timeout”一节第 65 页
url_suffix=*suffix*	“url_suffix”一节第 70 页
version=*HTTP-version-number*	“version”一节第 72 页
zlib_download_window_size=*window-bits*	“zlib_download_window_size”一节第 73 页
zlib_upload_window_size=*window-bits*	“zlib_download_window_size”一节第 74 页

- **HTTPS 协议** 如果指定 **https** 选项（该协议是具有 RSA 加密的 HTTP 协议），则可选择指定以下协议选项：

HTTPS 协议选项	有关详细信息，请参见...
buffer_size=*number*	“buffer_size”一节第 38 页
certificate_company=*company_name*	“certificate_company”一节第 39 页
certificate_name=*name*	“certificate_name”一节第 41 页
certificate_unit=*company_unit*	“certificate_unit”一节第 43 页
client_port=*nnnnn*[-*mmmmm*]	“client_port”一节第 44 页
compression={**zlib**|**none**}	“Compression”一节第 45 页
custom_header=*header*	“custom_header”一节第 46 页
e2ee_type={**rsa**|**ecc**}	“e2ee_type”一节第 47 页
e2ee_public_key=*file*	“e2ee_public_key”一节第 48 页
fips={**y**|**n**}	“fips”一节第 49 页
host=*hostname*	“host”一节第 51 页
http_password=*password*	“http_password”一节第 52 页
http_proxy_password=*password*	“http_proxy_password”一节第 53 页

HTTPS 协议选项	有关详细信息，请参见...
http_proxy_userid=*userid*	“http_proxy_userid”一节 第 54 页
http_userid=*userid*	“http_userid”一节第 55 页
network_leave_open={**off**|**on**}	“network_leave_open”一节 第 57 页
network_name=*name*	“network_name”一节第 58 页
persistent={**off**|**on**}	“persistent”一节第 60 页
port=*portnumber*	“port”一节第 61 页
proxy_host=*proxy-hostname-or-ip*	“proxy_host”一节第 62 页
proxy_port=*proxy-portnumber*	“proxy_port”一节第 63 页
set_cookie=*cookie-name*=*cookie-value*	“set_cookie”一节第 64 页
timeout=*seconds*	“timeout”一节第 65 页
tls_type={**rsa**|**ecc**}	“tls_type”一节第 66 页
trusted_certificates=*filename*	“trusted_certificates”一节 第 68 页
url_suffix=*suffix*	“url_suffix”一节第 70 页
version=*HTTP-version-number*	“version”一节第 72 页
zlib_download_window_size=*window-size*	“zlib_download_window_size”一节第 73 页
zlib_upload_window_size=*window-bits*	“zlib_download_window_size”一节第 74 页

需要单独授予许可的组成部分
ECC 加密和 FIPS 认证的加密需要单独的许可。所有高度加密技术受出口法规约束。

请参见“单独授权的组件”一节 《SQL Anywhere 11 - 简介》。

buffer_size

指定在向网络写入数据前要缓冲的最多字节数。对于 HTTP 和 HTTPS，这将转换为 HTTP 请求主体最大大小。

语法

buffer_size=*bytes*

协议

- TCPIP、TLS、HTTP、HTTPS

支持附注

- 不可以在重定向器的 ML 指令中使用。

缺省值

- Palm - 4K
- CE - 16K
- 所有其它平台 -64K

注释

通常，对于 HTTP 和 HTTPS 来说，缓冲区大小越大，HTTP 请求/响应循环的次数就越少，但需要的内存也越多。

对于 TCPIP 和 TLS，同样是缓冲区大小越大，执行的速度就越快，但需要更多的内存；然而性能差异就没有使用 HTTP 时那样显著。

单位均按字节计。指定 K 代表千字节；M 代表兆字节或 G 代表千兆字节。

最大值为 1G。

此选项用于控制客户端请求的大小，而与 MobiLink 响应的大小无关。

有关如何使用 dbmlsync 设置网络协议选项的信息，请参见“CommunicationAddress (adr) 扩展选项”一节第 179 页。

有关如何设置 UltraLite 网络协议选项的信息，请参见“UltraLite 同步流的网络协议选项”一节《UltraLite - 数据库管理和参考》。

示例

以下示例将最大字节数设置为 32K。

在 SQL Anywhere 客户端上，实现如下：

```
dbmlsync -e "adr=buffer_size=32K"
```

对于以嵌入式 SQL 或 C++ 编写的 UltraLite 应用程序，实现如下：

```
synch_info.stream_parms = TEXT("buffer_size=32K");
```


certificate_company

如果指定，仅当证书的 [组织] 字段与此值相匹配时，应用程序才接受服务器证书。

需要单独授予许可的组成部分
ECC 加密和 FIPS 认证的加密需要单独的许可。所有高度加密技术受出口法规约束。

请参见“单独授权的组件”一节 《SQL Anywhere 11 - 简介》。

语法

certificate_company=*organization*

协议

- TLS、HTTPS

缺省值

无

注释

MobiLink 客户端信任所有由证书颁发机构签名的证书，因此它们也可能信任同一证书颁发机构发放给其它公司的证书。如果没有方法予以区分，您的客户端可能会将竞争对手的 MobiLink 服务器误当成您自己的服务器，并无意将敏感信息发送给该服务器。此选项指定了更深一层的校验，即证书的标识部分中的 [组织] 字段也要与您指定的值匹配。

有关如何设置 dbmlsync 网络协议选项的信息，请参见“CommunicationAddress (adr) 扩展选项”一节第 179 页。

有关如何设置 UltraLite 网络协议选项的信息，请参见“UltraLite 同步流的网络协议选项”一节 《UltraLite - 数据库管理和参考》。

另请参见

- “加密 MobiLink 客户端/服务器通信”一节 《SQL Anywhere 服务器 - 数据库管理》
- “校验证书字段”一节 《SQL Anywhere 服务器 - 数据库管理》
- “-x 选项”一节 《MobiLink - 服务器管理》
- “trusted_certificates”一节第 68 页
- “certificate_name”一节第 41 页
- “certificate_unit”一节第 43 页

示例

下面的示例指示 SQL Anywhere 客户端检查所有三个标识字段并只接受指定的值。此示例将校验所有字段。您也可以改为仅校验一个或两个字段。

例如，如果您有 SQL Anywhere 客户端，则可以在预订中如下设置证书校验：

```
CREATE SYNCHRONIZATION SUBSCRIPTION
FOR 'user01'
TO test_pub
ADDRESS 'port=3333;
```

```
  trusted_certificates=certicom.crt;
  certificate_company=Sybase, Inc.;
  certificate_unit=iAnywhere;certificate_name=sample'
```

对于 C 或 C++ 语言编写的嵌入式 SQL UltraLite 应用程序，可按如下方式设置证书校验，假设条件是，受信任证书在创建数据库时已安装在数据库中：

```
ul_synch_info info;
info.stream = "tls";
info.stream_parms = UL_TEXT("port=9999;")
   UL_TEXT ( "certificate_company=Sybase, Inc.;" )
   UL_TEXT ( "certificate_unit=iAnywhere;" )
   UL_TEXT ( "certificate_name=sample;" );
...
ULSynchronize( &info );
```


certificate_name

如果指定，仅当证书的 [公用名] 字段与此值相匹配时，应用程序才接受服务器证书。

需要单独授予许可的组成部分
ECC 加密和 FIPS 认证的加密需要单独的许可。所有高度加密技术受出口法规约束。

请参见“单独授权的组件”一节 《SQL Anywhere 11 - 简介》。

语法

certificate_name=*common-name*

协议

- TLS、HTTPS

缺省值

无

注释

有关如何设置 dbmlsync 网络协议选项的信息，请参见“CommunicationAddress (adr) 扩展选项”一节第 179 页。

有关如何设置 UltraLite 网络协议选项的信息，请参见“UltraLite 同步流的网络协议选项”一节 《UltraLite - 数据库管理和参考》。

另请参见

- “加密 MobiLink 客户端/服务器通信”一节 《SQL Anywhere 服务器 - 数据库管理》
- “校验证书字段”一节 《SQL Anywhere 服务器 - 数据库管理》
- “-x 选项”一节 《MobiLink - 服务器管理》
- “trusted_certificates”一节第 68 页
- “certificate_company”一节第 39 页
- “certificate_unit”一节第 43 页

示例

以下示例为某个 HTTPS 协议设置了 RSA 加密。这需要在服务器和客户端上进行设置。每个命令都必须在一行中写入。

在服务器上，实现如下：

```
mlsrv11
   -c "dsn=SQL Anywhere 11 Demo;uid=DBA;pwd=sql"
   -x https(
     port=9999;
     identity=c:\sa10\bin32\rsaserver.id;
     identity_password=test)
```

在 SQL Anywhere 客户端上，实现如下：

```
dbmlsync
   -c "dsn=mydb;uid=DBA;pwd=sql"
   -e "ctp=https;
       adr='port=9999;
         trusted_certificates=c:\sa10\bin32\rsaroot.crt;
         certificate_name=RSA Server'"
```

在 UltraLite 客户端上，实现如下：

```
info.stream = "https";
info.stream_parms = TEXT(
  "port=9999;
   trusted_certificates=\sa10\bin32\rsaroot.crt;
   certificate_name=RSA Server");
```

certificate_unit

如果指定，仅当证书的 [组织单位] 字段与此值相匹配时，应用程序才接受服务器证书。

需要单独授予许可的组成部分
ECC 加密和 FIPS 认证的加密需要单独的许可。所有高度加密技术受出口法规约束。

请参见“单独授权的组件”一节 《SQL Anywhere 11 - 简介》。

语法

certificate_unit=*organization-unit*

协议

- TLS、HTTPS

缺省值

无

注释

有关如何设置 dbmlsync 网络协议选项的信息，请参见“CommunicationAddress (adr) 扩展选项”一节第 179 页。

有关如何设置 UltraLite 网络协议选项的信息，请参见“UltraLite 同步流的网络协议选项”一节 《UltraLite - 数据库管理和参考》。

另请参见

- “加密 MobiLink 客户端/服务器通信”一节 《SQL Anywhere 服务器 - 数据库管理》
- “校验证书字段”一节 《SQL Anywhere 服务器 - 数据库管理》
- “-x 选项”一节 《MobiLink - 服务器管理》
- “trusted_certificates”一节第 68 页
- “certificate_company”一节第 39 页
- “certificate_name”一节第 41 页

示例

有关安全性的示例，请参见“certificate_name”一节第 41 页和“trusted_certificates”一节第 68 页。

client_port

指定用于通信的客户端端口范围。

语法

client_port=*nnnnn*[**-***mmmmm*]

协议

- TCPIP、TLS、HTTP、HTTPS

支持附注

- 不可以在重定向器的 ML 指令中使用。

缺省值

无

注释

指定一个下限值和一个上限值以创建可用端口号的范围。要让客户端只使用特定的端口号，请为 *nnnnn* 和 *mmmmm* 指定相同的数值。如果仅指定一个值，则在端口总数为 101 时，该范围的结束值比初始值大 100。

此选项对于防火墙内那些与防火墙外某 MobiLink 服务器进行通信的客户端十分有用。

有关如何设置 dbmlsync 网络协议选项的信息，请参见“CommunicationAddress (adr) 扩展选项”一节第 179 页。

有关如何设置 UltraLite 网络协议选项的信息，请参见“UltraLite 同步流的网络协议选项”一节《UltraLite - 数据库管理和参考》。

示例

以下示例将允许的客户端端口的端口范围设置为 10000。

在 SQL Anywhere 客户端上，实现如下：

```
dbmlsync -e "adr=client_port=10000-19999"
```

对于以嵌入式 SQL 或 C++ 编写的 UltraLite 应用程序，实现如下：

```
synch_info.stream_parms = TEXT("client_port=10000-19999");
```


Compression

打开或关闭 MobiLink 服务器与 MobiLink 客户端之间的同步流压缩。

语法

compression= { zlib | none }

协议

- TCPIP、TLS、HTTP、HTTPS

支持附注

- 在 Palm OS 或 Symbian 上不受支持。
- 不可以在重定向器的 ML 指令中使用。

缺省值

对于 UltraLite，缺省情况下压缩是关闭的。

对于 dbmlsync，缺省情况下使用 zlib 压缩。

> 在 SQL Anywhere 客户端中，如果关闭压缩，则对数据完全进行非模糊化处理；如果需要考虑安全性，则将对流进行加密。
>
> 请参见“传送层安全”《SQL Anywhere 服务器 - 数据库管理》。

注释

使用 zlib 压缩时，可使用 zlib_download_window_size 选项和 zlib_upload_window_size 选项配置上载和下载压缩。使用这些选项，还可关闭上载或下载的压缩。

要在 UltraLite 中使用 zlib 压缩，必须部署 *mlczlib10.dll*，并且（仅限 C++）应用程序必须调用 `ULEnableZlibSyncCompression( sqlca )`。

另请参见

- “zlib_download_window_size”一节第 73 页
- “zlib_download_window_size”一节第 74 页

示例

以下选项只为上载设置压缩，并将上载窗口大小设置为 9：

```
"compression=zlib;zlib_download_window_size=0;zlib_upload_window_size=9"
```

custom_header

指定一个自定义 HTTP 标头。

语法

custom_header=*header*

HTTP 标头的格式为 *header-name*: *header-value*。

协议

- HTTP、HTTPS

支持附注

- 不可以在重定向器的 ML 指令中使用。

缺省值

无

注释

指定了自定义 HTTP 标头后，客户端会在发送的每个 HTTP 请求中都包含这些标头。要指定多个自定义标头，可多次使用 custom_header，并使用分号 (;) 作为分隔符。例如：
custom_header=*header1*:*value1*; **customer_header=***header2*:*value2*

自定义标头对于同步客户端与需要自定义标头的第三方工具进行交互十分有用。

有关如何设置 dbmlsync 网络协议选项的信息，请参见“CommunicationAddress (adr) 扩展选项”一节第 179 页。

有关如何设置 UltraLite 网络协议选项的信息，请参见“UltraLite 同步流的网络协议选项”一节《UltraLite - 数据库管理和参考》。

示例

有些 HTTP 代理要求所有请求都包含特殊标头。下面的示例在某个嵌入式 SQL 或 C++ UltraLite 应用程序中将一个名为 MyProxyHdr 的自定义 HTTP 标头设置为值 ProxyUser：

```
info.stream = "http";
info.stream_parms = TEXT(
  "host=www.myhost.com;proxy_host=www.myproxy.com;
  custom_header=MyProxyHdr:ProxyUser");
```


e2ee_type

指定要用于端对端加密的密钥交换的非对称算法。

语法

e2ee_type= { **rsa** | **ecc** }

协议

TCPIP、TLS、HTTP、HTTPS

缺省值

RSA

注释

必须为 **rsa** 或 **ecc**，并且必须与服务器上指定的值相匹配。

另请参见

- “e2ee_public_key”一节第 48 页
- “-x 选项”一节 《MobiLink - 服务器管理》
- “密钥对生成器实用程序 (createkey)”一节 《SQL Anywhere 服务器 - 数据库管理》

示例

以下示例显示 UltraLite 客户端的端对端加密：

```
info.stream = "https";
info.stream_parms =
"tls_type=rsa;trusted_certificates=rsaroot.crt;e2ee_type=rsa;e2ee_public_key=
rsapublic.pem"
```


e2ee_public_key

指定包含服务器 PEM 编码的公共密钥（用于端对端加密）的文件。

语法

e2ee_public_key=*file*

协议

TCPIP、TLS、HTTP、HTTPS

缺省值

无

注释

密钥类型必须与 e2ee_type 参数中指定的类型相匹配。

若要使端对端加密生效，则需要使用此选项。

端对端加密也可与 TLS/HTTPS 协议选项 fips 一起使用。使用 ECC 时不支持此选项。请参见“fips”一节第 49 页。

另请参见

- “e2ee_type”一节第 47 页
- “-x 选项”一节 《MobiLink - 服务器管理》
- “密钥对生成器实用程序 (createkey)”一节 《SQL Anywhere 服务器 - 数据库管理》

示例

以下示例显示 UltraLite 客户端的端对端加密：

```
info.stream = "https";
info.stream_parms =
"tls_type=rsa;trusted_certificates=rsaroot.crt;e2ee_type=rsa;e2ee_public_key=
rsapublic.pem"
```


fips

对 TLS 加密和端对端加密使用 FIPS 认可的加密实现。

需要单独授予许可的组成部分
ECC 加密和 FIPS 认证的加密需要单独的许可。所有高度加密技术受出口法规约束。

请参见“单独授权的组件”一节 《SQL Anywhere 11 - 简介》。

语法

fips={ y | n }

协议

HTTPS、TLS

缺省值

否

注释

只有在 RSA 加密中才支持 FIPS。

非 FIPS 客户端可连接到 FIPS 服务器，反之亦然。

此选项可以与端对端加密一起使用。如果 fips 设置为 **y**，则 MobiLink 客户端使用 RSA 和 AES 的经 FIPS 140-2 认证的实现。使用 ECC 时不支持此选项。请参见“e2ee_type”一节第 47 页和“e2ee_public_key”一节第 48 页。

有关如何使用 dbmlsync 设置网络协议选项的信息，请参见“CommunicationAddress (adr) 扩展选项”一节第 179 页。

另请参见

- “tls_type”一节第 66 页
- “e2ee_type”一节第 47 页

示例

以下示例为一个 TCP/IP 协议设置了 FIPS 认可的 RSA 加密。这需要在服务器和客户端上进行设置。每个命令都必须在一行中写入。

在服务器上，实现如下：

```
mlsrv11
  -c "dsn=SQL Anywhere 11 Demo;uid=DBA;pwd=sql"
  -x tls(
    port=9999;
    tls_type=rsa;
    fips=y;
    identity=c:\sa10\bin32\rsaserver.id;
    identity_password=test )
```

在 SQL Anywhere 客户端上，实现如下：

```
dbmlsync -e
   "CommunicationType=tls;
    CommunicationAddress=
       'tls_type=rsa;
        fips=y;
        trusted_certificates=\rsaroot.crt;
        certificate_name=RSA Server'"
```

对于以 C 或 C++ 语言编写的嵌入式 SQL UltraLite 应用程序，实现如下：

```
info.stream = "tls";
info.stream_parms = TEXT(
  "tls_type=rsa;
   fips=y;
   trusted_certificates=\rsaroot.crt;
   certificate_name=RSA Server");
```


host

为运行 MobiLink 服务器的计算机或者运行 Web 服务器的计算机（如果正通过 Web 服务器进行同步）指定主机名或 IP 地址。

语法

host=*hostname-or-ip*

协议

- TCPIP、TLS、HTTP、HTTPS

缺省值

- Windows Mobile - 缺省值是与设备建立了 ActiveSync 合作关系的台式计算机的 IP 地址。
- 所有其它设备 - 缺省值为 **localhost**。

注释

在 Windows Mobile 上不要使用 localhost，该值是指远程设备本身。缺省值允许 Windows Mobile 设备连接到与其建立了 ActiveSync 合作关系的台式计算机上的 MobiLink 服务器。

对于 Palm 计算平台，缺省值 localhost 指的是设备本身。应提供一个要连接到台式计算机的显式主机名或 IP 地址。

有关如何设置 dbmlsync 网络协议选项的信息，请参见“CommunicationAddress (adr) 扩展选项”一节第 179 页。

有关如何设置 UltraLite 网络协议选项的信息，请参见“UltraLite 同步流的网络协议选项”一节《UltraLite - 数据库管理和参考》。

示例

在以下示例中，客户端在端口 1234 上连接到名为 myhost 的计算机。

在 SQL Anywhere 客户端上，实现如下：

```
dbmlsync -e "adr='host=myhost;port=1234'"
```

对于以嵌入式 SQL 或 C++ 编写的 UltraLite 应用程序，实现如下：

```
synch_info.stream_parms = TEXT("host=myhost;port=1234");
```


http_password

使用 RFC 2617 基本或摘要式验证向第三方 HTTP 服务器和网关验证。

语法

http_password=*password*

协议

- HTTP、HTTPS

支持附注

- 不可以在重定向器的 ML 指令中使用。

缺省值

无

注释

此功能支持 RFC 2617 中所述的基本验证和摘要式验证。

使用基本验证时，口令以明文形式包含在 HTTP 标头中；但可以使用 HTTPS 加密标头并保护该口令。使用摘要式验证时，标头不以明文形式发送，而是采取散列码形式。

必须将 http_userid 和此选项一起使用。

有关如何设置 dbmlsync 网络协议选项的信息，请参见“CommunicationAddress (adr) 扩展选项”一节第 179 页。

有关如何设置 UltraLite 网络协议选项的信息，请参见“UltraLite 同步流的网络协议选项”一节《UltraLite - 数据库管理和参考》。

另请参见

- “http_userid”一节第 55 页
- “http_proxy_password”一节第 53 页
- “http_proxy_userid”一节第 54 页

示例

下面的嵌入式 SQL 或 C++ UltraLite 应用程序示例提供了一个用于 Web 服务器基本验证的用户 ID 和口令。

```
synch_info.stream = "https";
synch_info.stream_parms = TEXT("http_userid=user;http_password=pwd");
```


http_proxy_password

使用 RFC 2617 基本或摘要式验证向第三方 HTTP 代理验证。

语法

http_proxy_password=*password*

协议

- HTTP、HTTPS

支持附注

- 不可以在重定向器的 ML 指令中使用。

缺省值

无

注释

此功能支持 RFC 2617 中所述的基本验证和摘要式验证。

在基本验证中，口令以明文形式包含在 HTTP 标头中；您可以使用 HTTPS，但与代理的初始连接是通过 HTTP 实现，因此该口令为明文。使用摘要式验证时，标头不以明文形式发送，而是采取散列码形式。

必须将 http_proxy_userid 和此选项一起使用。

有关如何设置 dbmlsync 网络协议选项的信息，请参见“CommunicationAddress (adr) 扩展选项”一节第 179 页。

有关如何设置 UltraLite 网络协议选项的信息，请参见“UltraLite 同步流的网络协议选项”一节《UltraLite - 数据库管理和参考》。

另请参见

- “http_password”一节第 52 页
- “http_userid”一节第 55 页
- “http_proxy_userid”一节第 54 页

示例

下面的嵌入式 SQL 或 C++ UltraLite 应用程序示例提供了一个用于 Web 代理基本验证的用户 ID 和口令。

```
synch_info.stream = "https";
synch_info.stream_parms =
TEXT("http_proxy_userid=user;http_proxy_password=pwd");
```

http_proxy_userid

使用 RFC 2617 基本验证或摘要式验证向第三方 HTTP 代理验证身份。

语法

http_proxy_userid=*userid*

协议

- HTTP、HTTPS

支持附注

- 不可以在重定向器的 ML 指令中使用。

缺省值

无

注释

此功能支持 RFC 2617 中所述的基本验证和摘要式验证。

在基本验证中，口令以明文形式包含在 HTTP 标头中；您可以使用 HTTPS，但与代理的初始连接是通过 HTTP 实现，因此该口令为明文。使用摘要式验证时，标头不以明文形式发送，而是采取散列码形式。

必须将 http_proxy_password 和此选项一起使用。

有关如何设置 dbmlsync 网络协议选项的信息，请参见“CommunicationAddress (adr) 扩展选项”一节第 179 页。

有关如何设置 UltraLite 网络协议选项的信息，请参见“UltraLite 同步流的网络协议选项”一节《UltraLite - 数据库管理和参考》。

另请参见

- “http_password”一节第 52 页
- “http_userid”一节第 55 页
- “http_proxy_password”一节第 53 页

示例

下面的嵌入式 SQL 或 C++ UltraLite 应用程序示例提供了一个用于 Web 代理基本验证的用户 ID 和口令。

```
synch_info.stream = "https";
synch_info.stream_parms =
TEXT("http_proxy_userid=user;http_proxy_password=pwd");
```


http_userid

使用 RFC 2617 基本验证或摘要式验证向第三方 HTTP 服务器和网关验证身份。

语法

http_userid=*userid*

协议

- HTTP、HTTPS

支持附注

- 不可以在重定向器的 ML 指令中使用。

缺省值

无

注释

此功能支持 RFC 2617 中所述的基本验证和摘要式验证。

使用基本验证时，口令以明文形式包含在 HTTP 标头中；但可以使用 HTTPS 加密标头并保护该口令。使用摘要式验证时，标头不以明文形式发送，而是采取散列码形式。

必须将 http_password 和此选项一起使用。

有关如何设置 dbmlsync 网络协议选项的信息，请参见“CommunicationAddress (adr) 扩展选项”一节第 179 页。

有关如何设置 UltraLite 网络协议选项的信息，请参见“UltraLite 同步流的网络协议选项”一节《UltraLite - 数据库管理和参考》。

另请参见

- “http_password”一节第 52 页
- “http_proxy_password”一节第 53 页
- “http_proxy_userid”一节第 54 页

示例

下面的嵌入式 SQL 或 C++ UltraLite 应用程序示例提供了一个用于 Web 服务器基本验证的用户 ID 和口令。

```
synch_info.stream = "https";
synch_info.stream_parms = TEXT("http_userid=user;http_password=pwd");
```

identity_name

此功能支持使用通用访问卡 (CAC) 提供客户端身份（证书以及专用密钥）进行验证。只有 Windows Mobile 支持此功能。

此参数用于指定公共证书的公用名。

需要单独授予许可的组成部分
此功能是 CAC 验证插件的一部分，需要单独的许可。请参见“单独授权的组件”一节 《SQL Anywhere 11 - 简介》 。

语法

identity_name=*name*

协议

- TLS、HTTPS

缺省值

无。

注释

此参数只能与 FIPS 认可的 RSA 加密一起使用。

必须将公共证书安装在设备的证书存储库中。

需要单独授予许可的组成部分
ECC 加密和 FIPS 认证的加密需要单独的许可。所有高度加密技术受出口法规约束。

请参见“单独授权的组件”一节 《SQL Anywhere 11 - 简介》 。

network_leave_open

在指定 network_name 时，还可以选择指定在同步完成后让网络连接继续保持打开状态。

语法

network_leave_open={ off | on }

协议

- TCPIP、TLS、HTTP、HTTPS

支持附注

- 不可以在重定向器的 ML 指令中使用。

缺省值

在除 Palm 之外的设备上，缺省值为 **off**。

在 Palm 上，缺省值为 **on**。

注释

必须指定 network_name 才能使用此选项。

如果此选项设置为 on，则网络连接在同步完成后继续保持打开状态。

有关如何设置 dbmlsync 网络协议选项的信息，请参见“CommunicationAddress (adr) 扩展选项”一节第 179 页。

有关如何设置 UltraLite 网络协议选项的信息，请参见“UltraLite 同步流的网络协议选项”一节《UltraLite - 数据库管理和参考》。

另请参见

- “network_name”一节第 58 页

示例

在以下示例中，客户端使用网络名 MyNetwork，并指定在同步完成后使连接保持打开状态。

在 SQL Anywhere 客户端上，实现如下：

```
dbmlsync -e "adr='network_name=MyNetwork;network_leave_open=on'"
```

对于以嵌入式 SQL 或 C++ 编写的 UltraLite 应用程序，实现如下：

```
synch_info.stream_parms =
TEXT("network_name=MyNetwork;network_leave_open=on");
```


network_name

指定在尝试连接到网络失败时所要启动的网络名。

语法

network_name=*name*

协议

- TCPIP、TLS、HTTP、HTTPS

支持附注

- 不可以在重定向器的 ML 指令中使用。

缺省值

无

注释

指定网络名以便可以使用 MobiLink 自动拨号功能。这样，您从 Windows Mobile 设备或 Windows 台式计算机进行连接时不必手工拨号。自动拨号是连接 MobiLink 服务器的第二次尝试；首先，客户端尝试在不拨号的情况下连接，如果失败且指定了 network_name，则激活自动拨号。与调度一起使用时，远程设备可在无人照管的情况下进行同步。在不使用调度时使用此特性，您不必手工进行拨号连接即可运行 dbmlsync。

在 Windows Mobile 上，*name* 应该是 [设置] » [连接] » [连接] 的下拉列表中的网络配置文件之一。要将您所设置的所有内容用作 Internet 网络或工作网络的缺省值，请将 name 分别设置为关键字 **default_internet** 或 **default_work**。

在 Windows 桌面操作系统平台上，*name* 应该是 [网络和拨号连接] 中的网络配置文件之一。

有关如何设置 dbmlsync 网络协议选项的信息，请参见“CommunicationAddress (adr) 扩展选项”一节第 179 页。

有关如何设置 UltraLite 网络协议选项的信息，请参见“UltraLite 同步流的网络协议选项”一节《UltraLite - 数据库管理和参考》。

另请参见

- “调度同步”一节第 114 页
- “network_leave_open”一节第 57 页

示例

在以下示例中，客户端使用网络名 MyNetwork，并指定在同步完成后使连接保持打开状态。

在 SQL Anywhere 客户端上，实现如下：

```
dbmlsync -e "adr='network_name=MyNetwork;network_leave_open=on'"
```

对于以嵌入式 SQL 或 C++ 编写的 UltraLite 应用程序，实现如下：


```
synch_info.stream_parms =
TEXT("network_name=MyNetwork;network_leave_open=on");
```

persistent

为同步中的所有 HTTP 请求使用一个 TCP/IP 连接。

语法

persistent={ off | on }

协议

- HTTP、HTTPS

支持附注

- 不可以在重定向器的 ML 指令中使用。

缺省值

Off

注释

值 On 表示客户端尝试对一个同步过程中的所有 HTTP 请求都使用同一个 TCP/IP 连接。设置为 off 时通常会与中间代理取得较好的兼容性。

除了在 Palm 设备上之外，只有直接与 MobiLink 连接时才应将 persistent 设置为 on。如果您是通过中间代理（例如代理或重定向程序）连接的，则持续连接可能会造成问题。

有关如何设置 dbmlsync 网络协议选项的信息，请参见“CommunicationAddress (adr) 扩展选项”一节第 179 页。

有关如何设置 UltraLite 网络协议选项的信息，请参见“UltraLite 同步流的网络协议选项”一节《UltraLite - 数据库管理和参考》。

port

指定 MobiLink 服务器的套接字端口号。

语法

port=*port-number*

协议

- TCPIP、TLS、HTTP、HTTPS

缺省值

对于 TCP/IP，缺省值为 **2439**，这是 MobiLink 服务器的 IANA 注册端口号。

对于 HTTP，缺省值为 **80**。

对于 HTTPS，缺省值为 **443**。

注释

该端口号必须是十进制数，并应与 MobiLink 服务器所监听的端口相匹配。

如果您通过 Web 服务器进行同步，请指定接收 HTTP 或 HTTPS 请求的 Web 服务器端口。

有关如何设置 dbmlsync 网络协议选项的信息，请参见“CommunicationAddress (adr) 扩展选项”一节第 179 页。

有关如何设置 UltraLite 网络协议选项的信息，请参见“UltraLite 同步流的网络协议选项”一节《UltraLite - 数据库管理和参考》。

示例

在以下示例中，客户端在端口 1234 上连接到名为 myhost 的计算机。

在 SQL Anywhere 客户端上，实现如下：

```
dbmlsync -e "adr='host=myhost;port=1234'"
```

对于以嵌入式 SQL 或 C++ 编写的 UltraLite 应用程序，实现如下：

```
synch_info.stream_parms = TEXT("host=myhost;port=1234");
```

proxy_host

指定代理服务器的主机名或 IP 地址。

语法

proxy_host=*proxy-hostname-or-ip*

协议

- HTTP、HTTPS

支持附注

- 不可以在重定向器的 ML 指令中使用。

缺省值

无

注释

仅在经过 HTTP 代理时使用。

有关如何设置 dbmlsync 网络协议选项的信息，请参见“CommunicationAddress (adr) 扩展选项”一节第 179 页。

有关如何设置 UltraLite 网络协议选项的信息，请参见“UltraLite 同步流的网络协议选项”一节《UltraLite - 数据库管理和参考》。

示例

在以下示例中，客户端在端口 1234 上连接到名为 myproxyhost 的计算机上所运行的代理服务器。

在 SQL Anywhere 客户端上，实现如下：

```
dbmlsync -e "adr='proxy_host=myproxyhost;proxy_port=1234'"
```

对于以嵌入式 SQL 或 C++ 编写的 UltraLite 应用程序，实现如下：

```
synch_info.stream_parms = TEXT("proxy_host=myproxyhost;proxy_port=1234");
```


proxy_port

指定代理服务器的端口号。

语法

proxy_port=*proxy-port-number*

协议

- HTTP、HTTPS

支持附注

- 不可以在重定向器的 ML 指令中使用。

缺省值

无

注释

仅在经过 HTTP 代理时使用。

有关如何设置 dbmlsync 网络协议选项的信息，请参见“CommunicationAddress (adr) 扩展选项”一节第 179 页。

有关如何设置 UltraLite 网络协议选项的信息，请参见“UltraLite 同步流的网络协议选项”一节《UltraLite - 数据库管理和参考》。

示例

在以下示例中，客户端在端口 1234 上连接到名为 myproxyhost 的计算机上所运行的代理服务器。

在 SQL Anywhere 客户端上，实现如下：

```
dbmlsync -e "adr='proxy_host=myproxyhost;proxy_port=1234'"
```

对于以嵌入式 SQL 或 C++ 编写的 UltraLite 应用程序，实现如下：

```
synch_info.stream_parms = TEXT("proxy_host=myproxyhost;proxy_port=1234");
```


set_cookie

指定在同步操作中用到的 HTTP 请求中设置的自定义 HTTP Cookie。

语法

set_cookie=*cookie-name***=***cookie-value*,...

协议

- HTTP、HTTPS

支持附注

- 不可以在重定向器的 ML 指令中使用。

缺省值

无

注释

当同步客户端与使用 cookie 标识会话的第三方工具（例如验证工具）交互时，自定义 HTTP cookie 会非常有用。例如，您有一个系统，其中有一个用户代理连接到一个 Web 服务器、代理或网关并验证其本身。如果成功，该代理会从服务器接收到一个或多个 Cookie。然后该代理启动一个同步并通过 set_cookie 选项传递其会话 cookie。

如果具有多个名称值对，则用逗号分隔它们。

有关如何设置 dbmlsync 网络协议选项的信息，请参见“CommunicationAddress (adr) 扩展选项”一节第 179 页。

有关如何设置 UltraLite 网络协议选项的信息，请参见“UltraLite 同步流的网络协议选项”一节《UltraLite - 数据库管理和参考》。

示例

以下示例在某个嵌入式 SQL 或 C++ UltraLite 应用程序中设置了一个自定义 HTTP cookie。

```
info.stream = "http";
info.stream_parms = TEXT(
  "host=www.myhost.com;
  set_cookie=MySessionID=12345, enabled=yes;");
```


timeout

指定客户端在放弃之前等待网络操作成功的时间量（以秒为单位）。

语法

timeout=*seconds*

协议

- TCPIP、TLS、HTTP、HTTPS

缺省值

240 秒

支持附注

- 不可以在重定向器的 ML 指令中使用。

注释

如果有任何连接、读取或写入尝试未能在指定时间内完成，则客户端将放弃同步。

在整个同步过程中，客户端会在指定时间间隔内发送活动更新，使 MobiLink 服务器知道它仍处于活动状态，然后 MobiLink 会回发活动更新，使客户端知道它仍处于活动状态。为防止慢速网络的延迟超过指定的时间，MobiLink 客户端以超时值的一半为时间间隔向 MobiLink 服务器发送用于保持活动状态的字节。如果将此值设置为 240 秒，则每 120 秒发送一次用于保持活动状态的消息。

注意不要将超时值设置得过低。活动检查增加了网络通信量，因为 MobiLink 服务器和客户端必须在每个超时期内进行通信以确保连接仍处于活动状态。如果网络或服务器负荷非常重而超时期又非常短，则很有可能会放弃某个活动的连接，因为 MobiLink 服务器和 dbmlsync 无法确定该连接是否仍处于活动状态。活动超时时间通常不应少于 30 秒。

最大超时时间为 10 分钟。您可以指定大于 600 秒的数值，但仍会将其解释为 600 秒。

值为 0 表示超时时间为 10 分钟。

有关如何设置 dbmlsync 网络协议选项的信息，请参见“CommunicationAddress (adr) 扩展选项”一节第 179 页。

有关如何设置 UltraLite 网络协议选项的信息，请参见“UltraLite 同步流的网络协议选项”一节《UltraLite - 数据库管理和参考》。

示例

以下示例将超时时间设置为 300 秒。

在 SQL Anywhere 客户端上，实现如下：

```
dbmlsync -e "adr=timeout=300"
```

对于以嵌入式 SQL 或 C++ 编写的 UltraLite 应用程序，实现如下：

```
synch_info.stream_parms = TEXT("timeout=300");
```


tls_type

指定用于同步的加密编码器。

需要单独授予许可的组成部分
ECC 加密和 FIPS 认证的加密需要单独的许可。所有高度加密技术受出口法规约束。

请参见“单独授权的组件”一节 《SQL Anywhere 11 - 简介》。

语法

tls_type=*cipher*

协议

- TLS、HTTPS

缺省值

RSA

注释

此同步的所有通信都将使用指定编码器进行加密。编码器可以是以下各项之一：

- **ecc** 用于椭圆曲线加密。
- **rsa** 用于 RSA 加密。

有关如何设置 dbmlsync 网络协议选项的信息，请参见“CommunicationAddress (adr) 扩展选项”一节第 179 页。

另请参见

- “配置 MobiLink 客户端以使用传送层安全”一节 《SQL Anywhere 服务器 - 数据库管理》
- “fips”一节第 49 页
- “加密 MobiLink 客户端/服务器通信”一节 《SQL Anywhere 服务器 - 数据库管理》
- “-x 选项”一节 《MobiLink - 服务器管理》
- “certificate_company”一节第 39 页
- “certificate_name”一节第 41 页
- “certificate_unit”一节第 43 页
- “trusted_certificates”一节第 68 页

示例

以下示例为 TCP/IP 协议设置了 RSA 加密。这需要在服务器和客户端上进行设置。每个命令都必须在一行中写入。

在服务器上，实现如下：

```
mlsrv11
  -c "dsn=SQL Anywhere 11 Demo;uid=DBA;pwd=sql"
  -x tls(
    port=9999;
```



```
      tls_type=rsa;
      identity=c:\sa10\bin32\rsaserver.id;
      identity_password=test )
```

在 SQL Anywhere 客户端上，实现如下：

```
  dbmlsync -e
     "CommunicationType=tls;
      CommunicationAddress=
        'tls_type=rsa;
         trusted_certificates=\rsaroot.crt;
         certificate_name=RSA Server'"
```

对于以 C 或 C++ 语言编写的嵌入式 SQL UltraLite 应用程序，实现如下：

```
       info.stream = "tls";
       info.stream_parms = TEXT(
        "tls_type=rsa;
         trusted_certificates=\rsaroot.crt;
         certificate_name=RSA Server");
```

trusted_certificates

指定一个文件，其中包含一个用于安全同步的受信任根证书的列表。

需要单独授予许可的组成部分
ECC 加密和 FIPS 认证的加密需要单独的许可。所有高度加密技术受出口法规约束。

请参见“单独授权的组件”一节 《SQL Anywhere 11 - 简介》。

语法

trusted_certificates=*filename*

语法 2 (Palm OS)

trusted_certificates=vfs:[*volume-label*:| *volume-ordinal*:]*filename*

协议

- TLS、HTTPS

缺省值

无

注释

通过 Certicom TLS 同步流进行同步时，MobiLink 服务器将它的证书连同签名实体的证书直到自签名根证书一起发送给客户端。

客户端将检查该链是否有效，以及是否信任该链中的根证书。此功能允许您指定受信任的根证书。

对于 UltraLite 客户端，在创建数据库时，可将受信任的根证书提供给 ulinit、ulcreate 和 ulload。如果提供了 trusted_certificates 参数，则在文件中找到的受信任的根证书将替换存储在数据库中的那些根证书。

对于 32 位 Windows 和 Windows CE，如果未指定受信任的证书，客户端将从操作系统的受信任的证书存储库装载证书。当 Web 浏览器通过 HTTPS 与安全的 Web 服务器连接时，将使用这个证书存储库。

Palm OS 文件系统（而不是基于记录的数据存储区）支持受信任的证书。在 Palm OS 上，*volume-label* 可以是 **INTERNAL**（对于内置驱动器）、**CARD**（对于扩展卡）或卷的标签名。也可以使用 *volume-ordinal* 对卷进行标识（缺省值为 0，即平台枚举的第一个卷）。*filename* 必须是文件的完整路径，采用的是 Palm 平台的文件名和路径命名约定。

有关如何设置 dbmlsync 网络协议选项的信息，请参见“CommunicationAddress (adr) 扩展选项”一节第 179 页。

有关如何设置 UltraLite 网络协议选项的信息，请参见“UltraLite 同步流的网络协议选项”一节《UltraLite - 数据库管理和参考》。

另请参见

- “在 UltraLite 连接参数中指定文件路径”一节 《UltraLite - 数据库管理和参考》
- “加密 MobiLink 客户端/服务器通信”一节 《SQL Anywhere 服务器 - 数据库管理》
- “-x 选项”一节 《MobiLink - 服务器管理》
- “tls_type”一节第 66 页
- “certificate_company”一节第 39 页
- “certificate_name”一节第 41 页
- “certificate_unit”一节第 43 页

示例

以下示例为某个 HTTPS 协议设置了 RSA 加密。这需要在服务器和客户端上进行设置。每个命令都必须在一行中写入。

服务器实现如下：

```
mlsrv11
   -c "dsn=SQL Anywhere 11 Demo;uid=DBA;pwd=sql"
   -x https(
     port=9999;
     identity=c:\sa10\bin32\rsaserver.id;
     identity_password=test)
```

在 SQL Anywhere 客户端上，实现如下：

```
dbmlsync
   -c "dsn=mydb;uid=DBA;pwd=sql"
   -e "ctp=https;
       adr='port=9999;
         trusted_certificates=c:\sa10\bin32\rsaroot.crt;
         certificate_name=RSA Server'"
```

在 UltraLite 客户端上，实现如下：

```
     info.stream = "https";
     info.stream_parms = TEXT(
       "port=9999;
        trusted_certificates=\rsaroot.crt;
        certificate_name=RSA Server");
```

在运行 Palm OS 的 UltraLite 客户端上，stream 和 stream_parms 可设置如下：

```
info.stream = "https";
info.stream_parms = "trusted_certificates=vfs:/rsaroot.crt;port=9999";
```

url_suffix

指定要向同步过程中发送的每个 HTTP 请求的第一行中的 URL 添加的后缀。

语法

url_suffix=*suffix*

suffix 的语法取决于您所使用的重定向器的类型：

重定向器	***suffix* 的语法**
ISAPI	`exe_dir/iaredirect.dll/ml/[server-group/]` 其中 *exe-dir* 是 *iaredirect.dll* 的位置。
NSAPI	`mlredirect/ml/[server-group/]` 其中 *mlredirect* 是在 *obj.conf* 文件中映射的名称。
Apache	您在 *httpd.conf* 文件的重定向器 <location> 标记中选择的任何内容。
Servlet	`iaredirect/ml/`
M-Business Anywhere	您在 *sync.conf* 文件的重定向器 <location> 标记中选择的任何内容。

协议

- HTTP、HTTPS

支持附注

- 不可以在重定向器的 ML 指令中使用，但可在重定向器的客户端上设置。

缺省值

缺省值为 **MobiLink**。

注释

通过代理或 Web 服务器进行同步时，可能需要 url_suffix 才能找到 MobiLink 服务器。

只有某些重定向器支持服务器组。有关详细信息，请参见 http://www.sybase.com/detail?id=1062632。

有关在使用重定向器时如何设置此选项的信息，请参见“为重定向器配置 MobiLink 客户端和服务器”一节 《MobiLink - 服务器管理》。

有关如何设置 dbmlsync 网络协议选项的信息，请参见“CommunicationAddress (adr) 扩展选项”一节第 179 页。

有关如何设置 UltraLite 网络协议选项的信息，请参见“UltraLite 同步流的网络协议选项”一节 《UltraLite - 数据库管理和参考》。

另请参见

- “为重定向器配置 MobiLink 客户端和服务器”一节 《MobiLink - 服务器管理》
- “MobiLink 服务器组”一节 《MobiLink - 服务器管理》

示例

以下 SQL 语句将创建一个名为 sales5322 的同步用户，该用户将通过 HTTPS 进行同步。假设 MobiLink 服务器在公司防火墙后面运行，并且使用重定向器（与 NSAPI Web 服务器连接的反向代理）将同步请求重定向给该服务器。该 MobiLink 用户将同步到 URL *https://www.mycompany.com:80/mlredirect/ml/*。

```
CREATE SYNCHRONIZATION USER sales5322
TYPE https
ADDRESS 'host=www.mycompany.com;port=80;url_suffix=mlredirect/ml/'
```

示例

以下示例通过 ISAPI 重定向器为 UltraLite 客户端设置 HTTP。命令行的所有内容必须在一行中输入。

```
mlsrv11 -c "dsn=SQL Anywhere 11 CustDB"
  -dl -fr -ot mlserver.mls -zu+ -v+
  -x http(port=8081)
```

要与 ulsync 进行同步，请运行类似以下的命令：

```
ulsync -c "dbf=custdb.udb"
  -v "MobiLinkUid=50;ScriptVersion=custdb 11.0;
   Stream=http(port=80;url_suffix=Scripts/iaredirect.dll/ml/)"
```

version

指定用于进行同步的 HTTP 的版本。

语法

version=*HTTP-version-number*

协议

- HTTP、HTTPS

支持附注

- 不可以在重定向器的 ML 指令中使用。

缺省值

缺省值为 **1.1**。

注释

如果您的 HTTP 基础结构要求特定版本的 HTTP，此选项会非常有用。值可以是 **1.0** 或 **1.1**。

有关如何设置 dbmlsync 网络协议选项的信息，请参见“CommunicationAddress (adr) 扩展选项”一节第 179 页。

有关如何设置 UltraLite 网络协议选项的信息，请参见“UltraLite 同步流的网络协议选项”一节《UltraLite - 数据库管理和参考》。

示例

以下示例将 HTTP 版本设置为 1.0。

在 SQL Anywhere 客户端上，实现如下：

```
dbmlsync -e "adr=version=1.0"
```

对于以嵌入式 SQL 或 C++ 编写的 UltraLite 应用程序，实现如下：

```
synch_info.stream_parms = TEXT("version=1.0");
```


zlib_download_window_size

如果将 compression 选项设置为 zlib，则可以使用此选项指定下载的压缩窗口大小。

语法

zlib_download_window_size=*window-bits*

协议

- TCPIP、TLS、HTTP、HTTPS

支持附注

- 不可以在重定向器的 ML 指令中使用。
- 在 Palm OS 或 Symbian 上不受支持。

缺省值

在 Windows Mobile 上是 12，其它情况下是 15

注释

要关闭下载压缩，可将 *window-bits* 设置为 0。其它情况下，窗口大小可以是介于 9 和 15 之间的值（包括 9 和 15）。一般而言，使用较大的窗口大小可实现较高的压缩率，但也需要更多内存。

window-bits 是以 2 为底的窗口大小（历史缓冲区大小）的对数。以下公式可用于确定对于每个 *window-bits* 在客户端上要使用多少内存：

upload (compress): memory = $2^{(window\text{-}bits + 3)}$

download (decompress): memory = $2^{(window\text{-}bits)}$

要在 UltraLite 中支持 zlib 压缩，应用程序必须调用 `ULEnableZlibSyncCompression( sqlca )`，并且必须部署 *mlczlib10.dll*。

有关如何设置 dbmlsync 网络协议选项的信息，请参见“CommunicationAddress (adr) 扩展选项”一节第 179 页。

有关如何设置 UltraLite 网络协议选项的信息，请参见“UltraLite 同步流的网络协议选项”一节《UltraLite - 数据库管理和参考》。

另请参见

- “Compression”一节第 45 页
- “zlib_download_window_size”一节第 74 页

示例

以下选项只为上载设置压缩：

```
"compression=zlib;zlib_download_window_size=0"
```

zlib_download_window_size

如果将 compression 选项设置为 zlib，则可以使用此选项指定上载的压缩窗口大小。

语法

zlib_upload_window_size=*window-bits*

协议

- TCPIP、TLS、HTTP、HTTPS

支持附注

- 不可以在重定向器的 ML 指令中使用。
- 在 Palm OS 或 Symbian 上不受支持。

缺省值

在 Windows Mobile 上是 12，其它情况下是 15

注释

要关闭上载压缩，可将窗口大小设置为 0。其它情况下，窗口大小可以是介于 9 和 15 之间的值（包括 9 和 15）。一般而言，使用较大的窗口大小可实现较高的压缩率，但也需要更多内存。

window-bits 是以 2 为底的窗口大小（历史缓冲区大小）的对数。以下公式可用于确定对于每个 *window-bits* 在客户端上要使用多少内存：

upload (compress): memory = $2^{(window\text{-}size + 3)}$

download (decompress): memory = $2^{(window\text{-}size)}$

要在 UltraLite 中支持 zlib 压缩，应用程序必须调用 `ULEnableZlibSyncCompression( sqlca )`，并且必须部署 *mlczlib10.dll*。

有关如何设置 dbmlsync 网络协议选项的信息，请参见“CommunicationAddress (adr) 扩展选项”一节第 179 页。

有关如何设置 UltraLite 网络协议选项的信息，请参见“UltraLite 同步流的网络协议选项”一节《UltraLite - 数据库管理和参考》。

另请参见

- “Compression”一节第 45 页
- “zlib_download_window_size”一节第 73 页

示例

以下选项只为下载设置压缩：

```
"compression=zlib;zlib_upload_window_size=0"
```


远程客户端中的模式更改

目录

MobiLink 客户端模式更改简介

随着需求的发展，部署的远程数据库可能需要更改模式。最常见的模式更改包括向现有表中添加新列或向数据库中添加新表。

小心
更改模式前要先成功进行同步。升级模式时不应该有打开的事务。

SQL Anywhere 远程数据库的模式升级

可以在远程 SQL Anywhere 数据库部署后更改其模式。

如果可以确保没有其它连接连接到远程数据库，则可以使用 ALTER PUBLICATION 语句手工添加新表或更改表到发布中。否则，必须使用 sp_hook_dbmlsync_schema_upgrade 挂接来升级模式。

请参见“sp_hook_dbmlsync_schema_upgrade”一节第 276 页。

♦ 将表添加到 SQL Anywhere 远程数据库中

1. 在统一数据库中添加关联的表脚本。

   同一脚本版本既可用于不带新表的远程数据库，又可用于带有新表的远程数据库。不过，如果新表的存在会更改现有表的同步方式，则必须创建一个新的脚本版本，同时还必须为使用该新脚本版本同步的所有表创建新脚本。

2. 执行常规同步。继续操作前应确保同步成功完成。

3. 使用 ALTER PUBLICATION 语句添加表。例如，

   ```
   ALTER PUBLICATION your_pub
      ADD TABLE table_name;
   ```

   您可以在 sp_hook_dbmlsync_schema_upgrade 挂接中使用此语句。请参见“sp_hook_dbmlsync_schema_upgrade”一节第 276 页。

   有关详细信息，请参见“ALTER PUBLICATION 语句 [MobiLink] [SQL Remote]”一节 《SQL Anywhere 服务器 - SQL 参考》。

4. 执行同步。如果需要，请使用新的脚本版本。

在远程数据库中更改表定义

在现有的表中更改列的数目或类型时应谨慎。当 MobiLink 客户端使用新模式进行同步时，它需要的脚本（例如 upload_update 或 download_cursor 这样的脚本）中应该包含与远程表中的所有列相对应的参数。而旧的远程数据库则需要使用只包含原始列作为参数的脚本。

♦ 更改已部署的 SQL Anywhere 远程数据库中发布的表

1. 在统一数据库中，创建一个新的脚本版本。

   有关详细信息，请参见“脚本版本”一节 《MobiLink - 服务器管理》。

2. 使用新的脚本版本，为发布中所有的表创建脚本，这些发布包含要更改的表且已使用旧脚本版本进行同步。

3. 在远程数据库中，使用旧脚本版本执行常规同步。继续操作前应确保同步成功完成。

4. 在远程数据库中，使用 ALTER PUBLICATION 语句临时从发布中删除该表。例如，

   ```
   ALTER PUBLICATION your_pub
     DROP TABLE table_name;
   ```


有关详细信息，请参见“ALTER PUBLICATION 语句 [MobiLink] [SQL Remote]”一节 《SQL Anywhere 服务器 - SQL 参考》。

您可以在 sp_hook_dbmlsync_schema_upgrade 挂接中使用此语句。请参见“sp_hook_dbmlsync_schema_upgrade”一节第 276 页。

5. 在远程数据库中，使用 ALTER TABLE 语句更改表。

   有关详细信息，请参见“ALTER TABLE 语句”一节 《SQL Anywhere 服务器 - SQL 参考》。

6. 在远程数据库中，使用 ALTER PUBLICATION 语句将表添加回发布中。

   有关详细信息，请参见“ALTER PUBLICATION 语句 [MobiLink] [SQL Remote]”一节 《SQL Anywhere 服务器 - SQL 参考》。

   您可以在 sp_hook_dbmlsync_schema_upgrade 挂接中使用此语句。请参见“sp_hook_dbmlsync_schema_upgrade”一节第 276 页。

7. 使用新脚本版本进行同步。

UltraLite 远程数据库的模式升级

可以通过让现有的应用程序执行 DDL 来更改远程 UltraLite 数据库的模式。

- 如果部署新的应用程序时包含新的数据库，则需要通过与 MobiLink 服务器的同步来重新填充 UltraLite 数据库。
- 如果部署一个包含 DDL 的新应用程序来升级数据库，则您的数据将被保留。
- 如果现有应用程序采用常规方式接收 DDL 语句，则可将 DDL 应用到您的数据库且您的数据将被保留。

将所有用户同时都升级到应用程序的新版本通常是不切实际的。因此，必须能够使两个版本在某一领域中共存，并与一个统一数据库进行同步。您可以创建两个或更多存储在统一数据库中的同步版本，并控制 MobiLink 服务器的操作。每个应用程序版本都可以在启动同步时指定正确的版本名称，从而选择一组合适的同步脚本。

有关 UltraLite DDL 的信息，请参见“UltraLite SQL 语句” 《UltraLite - 数据库管理和参考》。

另请参见

- “部署 UltraLite 模式升级”一节 《UltraLite - 数据库管理和参考》

SQL 直通

目录

SQL 直通简介

通过 SQL 直通功能，可将 SQL 语句的脚本从统一数据库下载到 SQL Anywhere 或 UltraLite 客户端，并于适当的时间在客户端上执行这些 SQL 语句。已对这些脚本编号，可保证在客户端上按顺序执行。

在执行某个脚本或尝试执行某个脚本之后，会在下次同步时将状态信息发送回统一数据库，这样便可集中监控成功和失败的运行。如果在客户端上执行某个脚本时发生错误，那么直到已将状态信息上载到统一数据库并且统一数据库已将有关如何继续的指令下载到客户端时，才会在此客户端上继续执行其它脚本。这些指令可能意味着重试现有脚本、将其跳过或下载新的脚本以修正该错误。

以下是使用 SQL 直通所需步骤的概述：

- 创建一个脚本，并将其存储在统一数据库中。请参见“创建脚本”一节第 82 页和“ml_add_passthrough_script 系统过程”一节 《MobiLink - 服务器管理》。
- 为该脚本创建一个或多个用于标识应执行它的客户端的直通条目。请参见“创建直通条目”一节第 82 页和“ml_add_passthrough 系统过程”一节 《MobiLink - 服务器管理》。
- 下载该脚本。请参见“下载脚本”一节第 83 页和“ml_add_passthrough 系统过程”一节 《MobiLink - 服务器管理》。
- 执行该脚本。请参见“执行脚本”一节第 83 页。
- 获取结果。请参见“获取脚本结果”一节第 86 页。
- 查看结果。请参见“查看脚本结果”一节第 86 页。
- 处理错误（如有必要）。请参见“处理脚本错误”一节第 86 页。

创建脚本

使用 ml_add_passthrough_script 存储过程在统一数据库上创建并存储脚本。请参见“ml_add_passthrough_script 系统过程”一节 《MobiLink - 服务器管理》。

创建直通条目

ml_add_passthrough 系统过程用于指定哪个 MobiLink 客户端应接收在统一数据库上创建的直通脚本。请参见“ml_add_passthrough 系统过程”一节 《MobiLink - 服务器管理》。

下载脚本

在同步过程中，会自动将脚本从统一数据库下载到 MobiLink 客户端。对于下列情况，将**不会**下载脚本：

- 执行基于文件的下载时。
- 执行 ping 时。
- 重新启动下载时。

SQL Anywhere 客户端上的 SQL 直通脚本存储

从统一数据库下载到 SQL Anywhere 客户端的 SQL 直通脚本存储在 dbo.sync_passthrough_script 表中。

UltraLite 客户端上的 SQL 直通脚本存储

从统一数据库下载到 UltraLite 客户端的 SQL 直通脚本存储在 syssql 表中。

执行脚本

一旦脚本存储在 MobiLink 客户端数据库中，即可自动或手动执行该脚本。无论以何种方式执行，都必须根据 run_order 按顺序执行脚本。

对于 SQL Anywhere 客户端，在 DBO 帐户下具有 DBA 权限才能执行脚本。

在 SQL Anywhere 客户端上手动执行脚本

所有脚本均可手动执行。对于 SQL Anywhere 客户端，使用 sync_get_next_passthrough_script 和 sync_execute_next_passthrough_script 函数手动执行脚本。

sync_get_next_passthrough_script 函数不带有任何参数，并返回要执行的下一脚本的 run_order。然后，可以通过查询 dbo.sync_passthrough_script 表来找到该脚本。

dbo.sync_passthrough_script 表

dbo.sync_passthrough_script 表的定义如下：

列名	说明
run_order	INTEGER。run_order 参数决定脚本在远程数据库上的应用顺序。脚本始终根据 run_order 按顺序应用。 其值必须是非负整数。
script_id	INTEGER。该值唯一标识脚本。

列名	说明
script_name	VARCHAR(128)。脚本的名称。此列对应于在统一数据库上调用 ml_add_passthrough_script 时为脚本指定的 script_name 值。
flags	BIGINT。flags 列包含传递到 ml_add_passthrough_script 存储过程的 flags 参数中所指定的信息。通过将每个指定的标记转换为以下值，并使用 OR 运算符将结果值组合起来，可将指定的这些标记编码为一个整数。 ● **manual** 表示该脚本只能在手动执行模式下运行。缺省情况下，所有脚本既能在自动执行模式下运行，又能在手动执行模式下运行。 ● **exclusive** 表示该脚本只能在同步结束时，在所有的同步表上获得独占锁的情况下自动执行。如果 affected_ publications 值没有列出任何发布，则忽略该选项。该选项仅对 SQL Anywhere 远程数据库有意义。 ● **schema_diff** 表示该脚本应在模式比较模式下运行。此模式下，会根据脚本所描述的模式对数据库模式进行更改。例如，将现有表的 create 语句视为 alter 语句。该标记只适用于在 UltraLite 远程数据库上运行的脚本。
affected_pubs	LONG VARCHAR。脚本运行前必须进行同步的发布列表。此列对应于调用 ml_add_passthrough_script 时为脚本指定的 affected_pubs 值。
script	LONG VARCHAR。直通脚本的内容。此列对应于调用 ml_add_passthrough_script 时为脚本指定的 script 值。
description	VARCHAR(2000)。脚本的注释或说明。此列对应于调用 ml_add_passthrough_script 时为脚本指定的 description 值。

如果没有其它要执行的脚本，或执行的上一脚本生成了错误并且未从服务器收到有关如何继续的说明，则 sync_get_next_passthrough_script 函数会返回空值。

sync_execute_next_passthrough_script 函数不带任何参数。此函数执行下一个脚本并更新数据库中的进度和状态信息，以便可在稍后将脚本的结果上载到统一数据库。如果上一个脚本返回一个错误，并且未从 MobiLink 服务器收到有关如何处理该错误的说明，则不会执行任何脚本。如果执行了某个脚本，则会返回该脚本的运行顺序。如果未执行任何脚本，则返回空值。请参见“获取脚本结果”一节第 86 页。

在 UltraLite 客户端上手动执行脚本

对于 UltraLite 客户端，可使用多种 ESQL API 方法来手动应用脚本。这些方法是：

- “GetSQLPassthroughScriptCount 方法”一节 《UltraLite - .NET 编程》
- “ExecuteSQLPassthroughScripts 方法”一节 《UltraLite - .NET 编程》
- “ExecuteNextSQLPassthroughScript 方法”一节 《UltraLite - .NET 编程》

对于 C++、UltraLite.NET 和 M-Business Anywhere 接口，可使用相同的方法。

在 SQL Anywhere 客户端上自动执行脚本

在 SQL Anywhere 客户端上，在每个同步结束时会尝试执行任何等待的脚本。可用脚本按照 run_order 排序并一次执行一个脚本，直到发生以下情况为止：

- 所有脚本已经执行完毕。
- 某个脚本执行失败。
- 到达某个脚本，但该脚本无法自动执行。

如果满足以下任一情况，则无法自动执行脚本：

- 创建脚本时，指定了 **manual** 标记。
- 脚本具有非空 **affected publications** 值，并满足以下一个或多个条件：
  - 未执行上载。
  - 上载失败。
  - **affected publications** 值中所列的一个或多个发布未同步。
  - 创建脚本时指定了 **exclusive** 标记，而在同步开始时并未在所有同步表上获得独占锁。

> **注意**
> 切勿将仅下载发布列为受影响的发布。

在同步开始时，dbmlsync 可能会选择在同步表上获取比使用 LockTables 扩展选项所请求的具有更多限制的锁，以确保可在同步结束时执行脚本。例如，如果将 LockTables 设置为 SHARE，但要执行下一个脚本需要独占锁，则可能会获取独占锁。

在 UltraLite 客户端上自动执行脚本

在 UltraLite 客户端上，除非已经设置了连接参数 **dont_run_scripts**，否则下次启动数据库时会自动运行未标记为 **manual** 的脚本。

如果提供了进度观察器回调函数，则在手动和自动执行期间均可提供脚本的执行进度，该函数定义如下：

```
typedef void(UL_CALLBACK_FN *ul_sql_passthrough_observer_fn)
    ( ul_sql_passthrough_status * status );
```

ul_sql_passthrough_status 结构定义如下：

```
typedef struct {
    ul_sql_passthrough_state    state; // current state
    ul_u_long  script_count; // total number of scripts to execute
    ul_u_long  cur_script; // current script being executed (1-based)
    ul_bool  stop;  // set to true to stop script execution
         // can only be set in the starting state
    ul_void *  user_data; // user data provided in register call
    SQLCA *  sqlca;
} ul_sql_passthrough_status;
```

进度观察器回调函数通过以下方法注册：

```
UL_FN_SPEC ul_ret_void UL_FN_MOD ULRegisterSQLPassthroughCallback(
SQLCA *                          sqlca,
ul_sql_passthrough_observer_fn   callback,
ul_void *                        user_data );
```

获取脚本结果

无论手动还是自动执行脚本，脚本执行的结果都存储在 MobiLink 客户端。

在 SQL Anywhere 客户端上，结果存储在 dbo.sync_passthrough_status 表中。结果包括脚本在远程数据库执行的时间以及有关脚本是执行成功还是报告了错误的指示。此外，如果报告了错误，则还会存储 SQL 代码和错误消息的文本。

在 UltraLite 客户端上，结果存储在 syssql 表中。结果包括脚本在远程数据库执行的时间以及有关脚本是执行成功还是报告了错误的指示。此外，如果报告了错误，则还会存储 SQL 代码、脚本中失败的语句的行号以及错误参数的列表。

查看脚本结果

UltraLite 和 SQL Anywhere 客户端都会将已执行的脚本的结果作为每次同步的一部分进行上载。MobiLink 服务器将上载的结果存储于统一数据库的 ml_passthrough_status 表中。通过检查该表可以确定分发到客户端的直通脚本是否执行成功。请参见“ml_passthrough_status”一节 《MobiLink - 服务器管理》。

处理脚本错误

小心
SQL 直通功能极其强大，使用时务必**谨慎**。对脚本进行全面测试尤为重要，因为 SQL 直通脚本的错误可能会禁用或破坏您的所有远程数据库。请进行全面测试以避免出现错误。

当 SQL 直通脚本在客户端出现错误时，必须在统一数据库解决这些错误，否则，客户端无法再运行任何其它 SQL 直通脚本。

mlsrv11 的 -vo 服务器选项可用于在 MobiLink 服务器上捕获 SQL 直通活动，以帮助您解决错误。请参见“-v 选项”一节 《MobiLink - 服务器管理》。

解决客户端上的错误的主要机制是使用 ml_add_passthrough_repair 系统过程向 ml_passthrough_repair 表中添加行。ml_passthrough_repair 表中的行描述当特定脚本生成特定错误代码时客户端应执行的操作。

另请参见

- “ml_passthrough_repair”一节 《MobiLink - 服务器管理》
- “ml_add_passthrough_repair 系统过程”一节 《MobiLink - 服务器管理》
- “ml_delete_passthrough_repair 系统过程”一节 《MobiLink - 服务器管理》

用于 MobiLink 的 SQL Anywhere 客户端

本节包含介绍如何设置和运行用于 MobiLink 同步的 SQL Anywhere 客户端的材料。

SQL Anywhere 客户端

目录

创建远程数据库

任何 SQL Anywhere 数据库都可用作 MobiLink 系统中的远程数据库。您只需完成以下工作：创建发布、创建 MobiLink 用户、在统一数据库中注册该用户及为该 MobiLink 用户预订发布。

如果您使用 [**创建同步模型向导**] 创建 MobiLink 客户端应用程序，则当您部署该模型时会创建这些对象。即使这样，您也应该理解这些概念。

♦ **将 SQL Anywhere 数据库用作远程数据库**

1. 开始时可以使用现有 SQL Anywhere 数据库，也可以创建新的数据库，然后在其中添加表。
2. 在远程数据库中创建一个或多个发布。

   请参见“发布数据”一节第 94 页。
3. 在远程数据库中创建 MobiLink 用户。

   请参见“创建 MobiLink 用户”一节第 101 页。
4. 在统一数据库中注册用户。

   请参见“向统一数据库添加 MobiLink 用户名”一节第 10 页。
5. 为 MobiLink 用户预订一个或多个发布。

   请参见“创建同步预订”一节第 104 页。

部署远程数据库

若要部署 SQL Anywhere 远程数据库，需要创建数据库并添加相应的发布和预订。要执行此操作，可以对某个原型远程数据库进行自定义。

将启动数据库部署到多个位置时，最安全的方法是部署远程 ID 为 NULL 的数据库。如果已通过同步将数据库进行了预填充，可先将远程 ID 的设置恢复为 NULL，然后再进行部署。此方法确保了远程 ID 的唯一性，因为远程数据库首次进行同步时系统会为其指派唯一的远程 ID。也可以将远程 ID 作为一个远程设置步骤来进行设置，但它必须是唯一的。

请参见“设置远程 ID”一节第 91 页。

♦ **通过自定义原型部署 MobiLink 远程数据库：**

1. 创建原型远程数据库。

   原型数据库应包含所需的全部表和发布，但不包含特定于每个数据库的数据。这些信息通常包含以下内容：

   - MobiLink 用户名。
   - 同步预订。
   - 为全局自动增量键值提供起始值的 global_database_id 选项。
2. 对于每个远程数据库，执行以下操作：

- 创建一个存放远程数据库的目录。
- 将原型远程数据库复制到该目录下。

  如果事务日志保存在远程数据库所在的目录，则无需更改日志文件名。
- 运行 SQL 脚本，将单独的信息添加到该数据库。

  该 SQL 脚本可以是参数化脚本。有关参数化脚本的信息，请参见“PARAMETERS 语句 [Interactive SQL]”一节 《SQL Anywhere 服务器 - SQL 参考》和“使用 SQL 命令文件”一节 《SQL Anywhere 服务器 - SQL 的用法》。

当您使用 [**创建同步模型向导**] 创建 MobiLink 客户端应用程序时，您可以使用向导来部署数据库。请参见“部署模型”一节 《MobiLink - 入门》。

另请参见

- “部署 SQL Anywhere MobiLink 客户端”一节 《MobiLink - 服务器管理》
- “首次同步始终有效”一节第 93 页

示例

以下 SQL 脚本是从 Contact 示例中抽取的。可以在 *samples-dir\MobiLink\Contact\customize.sql* 中找到它。（有关 *samples-dir* 的信息，请参见“示例目录”一节 《SQL Anywhere 服务器 - 数据库管理》。）

```
PARAMETERS ml_userid, db_id;
go
SET OPTION PUBLIC.global_database_id = {db_id}
go

CREATE SYNCHRONIZATION USER {ml_userid}
        TYPE 'TCPIP'
        ADDRESS 'host=localhost;port=2439'
        OPTION MEM=''
go
CREATE SYNCHRONIZATION SUBSCRIPTION TO "DBA"."Product"
        FOR {ml_userid}
go
CREATE SYNCHRONIZATION SUBSCRIPTION TO "DBA"."Contact"
        FOR {ml_userid}
go
commit work
go
```

以下命令对数据源是 dsn_remote_1 的远程数据库执行该脚本：

```
dbisql -c "dsn=dsn_remote_1" read customize.sql [SSinger] [2]
```

设置远程 ID

远程 ID 是远程数据库在 MobiLink 同步系统中的唯一标识。创建 SQL Anywhere 数据库时，远程 ID 为 NULL。该数据库与 MobiLink 同步时，MobiLink 会检查其远程 ID 是否为 NULL，如果是，则会指派一个 GUID 作为远程 ID。设置后，数据库即一直使用该远程 ID，除非用户手工对其进行更改。

如果要在 MobiLink 事件脚本中或其它地方引用远程 ID，则可能需要将远程 ID 更改为更有意义的名称。要执行此操作，请设置远程数据库的 ml_remote_id 数据库选项。ml_remote_id 选项是一个存储在 SYSOPTION 系统表中的用户定义选项。更改该选项有两种方法，一种是使用 SET OPTION 语句，一种是使用 Sybase Central 的 SQL Anywhere 插件。

远程 ID 在同步系统内必须唯一。

有关更改数据库选项的详细信息，请参见：

- “SET OPTION 语句”一节 《SQL Anywhere 服务器 - SQL 参考》
- “设置数据库选项”一节 《SQL Anywhere 服务器 - 数据库管理》
- “SYSOPTION 系统视图”一节 《SQL Anywhere 服务器 - SQL 参考》

小心
更改远程 ID 的最安全时间是在首次同步之前。如果以后再更改远程 ID，则务必先完整、成功地执行同步，然后再进行更改。否则可能会丢失数据，进而使数据库处于不一致的状态。

另请参见

- “远程 ID”一节第 14 页

示例

以下 SQL 语句将远程 ID 的值设置为 HR001：

```
SET OPTION PUBLIC.ml_remote_id = 'HR001'
```

升级远程数据库

如果在旧版本上安装新版本的 SQL Anywhere 远程数据库，则统一数据库中的同步进度信息会变得不正确。更正此问题的方法是，将此用户 ml_user 表的 progress 列设置为 0（零）。

有关 ml_user MobiLink 系统表的详细信息，请参见“ml_user”一节 《MobiLink - 服务器管理》。

有关升级的详细信息，请参见“升级 SQL Anywhere MobiLink 客户端”一节 《SQL Anywhere 11 - 更改和升级》。

进度偏移

进度偏移是一个表示特定时间点的整数值，截至该时间点，预订的所有操作都已上载并确认完毕。dbmlsync 实用程序使用该偏移来决定要上载的数据。在远程数据库上，该偏移存储在 SYS.ISYSSYNC 系统表的 progress 列中。在统一数据库上，该偏移存储在 ml_subscription 表的 progress 列中。

对于每个远程数据库，远程数据库和统一数据库都会为每个预订保留一个偏移。MobiLink 用户进行同步时，将会确认与该 MobiLink 用户关联的所有预订的偏移，即使用户当时并不同步这些预订。这样做是必需的，因为多个发布可能包含相同的数据。唯一的例外情况是，dbmlsync 在尝试进行上载后才会检查预订的进度偏移。

如果远程数据库偏移与统一数据库偏移之间存在任何不一致，缺省行为是使用统一数据库的偏移值更新远程数据库偏移，然后发送一个基于这些偏移的新上载。大多数情况下，此缺省值是适用的。例如，它在以下情况下通常是适用的：使用备份恢复统一数据库而远程事务日志完好无损时，或虽然上载成功但因发生通信故障而使上载确认无法发送时。

大部分偏移不匹配问题都可以使用统一数据库进度值自动解决。在极少数情况下，必须进行干预才能解决进度偏移问题，这时可以使用 dbmlsync -r 选项。

有关详细信息，请参见“-r 选项”一节第 162 页。

首次同步始终有效

首次尝试同步新创建的预订时，并不会检查预订的进度偏移是否与统一数据库上的进度偏移相符。此功能使得在重新创建和同步远程数据库时不必删除其状态信息（在统一数据库中维护）。

如果远程数据库系统表 SYS.ISYSSYNC 中的列如下所述，dbmlsync 实用程序就会检测到首次同步：**progress** 列的值与 **created** 列的值相同，并且 **log_sent** 列的值为空。

不过，如果在同一上载中同步两个或更多个预订，且其中一个预订并不是首次进行同步，则会检查所有要同步的预订（包括首次进行同步的预订）的进度偏移。例如，如果为 dbmlsync -n 选项指定了两个发布 (-n pub1,pub2)，其中 pub1 进行过同步，而 pub2 未进行过同步，则会检查这两个预订的进度偏移是否与统一数据库中的进度偏移值相符。

有关详细信息，请参见：

- “ISYSSYNC 系统表”一节 《SQL Anywhere 服务器 - SQL 参考》
- “ml_subscription”一节 《MobiLink - 服务器管理》
- “事务日志文件”一节第 107 页

发布数据

发布是一个数据库对象，它标识将要同步的数据。它定义要上载的数据，并且限制可下载的表。（下载在 download_cursor 脚本中定义。）

发布由一个或多个项目组成。每个项目指定要同步的表的一个子集。该子集可以是整个表，也可以是其行和/或列的子集。发布中的每个项目都必须引用不同的表。

您可以创建预订，以将发布链接到用户。

可以使用 Sybase Central 或使用 CREATE PUBLICATION 语句创建发布。

在 Sybase Central 中，所有发布和项目都出现在 [**发布**] 文件夹中。

关于发布的说明

- 创建和删除发布需要 DBA 权限。
- 不能创建两个包含同一表的不同列子集的发布。
- 发布决定选择哪些列，但不决定这些列的发送顺序。这些列始终按在 CREATE TABLE 语句中定义它们的顺序进行发送。
- 每个项目都必须包括其引用的表主键中的所有列。
- 项目可以限制要同步的表列。通过使用 WHERE 子句，还可以限制要同步的表行。
- 发布中无法包含视图和存储过程。
- Sybase 基于消息的复制技术 SQL Remote 也使用发布和预订。SQL Remote 要求统一数据库和远程数据库中都有发布和预订。相比之下，MobiLink 发布只出现在 SQL Anywhere 远程数据库中。MobiLink 统一数据库使用同步脚本进行配置。

另请参见

- “CREATE PUBLICATION 语句 [MobiLink] [SQL Remote]”一节 《SQL Anywhere 服务器 - SQL 参考》

发布完整表

可创建的最简单发布由一组项目组成，每个项目包含一个表中的所有行和列。这些表必须已经存在。

♦ 发布一个或多个完整表（Sybase Central [管理] 模式）

1. 使用 SQL Anywhere 插件，以具有 DBA 权限的用户身份连接到远程数据库。
2. 打开 [**发布**] 文件夹。
3. 选择 [**文件**] » [**新建**] » [**发布**]。
4. 在 [**您要给新发布指定什么名称**] 字段中输入新发布的名称。单击 [**下一步**]。
5. 单击 [**下一步**]。

6. 在 [**可用表**] 列表中，选择一个表。单击 [**添加**]。
7. 单击 [**完成**]。

♦ **发布一个或多个完整表 (SQL)：**

1. 以具有 DBA 权限的用户身份连接到远程数据库。
2. 执行一个 CREATE PUBLICATION 语句，该语句指定新发布的名称和要发布的表。

   See “CREATE PUBLICATION 语句 [MobiLink] [SQL Remote]”一节 《SQL Anywhere 服务器 - SQL 参考》.

示例

以下语句创建一个发布，该发布将发布整个 customer 表：

```
CREATE PUBLICATION pub_customer (
 TABLE customer
)
```

以下语句创建一个发布，该发布包括 SQL Anywhere 示例数据库中一组表内每个表的所有列和行：

```
CREATE PUBLICATION sales (
 TABLE customer,
 TABLE sales_order,
 TABLE sales_order_items,
 TABLE product
)
```

只发布表中的某些列

您可以从 Sybase Central 创建一个包含表中所有行和部分列的发布，也可以通过在 CREATE PUBLICATION 语句中列出所需列来完成这种创建。

注意

- 如果创建的两个发布包含相同的表，但表中的列子集不同，则同时预订这两个发布的任何用户都无法进行同步。
- 项目必须包括表中的所有主键列。

♦ **只发布表中的某些列（Sybase Central [管理] 模式）**

1. 使用 SQL Anywhere 插件，以具有 DBA 权限的用户身份连接到远程数据库。
2. 打开 [**发布**] 文件夹。
3. 选择 [**文件**] » [**新建**] » [**发布**]。
4. 在 [**您要给新发布指定什么名称**] 字段中输入新发布的名称。单击 [**下一步**]。
5. 单击 [**下一步**]。

6. 在 [**可用表**] 列表中，选择一个表。单击 [**添加**]。
7. 单击 [**下一步**]。
8. 在 [**可用列**] 列表中，扩展可用列的列表。选择一列并单击 [**添加**]。
9. 单击 [**完成**]。

♦ **只发布表中的某些列 (SQL)：**

1. 以具有 DBA 权限的用户身份连接到远程数据库。
2. 执行指定了发布名和表名的 CREATE PUBLICATION 语句。在表名后面的括号中列出发布的列。

   See “CREATE PUBLICATION 语句 [MobiLink] [SQL Remote]”一节 《SQL Anywhere 服务器 - SQL 参考》.

示例

以下语句创建一个发布，该发布将发布 customer 表的 id、company_name 和 city 列的所有行：

```
CREATE PUBLICATION pub_customer (
 TABLE customer (id, company_name,
  city )
)
```

只发布表中的某些行

发布定义中未指定任何 WHERE 子句时，将上载发布中所有更改过的行。可在发布中的项目内添加 WHERE 子句，对要上载的行做以下限制：进行过更改并满足 WHERE 子句中的搜索条件。

WHERE 子句中的搜索条件只能引用项目中包含的列。此外，还不能在 WHERE 子句中使用以下任何一项：

- 子查询
- 变量
- 非确定型函数

这些条件不会强制执行，但违反这些条件可能会导致意外的结果。任何与 WHERE 子句有关的错误都是在对 WHERE 子句引用的表运行 DML 时生成的，定义发布时并不会生成此类错误。

♦ **使用 WHERE 子句创建发布（Sybase Central [管理] 模式）**

1. 使用 SQL Anywhere 插件，以具有 DBA 权限的用户身份连接到远程数据库。
2. 打开 [**发布**] 文件夹。
3. 选择 [**文件**] » [**新建**] » [**发布**]。
4. 在 [**您要给新发布指定什么名称**] 字段中输入新发布的名称。单击 [**下一步**]。

5. 单击 [**下一步**]。
6. 在 [**可用表**] 列表中，选择一个表。单击 [**添加**]。
7. 单击 [**下一步**]。
8. 单击 [**下一步**]。
9. 在 [**项目**] 列表中选择一个表，并在 [**所选项目具有以下 WHERE 子句**] 窗格中输入搜索条件。
10. 单击 [**完成**]。

♦ **使用 WHERE 子句创建发布 (SQL)：**

1. 以具有 DBA 权限的用户身份连接到远程数据库。
2. 执行 CREATE PUBLICATION 语句，其中包括想要包含在发布中的表和一个 WHERE 条件。

   See “CREATE PUBLICATION 语句 [MobiLink] [SQL Remote]”一节 《SQL Anywhere 服务器 - SQL 参考》.

示例

以下示例创建一个发布，该发布包括整个雇员表和 SalesOrder 表中未被标记为归档的所有行。

```
CREATE PUBLICATION main_publication (
TABLE Employees,
TABLE SalesOrders
WHERE archived = 'N'
);
```

通过将该表中的归档列从任何其它值更改为 N，在下次同步时会向 MobiLink 服务器发送一个删除操作。相反，如果将归档列从 N 更改为任何其它值，则会发送一个插入操作。对归档列的更新*不*发送到 MobiLink 服务器。

仅下载发布

可以创建仅下载数据到远程数据库而从不上载数据的发布。仅下载发布不使用客户端上的事务日志。

仅下载方法之间的差异

指定只进行下载（不进行上载）有两种方法：

- **仅下载同步** 使用 dbmlsync 选项 -e DownloadOnly 或 -ds。
- **仅下载发布** 使用 FOR DOWNLOAD ONLY 关键字创建发布。

这两种方法很不一样：

仅下载同步	仅下载发布
如果下载试图更改远程数据库上已修改但尚未上载的行，下载便会失败。	下载可以覆盖远程数据库上已修改但尚未上载的行。
使用可以上载和/或下载的常规发布。仅下载同步使用 dbmlsync 命令行选项或扩展选项进行指定。	使用仅下载发布。这些发布中的所有同步均为仅下载同步。无法使常规发布变为仅下载发布。
需要日志文件。	不需要日志文件。
长时间不上载这些预订时不会截断日志文件，因此日志文件会消耗相当大的存储空间。	如果有日志文件，同步不会影响同步截断点。这意味着，即使长时间不同步发布，也仍然可以截断日志文件。仅下载发布不影响日志文件截断。
偶尔需要进行上载，以减少仅下载同步扫描的日志量。否则，完成仅下载同步所需的时间将越来越长。	从不需要进行上载。

另请参见

- “CREATE PUBLICATION 语句 [MobiLink] [SQL Remote]”一节 《SQL Anywhere 服务器 - SQL 参考》
- “仅上载同步和仅下载同步”一节 《MobiLink - 服务器管理》

变更现有发布

在创建了发布之后，可以通过添加、修改或删除项目或重命名发布对发布进行变更。如果修改了某个项目，必须输入所修改项目的完整说明。

您可以使用 Sybase Central 或通过 ALTER PUBLICATION 语句执行这些任务。

注意

- 只有 DBA 或发布的所有者才可以变更发布。
- 请谨慎行事。在运行的 MobiLink 设置中变更发布可能会导致错误和数据丢失。要变更的发布只要有预订，就必须将此更改作为模式升级来对待。请参见“远程客户端中的模式更改” 第 75 页。

♦ **修改现有发布或项目的属性（Sybase Central [管理] 模式）**

1. 以发布的所有者或具有 DBA 权限的用户身份连接到远程数据库。
2. 单击左窗格中的发布或项目。右窗格中将出现属性。
3. 配置属性。

♦ **添加项目（Sybase Central [管理] 模式）**

1. 使用 SQL Anywhere 插件，以发布的所有者或具有 DBA 权限的用户身份连接到远程数据库。
2. 展开 [**发布**] 文件夹。
3. 选择一个发布。
4. 选择 [**文件**] » [**新建**] » [**项目**]。
5. 在 [**创建项目向导**] 中，执行以下操作：
   - 在 [**您要将哪张表用于此项目**] 列表中，选择一个表。单击 [**下一步**]。
   - 单击 [**所选列**] 并选择列。单击 [**下一步**]。
   - 在 [**您可以为此项目指定 WHERE 子句**] 窗格中，输入可选 WHERE 子句。单击 [**完成**]。

♦ **删除项目（Sybase Central [管理] 模式）**

1. 使用 SQL Anywhere 插件，以发布的所有者或具有 DBA 权限的用户身份连接到数据库。
2. 展开 [**发布**] 文件夹。
3. 右击发布，然后选择 [**删除**]。
4. 单击 [**是**]。

♦ **修改现有发布 (SQL)：**

1. 以发布的所有者或具有 DBA 权限的用户身份连接到远程数据库。
2. 执行 ALTER PUBLICATION 语句。

   请参见“ALTER PUBLICATION 语句 [MobiLink] [SQL Remote]”一节 《SQL Anywhere 服务器 - SQL 参考》。

示例

- 以下语句将 customer 表添加到 pub_contact 发布中。

```
ALTER PUBLICATION pub_contact (
 ADD TABLE customer
)
```

另请参见“ALTER PUBLICATION 语句 [MobiLink] [SQL Remote]”一节 《SQL Anywhere 服务器 - SQL 参考》。

删除发布

您可以使用 Sybase Central 或 DROP PUBLICATION 语句删除发布。删除发布之前，必须先删除与其关联的所有预订。

若要删除发布，您必须具有 DBA 权限。

- **删除发布（Sybase Central [管理] 模式）**

1. 使用 SQL Anywhere 插件，以具有 DBA 权限的用户身份连接到远程数据库。
2. 打开 [**发布**] 文件夹。
3. 右击发布，然后选择 [**删除**]。

- **删除发布 (SQL)：**

1. 以具有 DBA 权限的用户身份连接到远程数据库。
2. 执行 DROP PUBLICATION 语句。

   请参见“DROP PUBLICATION 语句 [MobiLink] [SQL Remote]”一节 《SQL Anywhere 服务器 - SQL 参考》。

示例

以下语句删除名为 pub_orders 的发布。

```
DROP PUBLICATION pub_orders
```


创建 MobiLink 用户

连接到 MobiLink 服务器时使用 MobiLink 用户名进行验证。必须在远程数据库中创建 MobiLink 用户，然后在统一数据库中注册这些用户。

MobiLink 用户与数据库用户不同。所创建的 MobiLink 用户名可以与数据库用户名相同，但这种名称上的一致既不会影响 MobiLink，也不会影响 SQL Anywhere。

♦ **将 MobiLink 用户添加到远程数据库（Sybase Central [管理] 模式）**

1. 通过 SQL Anywhere 插件以具有 DBA 权限的用户身份连接到数据库。
2. 单击 [**MobiLink 用户**] 文件夹。
3. 选择 [**文件**] » [**新建**] » [**MobiLink 用户**]。
4. 在 [**您要给新用户指定什么名称**] 字段中输入 MobiLink 用户的名称。
5. 单击 [**完成**]。

♦ **将 MobiLink 用户添加到远程数据库 (SQL)：**

1. 以具有 DBA 权限的用户身份连接到数据库。
2. 执行 CREATE SYNCHRONIZATION USER 语句。MobiLink 用户名对远程数据库进行唯一标识，因此在同步系统内必须唯一。

   下例将添加名为 SSinger 的 MobiLink 用户：

   ```
   CREATE SYNCHRONIZATION USER SSinger
   ```

   可以在 CREATE SYNCHRONIZATION USER 语句中指定 MobiLink 用户的属性，也可以另外使用 ALTER SYNCHRONIZATION USER 语句进行指定。

   有关详细信息，请参见“CREATE SYNCHRONIZATION USER 语句 [MobiLink]”一节 《SQL Anywhere 服务器 - SQL 参考》。

   有关设置 MobiLink 用户属性（包括口令）的信息，请参见“存储 MobiLink 用户的扩展选项”一节第 101 页。

有关注册 MobiLink 用户的信息，请参见“向统一数据库添加 MobiLink 用户名”一节第 10 页。

存储 MobiLink 用户的扩展选项

通过使用扩展选项，可在远程数据库中为每个 MobiLink 用户指定选项。扩展选项可在命令行中指定、存储在数据库中或使用 sp_hook_dbmlsync_set_extended_options 事件挂接来指定。

有关扩展选项的列表，请参见“MobiLink SQL Anywhere 客户端扩展选项”第 175 页。

♦ **在数据库中存储 MobiLink 扩展选项（Sybase Central [管理] 模式）**

1. 通过 SQL Anywhere 插件以具有 DBA 权限的用户身份连接到数据库。
2. 打开 [**MobiLink 用户**] 文件夹。
3. 右击该 MobiLink 用户名，然后选择 [**属性**]。
4. 按需要更改其属性。

♦ **在数据库 (SQL) 中存储 MobiLink 扩展选项**

1. 以具有 DBA 权限的用户身份连接到数据库。
2. 执行 ALTER SYNCHRONIZATION USER 语句。

   下例将名为 SSinger 的 MobiLink 用户的扩展选项更改为其缺省值：

   ```
   ALTER SYNCHRONIZATION USER SSinger
   DELETE ALL OPTION
   ```

   有关详细信息，请参见“ALTER SYNCHRONIZATION USER 语句 [MobiLink]”一节 《SQL Anywhere 服务器 - SQL 参考》。

   也可以在创建 MobiLink 用户名时指定属性。

   有关详细信息，请参见“CREATE SYNCHRONIZATION USER 语句 [MobiLink]”一节 《SQL Anywhere 服务器 - SQL 参考》。

♦ **使用客户端事件挂接指定 MobiLink 用户属性**

- 可以采用编程方式自定义即将发生的同步的行为。

  有关详细信息，请参见“sp_hook_dbmlsync_set_extended_options”一节第 278 页。

另请参见

- “使用 dbmlsync 扩展选项”一节第 106 页

删除 MobiLink 用户

在从远程数据库中删除 MobiLink 用户之前，必须首先删除该用户的所有预订。

♦ **从远程数据库中删除 MobiLink 用户（Sybase Central [管理] 模式）**

1. 通过 SQL Anywhere 插件以具有 DBA 权限的用户身份连接到数据库。
2. 在 [**MobiLink 用户**] 文件夹中找到该 MobiLink 用户。
3. 右击该 MobiLink 用户，然后选择 [**删除**]。

♦ **从远程数据库中删除 MobiLink 用户 (SQL)：**

1. 以具有 DBA 权限的用户身份连接到数据库。
2. 执行 DROP SYNCHRONIZATION USER 语句。

   下例将从该数据库中删除名为 SSinger 的 MobiLink 用户：

   ```
   DROP SYNCHRONIZATION USER SSinger
   ```

   有关详细信息，请参见“DROP SYNCHRONIZATION USER 语句 [MobiLink]”一节 《SQL Anywhere 服务器 - SQL 参考》。

创建同步预订

创建 MobiLink 用户和发布后，必须为至少一个 MobiLink 用户预订一个或多个预先存在的发布。通过创建同步预订来实现此目的。

有关创建发布的信息，请参见“发布数据”一节第 94 页。有关创建 MobiLink 用户的信息，请参见“创建 MobiLink 用户”一节第 101 页。

注意
您必须确保 MobiLink 用户的所有预订仅与一个统一数据库同步。否则，您可能会遇到数据丢失和不可预知的行为。

同步预订会将特定 MobiLink 用户与发布相链接。它还可能包含同步所需要的其它信息。例如，您可以指定 MobiLink 服务器的地址以及同步预订的选项。特定于同步预订的值将替换为 MobiLink 用户设置的值。

只有在 MobiLink SQL Anywhere 远程数据库中才需要同步预订。服务器逻辑通过同步脚本得以实现，这些脚本存储在统一数据库的 MobiLink 系统表中。

单个 SQL Anywhere 数据库可以与多个 MobiLink 服务器进行同步。若要实现与多个服务器进行同步，请为每个服务器创建不同的 MobiLink 用户。

请参见“CREATE SYNCHRONIZATION SUBSCRIPTION 语句 [MobiLink]”一节 《SQL Anywhere 服务器 - SQL 参考》。

示例

若要同步 SQL Anywhere 示例数据库中的 customer 表与 sales_order 表，可以使用以下语句。

1. 首先，发布 customer 表和 sales_order 表。将该发布命名为 testpub。

```
CREATE PUBLICATION testpub
 (TABLE customer, TABLE sales_order)
```

2. 接下来，创建 MobiLink 用户。在本例中，MobiLink 用户为 demo_ml_user。

```
CREATE SYNCHRONIZATION USER demo_ml_user
```

3. 最后，创建预订以将用户与发布相链接。

```
CREATE SYNCHRONIZATION SUBSCRIPTION TO testpub
 FOR demo_ml_user
 TYPE tcpip
 ADDRESS 'host=localhost;port=2439;'
 OPTION sv='version1'
```

变更 MobiLink 预订

可以使用 Sybase Central 或 ALTER SYNCHRONIZATION SUBSCRIPTION 语句变更同步预订。该语句在语法上与 CREATE SYNCHRONIZATION SUBSCRIPTION 语句类似，但还提供了一项扩充，可利用它更方便地添加、修改及删除选项。

♦ **变更同步预订（Sybase Central [管理] 模式）**

1. 以具有 DBA 权限的用户身份连接到数据库。
2. 打开 [**MobiLink 用户**] 文件夹。
3. 单击一个用户。右窗格中将出现属性。
4. 在右窗格中，单击 [**预订**] 选项卡。右击要更改的预订，然后选择 [**属性**]。
5. 按需要更改其属性。

♦ **变更同步预订 (SQL)：**

1. 以具有 DBA 权限的用户身份连接到数据库。
2. 执行 ALTER SYNCHRONIZATION SUBSCRIPTION 语句。

请参见“ALTER SYNCHRONIZATION USER 语句 [MobiLink]”一节 《SQL Anywhere 服务器 - SQL 参考》。

删除 MobiLink 预订

可以使用 Sybase Central 或 DROP SYNCHRONIZATION SUBSCRIPTION 语句删除同步预订。

删除同步预订要求具有 DBA 权限。

♦ **删除同步预订（Sybase Central [管理] 模式）**

1. 以具有 DBA 权限的用户身份连接到数据库。
2. 打开 [**MobiLink 用户**] 文件夹。
3. 选择一个 MobiLink 用户。
4. 右击预订，然后选择 [**删除**]。

♦ **删除同步预订 (SQL)：**

1. 以具有 DBA 权限的用户身份连接到数据库。
2. 执行 DROP SYNCHRONIZATION SUBSCRIPTION 语句。

示例

下面的语句将删除 MobiLink 用户 jsmith 对名为 pub_orders 的发布的同步预订。

```
DROP SYNCHRONIZATION SUBSCRIPTION
FOR jsmith TO pub_orders
```

请参见“DROP SYNCHRONIZATION SUBSCRIPTION 语句 [MobiLink]”一节 《SQL Anywhere 服务器 - SQL 参考》。

启动同步

MobiLink 同步总是由客户端启动。如果是 SQL Anywhere 客户端，同步通过运行 dbmlsync 实用程序来启动。此实用程序会连接并同步 SQL Anywhere 远程数据库。

请参见“MobiLink SQL Anywhere 客户端实用程序 (dbmlsync)”第 121 页。

可以使用 -c 选项在 dbmlsync 命令行中指定连接参数。这些参数用于远程数据库。如果不指定连接参数，将出现一个连接窗口，要求提供缺少的连接参数及启动选项。

请参见“-c 选项”一节第 134 页。

客户端网络协议选项可以存储在远程数据库中的同步预订、发布或用户中，也可以在 dbmlsync 命令行中进行指定。这些选项用于查找相应的 MobiLink 服务器。

请参见“CommunicationAddress (adr) 扩展选项”一节第 179 页。

dbmlsync 权限

当 dbmlsync 连接到数据库时，它必须具有应用所做的全部更改的权限。dbmlsync 命令行包含用于此连接的口令。这可能会带来安全问题。

为避免出现安全问题，请为用户授予（非 DBA）REMOTE DBA 权限，然后在 dbmlsync 连接字符串中使用此用户 ID。具有 REMOTE DBA 权限的用户 ID 只有在通过 dbmlsync 实用程序建立连接时才有 DBA 权限。任何其它使用同一用户 ID 的连接都不会被授予特殊权限。

请参见“GRANT REMOTE DBA 语句 [MobiLink] [SQL Remote]”一节 《SQL Anywhere 服务器 - SQL 参考》。

自定义同步

请参见“自定义 dbmlsync 同步”一节第 116 页。

使用 dbmlsync 扩展选项

MobiLink 提供了几个用于自定义同步过程的扩展选项。扩展选项可以在发布、用户或预订中设置。此外，还可以在 dbmlsync 命令行中使用选项来覆盖扩展选项值。

有关扩展选项的完整列表，请参见“MobiLink SQL Anywhere 客户端扩展选项”第 175 页。

♦ **在 dbmlsync 命令行中替换扩展选项：**

- 在 dbmlsync 的 -e 或 -eu dbmlsync 选项中以 *option-name=value* 形式提供扩展选项值。例如：

```
dbmlsync -e "v=on;sc=low"
```

♦ **为预订、发布或用户设置扩展选项：**

- 在 SQL Anywhere 远程数据库中为 CREATE SYNCHRONIZATION SUBSCRIPTION 语句或 CREATE SYNCHRONIZATION USER 语句添加该选项。

为发布添加扩展选项的方式稍有不同。若要为发布添加扩展选项，请使用 ALTER/CREATE SYNCHRONIZATION SUBSCRIPTION 语句，并省略 FOR 子句。

示例

以下语句创建一个同步预订，此预订使用扩展选项将用于准备上载的高速缓存大小设置为 3 MB，并将上载增量大小设置为 3 KB。

```
CREATE SYNCHRONIZATION SUBSCRIPTION TO my_pub
FOR ml_user
ADDRESS 'host=test.internal;port=2439;'
OPTION memory='3m',increment='3k'
```

请注意，选项值可用单引号括起来，但选项名仍然不能使用引号。

Dbmlsync 网络协议选项

Dbmlsync 连接信息包括用于与服务器进行通信的协议、MobiLink 服务器的地址及其它连接参数。

有关详细信息，请参见：

- “CommunicationType (ctp) 扩展选项”一节第 180 页
- “CommunicationAddress (adr) 扩展选项”一节第 179 页

事务日志文件

在大多数情况下，dbmlsync 通过使用 SQL Anywhere 事务日志来确定上载的内容。偏移表示一个时间点，截至该时间点，预订的所有操作都已上载并确认完毕。

缺省情况下，SQL Anywhere 数据库会保留事务日志。可以在创建数据库时确定事务日志的位置或是否保留事务日志。

如果实现了脚本式上载或只使用仅下载发布，则事务日志可能并非必需。

为准备上载，dbmlsync 实用程序需要访问自上次成功同步正在同步的 MobiLink 用户的所有预订以来写入的所有事务日志。不过，常规数据库维护过程中通常会截断并重命名 SQL Anywhere 日志文件。在此类情况下，必须将旧日志文件重命名并保存在单独的目录中，直到其描述的所有更改都已成功同步。

可以在 dbmlsync 命令行中指定包含重命名的日志文件的目录。如果工作日志文件自上次同步后未被截断并重命名，或您是从包含重命名的日志文件的目录中运行 dbmlsync，则可以忽略此参数。

另请参见

- “备份和数据恢复” 《SQL Anywhere 服务器 - 数据库管理》
- “进度偏移”一节第 92 页
- “事务日志”一节 《SQL Anywhere 服务器 - 数据库管理》
- “初始化实用程序 (dbinit)”一节 《SQL Anywhere 服务器 - 数据库管理》
- “脚本式上载”第 355 页

示例

假定旧日志文件存储在目录 *c:\oldlogs* 中。可以使用下列命令同步远程数据库。

```
dbmlsync -c "dbn=remote;uid=syncuser" c:\oldlogs
```

旧日志目录的路径必须为命令行中的最后一个参数。

同步中的并发

为确保同步的完整性，dbmlsync 必须确保在构建上载到应用下载之间的这段时间内下载中的任何行都未被修改。

在除 Windows Mobile 以外的所有平台上，缺省情况下 dbmlsync 会在正在同步的任何发布中涉及的所有表上获得共享锁。在 Windows Mobile 上，缺省情况下 dbmlsync 获得独占锁。Dbmlsync 会在构建上载之前获得锁，并持有该锁直到应用下载之时。

有关锁的详细信息，请参见“行锁”一节 《SQL Anywhere 服务器 - SQL 的用法》。

使用以下选项可自定义此锁定行为：

- -d 选项
- LockTables 选项

-d 选项

使用锁定机制时，如果存在到数据库的其它连接，则只要这些连接在同步表上有锁，同步就会失败。如果要确保同步即使在存在其它锁时仍立即进行，请使用 dbmlsync -d 选项。指定此选项时，数据库将删除任何会干扰同步的带锁连接，以便同步能够进行。被删除的连接上的未提交更改将被回退。

有关详细信息，请参见“-d 选项”一节第 135 页。

LockTables 选项

保护数据完整性的一个替代方法是将扩展选项 LockTables 设置为 OFF，这可以防止项目的表被锁定。这会使 dbmlsync 跟踪构建上载后修改过的所有行。当接收下载时，如果下载中有任何行被修改，则不应用该下载。Dbmlsync 随后将重新尝试该同步。如果没有检测到新的下载冲突，重新尝试就会成功。

有关详细信息，请参见“LockTables (lt) 扩展选项”一节第 194 页。

如果检测到冲突，将会取消下载阶段并回退下载操作，以避免将新更改覆盖。然后 dbmlsync 实用程序将重新尝试包含上载步骤的同步。这次进行同步尝试时，因为同步过程开始时该行就已存在，所以上载会包含该行，该行也就不会丢失。

缺省情况下，dbmlsync 将一直重新尝试同步，直到成功为止。您可以使用扩展选项 ConflictRetries 来限制重新尝试的次数。将 ConflictRetries 设置为 -1 将使 dbmlsync 一直重新尝试，直到同步成功。将该选项设置为非负整数时，将使 dbmlsync 最多只重新尝试所指定的次数。

有关详细信息，请参见“ConflictRetries (cr) 扩展选项”一节第 181 页。

从应用程序中启动同步

您可能希望在应用程序中加入 dbmlsync 的功能，从而不必单独给远程用户提供可执行文件。

有两种方法可实现这一点：

- 使用 Dbmlsync 集成组件。

  有关详细信息，请参见“Dbmlsync 集成组件（不建议使用）”第 319 页。

- 如果开发时所使用的语言可以调用 DLL，则可通过 DBTools 接口访问 dbmlsync。如果是使用 C 或 C++ 进行编程，则可以加入 *dbtools.h* 头文件，该文件位于 SQL Anywhere 11 目录的 *SDK\Include* 子目录中。此文件包含对 a_sync_db 结构和 DBSynchronizeLog 函数的描述，您可以用它们为您的应用程序添加此功能。此方法适用于所支持的全部平台，其中包括 Windows 和 Unix。

  有关详细信息，请参见：

  - “dbmlsync 的 DBTools 接口”第 347 页
  - “DBSynchronizeLog 函数”一节 《SQL Anywhere 服务器 - 编程》
  - “a_sync_db 结构”一节 《SQL Anywhere 服务器 - 编程》

使用 ActiveSync 同步

ActiveSync 是用于 Microsoft Windows Mobile 手持式设备的同步软件。ActiveSync 用于控制 Windows Mobile 设备与台式计算机之间的同步。MobiLink ActiveSync 提供程序控制到 MobiLink 服务器的同步。

为 SQL Anywhere 客户端设置 ActiveSync 同步包括以下步骤：

- 为进行 ActiveSync 同步而配置 SQL Anywhere 远程数据库。

  请参见“为 ActiveSync 配置 SQL Anywhere 远程数据库”一节第 110 页。
- 为 ActiveSync 安装 MobiLink 提供程序。

  请参见“安装 ActiveSync 的 MobiLink 提供程序”一节第 111 页。
- 注册 SQL Anywhere 客户端，以与 ActiveSync 配合使用。

  请参见“为 ActiveSync 注册 SQL Anywhere 客户端”一节第 112 页。

如果使用 ActiveSync 同步，则必须从 ActiveSync 软件启动同步。MobiLink ActiveSync 提供程序可以启动 dbmlsync，也可以唤醒根据调度字符串的安排处于休眠状态的 dbmlsync。

您也可以使用远程数据库中的延迟挂接将 dbmlsync 转为休眠模式，但 ActiveSync 的 MobiLink 提供程序无法从此状态中调用同步。

有关调度同步的信息，请参见“调度同步”一节第 114 页。

为 ActiveSync 配置 SQL Anywhere 远程数据库

♦ 为 ActiveSync 配置 SQL Anywhere 远程数据库

1. 选择同步类型（TCP/IP、TLS、HTTP 或 HTTPS）。

   可以为同步发布、同步用户或同步预订设置同步类型，并且设置方式类似。下面是典型 CREATE SYNCHRONIZATION USER 语句的一部分：

   ```
   CREATE SYNCHRONIZATION USER SSinger
   TYPE tcpip
   ...
   ```

2. 提供 address 子句以指定 MobiLink ActiveSync 提供程序与 MobiLink 服务器之间的通信。

   如果是 HTTP 或 TCP/IP 同步，CREATE SYNCHRONIZATION USER 或 CREATE SYNCHRONIZATION SUBSCRIPTION 语句的 ADDRESS 子句指定 MobiLink 客户端与服务器之间的通信。对于 ActiveSync，通信分以下两个阶段进行：第一阶段是设备上的 dbmlsync 实用程序与台式计算机上的 MobiLink ActiveSync 提供程序通信，第二阶段是台式计算机与 MobiLink 服务器通信。ADDRESS 子句指定 MobiLink ActiveSync 提供程序与 MobiLink 服务器之间的通信。

   以下语句指定与名为 kangaroo 的计算机上的 MobiLink 服务器进行 TCP/IP 通信：


```
CREATE SYNCHRONIZATION USER SSinger
TYPE tcpip
ADDRESS 'host=kangaroo;port=2439'
```

有关详细信息，请参见“CREATE SYNCHRONIZATION USER 语句 [MobiLink]”一节 《SQL Anywhere 服务器 - SQL 参考》。

安装 ActiveSync 的 MobiLink 提供程序

注册与 ActiveSync 配合使用的 SQL Anywhere MobiLink 客户端之前，必须使用安装实用程序 (*mlasinst.exe*) 安装 MobiLink ActiveSync 提供程序。

SQL Anywhere for Windows Mobile 安装程序会安装 MobiLink ActiveSync 提供程序。如果安装了 SQL Anywhere for Windows Mobile，则无需执行本节中的步骤。

如果已经安装了 MobiLink ActiveSync 提供程序，则必须分别注册每个应用程序。有关说明，请参见“为 ActiveSync 注册 SQL Anywhere 客户端”一节第 112 页。

♦ 为 ActiveSync 安装 MobiLink 提供程序：

1. 确保计算机上已安装了 ActiveSync 软件，而且已连接了 Windows Mobile 设备。
2. 运行以下命令来安装 MobiLink 提供程序：

   ```
   mlasinst -k desk-path -v dev-path
   ```

   其中 *desk-path* 是提供程序台式机组件 (*mlasdesk.dll*) 的位置，*dev-path* 是提供程序设备组件 (*mlasdev.dll*) 的位置。

   如果计算机上已安装 SQL Anywhere，则 *mlasdesk.dll* 位于 *install-dir\bin32* 中；*mlasdev.dll* 位于 *install-dir\CE* 中。如果省略 -v 或 -k，则缺省情况下将搜索这些目录。

   如果收到一条消息，指示无法打开远程提供程序，请软重启该设备，并重复该命令：

   有关详细信息，请参见“ActiveSync 提供程序的安装实用程序 (mlasinst)”一节第 27 页。
3. 重新启动计算机。

   ActiveSync 在计算机重新启动之后才能识别新的提供程序。
4. 启用 MobiLink 提供程序。

   对于 Vista 之前的 Windows 版本：

   - 在 [ActiveSync] 窗口中，单击 [**选项**]。
   - 选中列表中的 MobiLink 项，并单击 [**确定**] 以激活该提供程序。
   - 若要查看已注册应用程序的列表，请再次单击 [**选项**]，选择该 MobiLink 提供程序，并单击 [**设置**]。

     有关注册应用程序的详细信息，请参见“为 ActiveSync 注册 SQL Anywhere 客户端”一节第 112 页。

   对于 Windows Vista：

- 从 [Windows 移动设备中心] 窗口，单击 [**移动设备设置**]，然后单击 [**更改内容设置**]。
- 选择 [**MobiLink 客户端**]，并单击 [**保存**] 以激活该提供程序。
- 要查看已注册应用程序的列表，请单击 [**更改内容设置**]，再单击 [**MobiLink 客户端**]；然后单击 [**同步设置**]。

为 ActiveSync 注册 SQL Anywhere 客户端

注册应用程序以与 ActiveSync 配合使用无外乎两种方法：一种是使用 ActiveSync 提供程序安装实用程序进行注册，一种是使用 ActiveSync 软件本身进行注册。本节介绍如何使用 ActiveSync 软件。

有关替代方法的信息，请参见“ActiveSync 提供程序的安装实用程序 (mlasinst)”一节第 27 页。

♦ 注册 SQL Anywhere 客户端以与 ActiveSync 配合使用

1. 确保 ActiveSync 的 MobiLink 提供程序已安装。

   有关信息，请参见“安装 ActiveSync 的 MobiLink 提供程序”一节第 111 页。
2. 启动台式计算机上的 ActiveSync 软件。
3. 对于 Vista 之前的 Windows 版本：
   - 从 [**ActiveSync**] 窗口，选择 [**选项**]。
   - 从信息类型列表中，选择 [**MobiLink**] 并单击 [**设置**]。
   - 在 [**MobiLink 同步**] 窗口中，单击 [**新建**]。

   对于 Windows Vista：
   - 从 [**Windows 移动设备中心**] 窗口，单击 [**移动设备设置**]，然后单击 [**更改内容设置**]。
   - 单击 [**更改内容设置**]。
   - 单击 [**MobiLink 客户端**]。
   - 单击 [**同步设置**]。
4. 为应用程序输入下列信息：
   - **应用程序名**　标识该应用程序的名称将显示于 ActiveSync 用户接口中。
   - **类名**　dbmlsync 客户端的类名称，例如使用 -wc 选项设置的类名称。

     有关详细信息，请参见“-wc 选项”一节第 172 页。
   - **路径**　设备上 dbmlsync 应用程序的位置。
   - **参数**　在 ActiveSync 启动 dbmlsync 时使用的任何命令行参数。

     可以使用以下两种模式之一启动 dbmlsync：
     - 如果指定调度选项，dbmlsync 会进入悬停模式。在这种情况下，请使用 dbmlsync -wc 选项，类名设置中有该选项的匹配值。

       有关详细信息，请参见“-wc 选项”一节第 172 页和“调度同步”一节第 114 页。

- 否则，dbmlsync 将不会处于悬停模式。在这种情况下，请使用 -k 关闭 dbmlsync。
  有关详细信息，请参见“-k 选项（不建议使用）”一节第 147 页。

5. 单击 [**确定**] 注册该应用程序。

调度同步

可以将 dbmlsync 设置为根据定义的规则定期进行同步。可通过以下两种方法进行此项设置：

- 使用 dbmlsync 扩展选项 SCHEDULE 在一天或一周的特定时间或按固定间隔启动同步。在这种情况下，dbmlsync 将一直运行到用户将其停止为止。

  请参见“使用 dbmlsync 选项设置调度”一节第 114 页。

- 使用 dbmlsync 事件挂接根据所定义的逻辑启动同步。对于按不固定间隔实现同步或为响应事件而实现同步而言，这是最好的方法。在这种情况下，可以通过挂接代码以编程方式停止 dbmlsync。

  请参见“使用事件挂接启动同步”一节第 115 页。

悬停

如果指定了调度选项或挂接，dbmlsync 会进入悬停模式。悬停功能可减少花费在日志扫描上的时间。通过设置 dbmlsync 扩展选项 HoverRescanThreshold 或使用 dbmlsync 存储过程 sp_hook_dbmlsync_log_rescan，可以增强悬停的性能优势。

有关详细信息，请参见：

- “HoverRescanThreshold (hrt) 扩展选项”一节第 190 页
- “sp_hook_dbmlsync_log_rescan”一节第 263 页

使用 dbmlsync 选项设置调度

可以不按批处理方式运行 dbmlsync，即进行同步后将 dbmlsync 关闭，而是对 SQL Anywhere 客户端进行设置，使 dbmlsync 持续运行并在预定的时间进行同步。

可以将同步调度指定为扩展选项。可以在 dbmlsync 命令行中指定它，也可以将它存储在同步用户、预订或发布的数据库中。

有关调度语法的详细信息，请参见“Schedule (sch) 扩展选项”一节第 202 页。

有关扩展选项的详细信息，请参见：

- “MobiLink SQL Anywhere 客户端扩展选项”第 175 页
- “-eu 选项”一节第 145 页

♦ 将调度添加到同步预订：

- 在同步预订中设置调度扩展选项。例如，

```
CREATE SYNCHRONIZATION SUBSCRIPTION TO mypub
FOR mluser
ADDRESS 'host=localhost'
OPTION schedule='weekday@11:30am-12:30pm'
```

  可以使用 dbmlsync -is 选项替换调度并立即进行同步。-is 选项指示 dbmlsync 忽略使用调度扩展选项指定的调度。有关详细信息，请参见“-is 选项”一节第 146 页。

♦ **从 dbmlsync 命令行添加调度：**

● 设置调度扩展选项。扩展选项是使用 -e 或 -eu 设置的。例如，

```
dbmlsync -e "sch=weekday@11:30am-12:30pm" ...
```

如果在两个位置中的任一位置指定了已调度的同步，同步后 dbmlsync 将不关闭，而是继续运行。

使用事件挂接启动同步

可以实现 dbmlsync 事件挂接来控制同步的执行时间。

通过 sp_hook_dbmlsync_end 挂接，可以使用 #hook_dict 表中的 Restart 行在每个同步结束时决定 dbmlsync 是否需要重复执行同步。

有关详细信息，请参见“sp_hook_dbmlsync_end”一节第 261 页。

通过 sp_hook_dbmlsync_delay 挂接，可以在每个同步开始时刻前创建一个延迟，以便选择开始执行同步的时间。通过此挂接，可以设置固定时间长度的延迟，或定期进行轮询以等待某种条件得到满足。

有关详细信息，请参见“sp_hook_dbmlsync_delay”一节第 241 页。

自定义 dbmlsync 同步

dbmlsync 客户端事件挂接

通过事件挂接，可以使用 SQL 存储过程来管理 dbmlsync 的客户端同步过程。可以将客户端事件挂接与 dbmlsync 命令行实用程序或 dbmlsync 编程接口配合使用。

可使用事件挂接记录和处理同步事件。例如，可根据逻辑事件调度同步、重试连接故障或处理特定错误和参照完整性违规。

有关客户端事件挂接的详细信息，请参见“SQL Anywhere 客户端的事件挂接”第 225 页。

dbmlsync 编程接口

可以使用以下编程接口将 MobiLink 客户端集成到应用程序中并启动同步。这些接口可以替代 dbmlsync 命令行实用程序的功能。

- **dbmlsync API** Dbmlsync API 提供了一个编程接口，允许使用 C++ 或 .Net 编写的 MobiLink 客户端启动同步并接收有关这些客户端所请求的同步进度的反馈。这种新的编程接口便于您访问同步结果的更多信息，您也可排列同步，使其更易于管理。

  请参见“Dbmlsync API 简介”一节第 292 页。

- **dbmlsync 的 DBTools 接口** 可以使用 dbmlsync 的 DBTools 接口将同步功能集成到 SQL Anywhere 同步客户端应用程序中。SQL Anywhere 的所有数据库管理实用程序都是在 DBTools 上构建的。

  请参见“dbmlsync 的 DBTools 接口”第 347 页。

脚本式上载

也可覆盖所使用的客户端事务日志，并定义您自己的上载流。请参见“脚本式上载”第 355 页。

SQL Anywhere 客户端记录

创建与 SQL Anywhere 远程数据库配合使用的 MobiLink 应用程序时，需要知道两种类型的客户端日志文件：

- dbmlsync 消息日志
- SQL Anywhere 事务日志

dbmlsync 消息日志

缺省情况下，dbmlsync 消息发送到 dbmlsync 消息窗口。此外，还可以使用 -o 或 -ot 选项将输出发送到控制台日志文件。以下是命令行的一部分，它将输出发送到名为 *dbmlsync.log* 的日志文件。

```
dbmlsync -o dbmlsync.log ...
```

记录 dbmlsync 活动在开发过程中和进行故障排除时特别有用。建议不对生产环境中的常规操作使用详细输出，因为这会降低性能。

可以控制日志文件的大小，并可指定当文件大小达到其最大值时所要执行的操作：

- 使用 -o 选项指定日志文件并将输出追加到该文件。
- 使用 -ot 选项指定日志文件，但在向该文件追加输出之前删除其内容。
- 除了 -o 或 -ot 之外，还可以使用 -os 选项来指定一个大小，日志文件达到该大小时会被重命名，并使用其原来名称启动新日志文件。

有关详细信息，请参见：

- “-o 选项”一节第 152 页
- “-ot 选项”一节第 154 页
- “-os 选项”一节第 153 页

可以使用 -v 选项控制记录到消息日志文件中的信息和显示在 dbmlsync 消息窗口中的信息。

有关详细信息，请参见“-v 选项”一节第 171 页。

可以使用 delete_old_logs 选项来管理日志文件。

有关详细信息，请参见“delete_old_logs 选项 [MobiLink 客户端] [SQL Remote] [复制代理]”一节《SQL Anywhere 服务器 - 数据库管理》。

如果未指定消息日志文件，则所有输出都显示在 dbmlsync 消息窗口中。如果指定了消息日志文件，则发送到 dbmlsync 消息窗口的输出就会减少。

SQL Anywhere 事务日志

请参见“事务日志文件”一节第 107 页。

在 Mac OS X 上运行 MobiLink

可以在 Mac OS X 上运行 MobiLink 服务器和 SQL Anywhere MobiLink 客户端。不能在 Mac OS X 上运行 UltraLite。

为了在 Mac OS X 上同步 MobiLink 统一数据库，您可以将 SQL Anywhere ODBC 驱动程序用作驱动程序管理器。请参见“在 Mac OS X 上创建 ODBC 数据源”一节 《SQL Anywhere 服务器 - 数据库管理》。

♦ 在 Mac OS X 上启动 MobiLink 服务器

1. 启动 SyncConsole。

   在 Finder 中，双击 [SyncConsole]。SyncConsole 应用程序位于 */Applications/SQLAnywhere11* 中。

2. 选择 [**文件**] » [**新建**] » [**MobiLink 服务器**]。

3. 配置 MobiLink 服务器：

   a. 在 [**连接参数**] 字段中，输入以下字符串：

      ```
      dsn=dsn-name
      ```

      dsn-name 是 SQL Anywhere ODBC 数据源名称。有关创建 ODBC 数据源的信息，请参见“在 Unix 和 Mac OS X 上设置环境变量”一节 《SQL Anywhere 服务器 - 数据库管理》。

      如果 *dsn-name* 包含空格，则使用双引号将字符串引起来。例如：

      ```
      dsn="SQL Anywhere 11 Demo"
      ```

   b. 在 [**选项**] 字段中设置选项（如果需要）。

      [**选项**] 字段允许您控制 MobiLink 服务器行为的许多方面。有关选项的完整列表，请参见“mlsrv11 语法”一节 《MobiLink - 服务器管理》。

4. 单击 [**启动**] 以启动 MobiLink 服务器。

   将会出现数据库服务器消息窗口并显示消息，表示服务器已准备就绪可以接受同步请求了。

♦ 在 Mac OS X 上启动 dbmlsync

1. 启动 SyncConsole。

   在 [**Finder**] 中，双击 [**SyncConsole**]。SyncConsole 应用程序位于 */Applications/SQLAnywhere11* 中。

2. 选择 [**文件**] » [**新建**] » [**MobiLink 客户端**]。

   将出现客户端选项窗口。选项窗口具有许多配置选项，这些配置选项对应于 dbmlsync 命令行选项。有关完整列表，请参见“dbmlsync 语法”一节第 123 页。

   [**登录**]、[**数据库**]、[**网络**] 和 [**高级**] 选项卡上的选项均用于定义从 MobiLink 客户端到 SQL Anywhere 远程数据库的连接。通常只需在 [**登录**] 选项卡上指定 ODBC 数据源即可进行连接。

   [**DBMLSync**] 选项卡上的选项用于定义到 MobiLink 服务器的连接的各个方面。如果在远程数据库发布和预订中定义了这些功能，则可将此选项卡上的选项留空。

♦ 在 Mac OS X 上运行示例数据库

1. 指定 *sa_config* 配置脚本的来源。

   有关详细信息，请参见“在 Unix 和 Mac OS X 上设置环境变量”一节 《SQL Anywhere 服务器 - 数据库管理》。

2. 设置 ODBC 数据源。例如：

   ```
   dbdsn -w  "SQL Anywhere 11 Demo"
   -c "uid=DBA;pwd=sql;dbf=/Applications/SQLAnywhere11/System/demo.db"
   ```

3. 运行 MobiLink 服务器。例如：

   ```
   mlsrv11 -c "dsn=SQL Anywhere 11 Demo"
   ```

版本的注意事项

要使 dbmlsync 正常运行，dbmlsync.exe 的主要版本和次要版本都必须与数据库服务器的版本相匹配。此外，数据库文件的主要版本必须与 dbmlsync.exe 的主要版本匹配，并且数据库文件的次要版本必须等同于或低于 *dbmlsync.exe* 的次要版本。数据库文件的版本为其升级后的最新版本

例如，9.0.2 版本的 dbmlsync 只能与 9.0.2 版本的数据库服务器 (*dbeng9.exe*) 一起使用，但是它可以与 9.00、9.01 和 9.02 版本的数据库文件一起使用。

MobiLink SQL Anywhere 客户端实用程序 (dbmlsync)

目录

dbmlsync 语法

使用 dbmlsync 实用程序将 SQL Anywhere 远程数据库与统一数据库同步。

语法

dbmlsync [*options*] [*transaction-logs-directory*]

选项	说明
@data	从指定的环境变量或配置文件中读入选项。请参见“@data 选项”一节第 127 页。
-a	发生错误时不提示重新输入。请参见“-a 选项”一节第 128 页
-ap	指定验证参数。请参见“-ap 选项”一节第 129 页。
-ba *filename*	应用下载文件。请参见“-ba 选项”一节第 130 页。
-bc *filename*	创建一个下载文件。请参见“-bc 选项”一节第 131 页。
-be *string*	创建下载文件时，添加字符串。请参见“-be 选项”一节第 132 页。
-bg	创建下载文件时，使其适用于新的远程数据库。请参见“-bg 选项”一节第 133 页。
-c *connection-string*	提供数据库的连接参数，形式为 *parm1=value1*；*parm2=value2*，……如果您不提供此选项，则会出现一个窗口，要求您必须提供连接信息。请参见“-c 选项”一节第 134 页。
-d	删除数据库的所有其它连接，这些连接与要同步的项目产生了锁冲突。请参见“-d 选项”一节第 135 页
-dc	继续以前失败的下载。请参见“-dc 选项”一节第 136 页。
-dl	在 dbmlsync 消息窗口上显示日志消息。请参见“-dl 选项”一节第 137 页。
-do	禁用对脱机事务日志的扫描。请参见“-do 选项”一节第 138 页。
-drs *bytes*	对于可重新启动的下载，指定发生通信故障后需要重新发送的最大数据量。请参见“-drs 选项”一节第 139 页。
-ds	执行仅下载同步。请参见“-ds 选项”一节第 140 页。
-e "*option=value*"...	指定扩展选项。请参见“MobiLink SQL Anywhere 客户端扩展选项”第 175 页。

选项	说明
-eh	忽略挂接函数中出现的错误。
-ek *key*	指定加密密钥。请参见“-ek 选项”一节第 143 页。
-ep	提示输入加密密钥。请参见“-ep 选项”一节第 144 页。
-eu	为最新的 -n 选项所定义的上载指定扩展选项。请参见“-eu 选项”一节第 145 页。
-is	忽略调度。请参见“-is 选项”一节第 146 页。
-k	完成时关闭窗口。请参见“-k 选项（不建议使用）”一节第 147 页。
-l	列出可用的扩展选项。请参见“-l 选项”一节第 148 页。
-mn *password*	指定新的 MobiLink 口令。请参见“-mn 选项”一节第 149 页。
-mp *password*	指定 MobiLink 口令。请参见“-mp 选项”一节第 150 页。
-n *name*	指定同步发布名称。请参见“-n 选项”一节第 151 页。
-o *logfile*	将输出消息记录到此文件。请参见“-o 选项”一节第 152 页。
-os *size*	指定消息日志文件的最大大小，达到此大小时将重命名日志。请参见“-os 选项”一节第 153 页。
-ot *logfile*	删除消息日志文件内容，然后将输出消息记录到日志文件。请参见“-ot 选项”一节第 154 页。
-p	禁止日志扫描轮询。请参见“-p 选项”一节第 155 页。
-pc+	保持在多个同步之间与 MobiLink 服务器的连接处于打开状态。请参见“-pc 选项”一节第 156 页
-pd *dllname*;...	为 Windows Mobile 预装载指定的 DLL。请参见“-pd 选项”一节第 157 页。
-pi	测试您是否可以连接到 MobiLink。请参见“-pi 选项”一节第 158 页。
-pp *number*	设置日志扫描轮询周期。请参见“-pp 选项”一节第 159 页。
-q	以最小化窗口运行。请参见“-q 选项”一节第 160 页。
-qc	同步完成后关闭 dbmlsync。请参见“-qc 选项”一节第 161 页。

选项	说明
-r[**a** | **b**]	使用客户端进度值进行上载重试。请参见“-r 选项”一节第 162 页。
-sc	每次同步前重装模式信息。请参见“-sc 选项”一节第 163 页。
-sp *sync profile*	将同步配置文件中的选项添加到在命令行上指定的同步选项。请参见“-sp 选项”一节第 164 页。
-tu	执行事务性上载。请参见“-tu 选项”一节第 165 页。
-u *ml_username*	指定要同步的 MobiLink 用户。请参见“-u 选项”一节第 166 页。
-ui	对于使用 X 窗口的 Linux，如果没有提供可用的显示则以 shell 模式启动 dbmlsync。请参见“-ui 选项”一节第 167 页。
-uo	执行仅上载同步。请参见“-uo 选项”一节第 168 页。
-urc *row-estimate*	指定对要上载的行的估算行数。请参见“-urc 选项”一节第 169 页。
-ux	对于 Solaris 和 Linux，打开 dbmlsync 消息窗口。请参见“-ux 选项”一节第 170 页。
-v[*levels*]	详细操作。请参见“-v 选项”一节第 171 页。
-wc *classname*	指定窗口类名。请参见“-wc 选项”一节第 172 页。
-x	重新命名并重新启动事务日志。请参见“-x 选项”一节第 173 页。
transaction-logs-directory	指定事务日志的位置。请参见下面的事务日志文件。

注释

运行 dbmlsync 以将 SQL Anywhere 远程数据库与统一数据库同步。

为找到并连接到 MobiLink 服务器，dbmlsync 使用有关发布、同步用户、同步预订或命令行的信息。

事务日志文件 *transaction-logs-directory* 是包含 SQL Anywhere 远程数据库事务日志的目录。有一个活动事务日志和若干事务日志档案文件，dbmlsync 可能需要两者来确定上载内容。如果满足以下所有条件，则必须指定此参数：

- 自上次同步以来，已删除工作日志文件内容并已重命名工作日志文件
- 从存储重命名日志文件以外的目录运行 dbmlsync 实用程序

有关详细信息，请参见“事务日志文件”一节第 107 页。

dbmlsync 事件挂接 dbmlsync 客户端存储过程也可以帮助您自定义同步过程。有关详细信息，请参见“dbmlsync 挂接简介”一节第 227 页和“SQL Anywhere 客户端的事件挂接”第 225 页。

使用 dbmlsync 有关使用 dbmlsync 的详细信息，请参见“启动同步”一节第 106 页。

另请参见

- “启动同步”一节第 106 页
- “SQL Anywhere 客户端的事件挂接”第 225 页
- “Dbmlsync API”第 291 页
- “dbmlsync 的 DBTools 接口”第 347 页

@data 选项

读取来自指定的环境变量或配置文件的选项。

语法

dbmlsync @*data* ...

注释

使用此选项，可以将命令行选项放在环境变量或配置文件中。如果两者都存在并使用您指定的名称，则使用环境变量。

有关配置文件的详细信息，请参见“使用配置文件”一节 《SQL Anywhere 服务器 - 数据库管理》。

如果要保护口令或配置文件中的其它信息，可以使用文件隐藏实用程序对配置文件的内容进行模糊处理。

请参见“文件隐藏实用程序 (dbfhide)”一节 《SQL Anywhere 服务器 - 数据库管理》。

-a 选项

指定在发生错误时 dbmlsync 不应显示提示重新输入的窗口。

语法

dbmlsync -a ...

-ap 选项

将参数提供给 authenticate_parameters 脚本和验证参数。

语法

dbmlsync -ap "*parameters,...***"** ...

注释

在使用 authenticate_parameters 连接脚本或验证参数时使用。例如，

```
dbmlsync -ap "parm1,parm2,parm3"
```

这些参数将发送到 MobiLink 服务器，同时还传递到统一数据库上的 authenticate_parameters 脚本或其它事件。

另请参见

- “验证参数”一节 《MobiLink - 服务器管理》
- “authenticate_parameters 连接事件”一节 《MobiLink - 服务器管理》

-ba 选项

应用一个下载文件。

语法

dbmlsync -ba "*filename***"** ...

注释

指定将应用于远程数据库的现有下载文件的名称。您可以选择指定一个路径。如果未指定路径，则缺省位置为 dbmlsync 的启动目录。

另请参见

- “MobiLink 基于文件的下载” 《MobiLink - 服务器管理》
- “-bc 选项”一节第 131 页
- “-be 选项”一节第 132 页
- “-bg 选项”一节第 133 页

-bc 选项

创建一个下载文件。

语法

dbmlsync -bc "*filename***"** ...

注释

创建一个下载文件，并赋予指定的名称。下载文件应使用扩展名 .df。

您可以选择指定一个路径。如果未指定路径，缺省位置为 dbmlsync 的当前工作目录，即 dbmlsync 的启动目录。

或者，在创建下载文件的同一 dbmlsync 命令行中，可以使用 -be 选项指定一个可以在远程数据库中校验的字符串，使用 -bg 选项为新的远程数据库创建一个下载文件。

另请参见

- “MobiLink 基于文件的下载” 《MobiLink - 服务器管理》
- “-ba 选项”一节第 130 页
- “-be 选项”一节第 132 页
- “-bg 选项”一节第 133 页

-be 选项

创建下载文件时，此选项用于指定要包含在文件中的额外字符串。

语法

dbmlsync -bc "*filename***" -be "***string***"** ...

注释

该字符串可用于验证或其它目的。应用下载文件时，它将传递到远程数据库上的 sp_hook_dbmlsync_validate_download_file 存储过程。

另请参见

- “sp_hook_dbmlsync_validate_download_file”一节第 288 页
- “MobiLink 基于文件的下载” 《MobiLink - 服务器管理》
- “-bc 选项”一节第 131 页
- “-ba 选项”一节第 130 页

-bg 选项

创建下载文件时，此选项可创建能用于尚未同步的远程数据库的文件。

语法

dbmlsync -bc "*filename***" -bg** ...

注释

-bg 选项会使下载文件更新远程数据库上的世代号。

此选项允许构建可应用于从未进行过同步的远程数据库的下载文件。否则，必须先执行同步，然后才能应用下载文件。

使用 -bg 选项构建的下载文件应为快照下载文件。基于时间戳的下载不适用于尚未同步的远程数据库，因为新的远程数据库上的上次下载时间戳缺省为 1900 年 1 月 1 日，该时间比下载文件中的上次下载时间戳还早。要使基于文件和基于时间戳的下载可用，下载文件中的上次下载时间戳必须与远程数据库上的相同或者比之更早。

如果系统依赖世代号提供的功能，则切勿将 -bg 下载文件应用到已同步的远程数据库，因为此选项会妨碍该功能。

另请参见

- “MobiLink 基于文件的下载” 《MobiLink - 服务器管理》
- “-ba 选项”一节第 130 页
- “-bc 选项”一节第 131 页
- “MobiLink 世代号”一节 《MobiLink - 服务器管理》
- “对新的远程数据库进行同步”一节 《MobiLink - 服务器管理》

-c 选项

指定远程数据库的连接参数。

语法

dbmlsync -c "*connection-string***"** ...

注释

连接字符串必须赋予 dbmlsync 连接到 SQL Anywhere 远程数据库的 DBA 或 REMOTE DBA 权限。建议使用具有 REMOTE DBA 权限的用户 ID。

指定 *keyword=value* 形式的连接字符串，由分号分隔多个参数。如果某些参数名中包含空格，需要将连接字符串用双引号括起来。

如果未指定 -c，将出现 dbmlsync 设置窗口。您可以在连接窗口的字段中指定其余的命令行选项。

有关连接到 SQL Anywhere 数据库的连接参数的完整列表，请参见“连接参数”一节 《SQL Anywhere 服务器 - 数据库管理》。

-d 选项

删除远程数据库的有冲突的锁定。

语法

dbmlsync -d ...

注释

在同步过程中，除非 LockTables 扩展选项设置为 OFF，否则将锁定正在同步的发布中所涉及的所有表，以防止任何其它进程进行更改。如果另一连接锁定了这些表中的某个，同步会失败或延迟。指定此选项将强制 SQL Anywhere 删除所有其它包含冲突锁定的远程数据库连接，从而可立即进行同步。

另请参见

- “同步中的并发”一节第 108 页

-dc 选项

重新启动以前执行失败的下载。

语法

dbmlsync -dc ...

注释

缺省情况下，如果下载过程中 MobiLink 失败，则不会将任何下载数据应用到远程数据库。不过，它会将接收到的下载部分存储在远程设备上的一个临时文件中，以便在您下次启动 dbmlsync 时指定 -dc 的情况下，可以更快捷地完成下载。指定 -dc 时，dbmlsync 重新启动下载，并尝试下载其在先前的下载中未接收到的部分。如果可以下载剩余的数据，则它会将完整的下载应用到远程数据库。

如果在您使用 -dc 时有新的数据需要上载，则可重新启动的下载将失败。

还可以使用 ContinueDownload 扩展选项或 sp_hook_dbmlsync_end 挂接来重新启动失败的下载。

另请参见

- “恢复失败的下载”一节 《MobiLink - 服务器管理》
- “ContinueDownload (cd) 扩展选项”一节第 182 页
- “sp_hook_dbmlsync_end”一节第 261 页
- “DownloadReadSize (drs) 扩展选项”一节第 186 页

-dl 选项

显示 dbmlsync 消息窗口中（或命令提示符处）和消息日志文件中的消息。

语法

dbmlsync -dl ...

注释

通常向文件记录输出时，在该日志文件中写入的消息将多于输出到 dbmlsync 窗口的消息。此选项将迫使 dbmlsync 将通常只写入文件的信息也输出到窗口。使用此选项可能会影响同步速度。

-do 选项

禁用对脱机事务日志的扫描。

语法

dbmlsync -do ...

注释

如果将多个数据库的事务日志文件存储在一个目录中，那么 dbmlsync 可能无法从这些数据库中的任何一个进行同步，即使这些数据库中的任何一个都不存在脱机事务日志文件。如果使用了启用 -do 选项的 dbmlsync，则 dbmlsync 不会尝试扫描任何脱机事务日志。因此，如果数据库不具有任何脱机事务日志文件，当此数据库与所有其它数据库存储在一个目录中时，启用 -do 选项的 dbmlsync 将能够从此数据库进行同步。

如果使用此选项，但还需要脱机事务日志，则 dbmlsync 将无法同步。

不能与 -x 选项一起使用。

-drs 选项

对于可重新启动的下载，指定发生通信故障后需要重新发送的最大数据量。

语法

dbmlsync -drs *bytes* ...

注释

-drs 选项指定下载读取大小，此选项仅在进行可重新启动的下载时才有用。

Dbmlsync 按块的形式读取下载。下载读取大小定义这些块的大小。发生通信错误时，dbmlsync 会丢失正在处理的整个块。根据错误发生的时间，丢失的字节数范围在 0 到下载读取大小 - 1 之间。例如，如果 DownloadReadSize 为 100 字节，错误在读取 497 字节后发生，则最后读取的 97 字节将丢失。以此种方式丢失的字节会在下载重新启动时重新发送。

通常，较大的下载读取大小值在成功同步时可带来较好的性能，但在发生错误时会导致更多数据需要重新发送。

通信不可靠时，使用此选项的通常做法是减小缺省大小。

缺省值为 **32767**。如果设置此选项的值大于 32767，则使用值 32767。

也可以使用 DownloadReadSize 扩展选项指定下载读取大小。

另请参见

- “DownloadReadSize (drs) 扩展选项”一节第 186 页
- “恢复失败的下载”一节 《MobiLink - 服务器管理》
- “ContinueDownload (cd) 扩展选项”一节第 182 页
- “sp_hook_dbmlsync_end”一节第 261 页
- “-dc 选项”一节第 136 页

示例

以下 dbmlsync 命令行说明在启动 dbmlsync 时如何设置此选项：

```
dbmlsync -drs 100
```

-ds 选项

执行仅下载同步。

语法

dbmlsync -ds ...

注释

当发生仅下载同步时，dbmlsync 不上载任何行操作或数据。但是，它将上载有关模式和进程偏移的信息。

另外，dbmlsync 确保在仅下载同步过程中不会覆盖远程数据库上的更改。这可以通过扫描日志以检测具有等待被上载的操作的行来实现。如果这些行中任何行被下载修改，则将回退下载，而同步将失败。如果同步因此而失败，则您必须执行完全同步来更正该问题。

当具有通过仅下载同步所同步的远程数据库时，您应该定期执行完全双向同步以减少仅下载同步扫描的日志量。否则，完成仅下载同步所需的时间将越来越长。

使用 -ds 时，ConflictRetries 扩展选项将被忽略。dbmlsync 将不会再次尝试进行仅下载同步。当仅下载同步失败时，它在执行正常同步之后才会停止失败。

有关必须为仅下载同步定义的脚本列表，请参见“必需的脚本”一节 《MobiLink - 服务器管理》。

另请参见

- “仅上载同步和仅下载同步”一节 《MobiLink - 服务器管理》
- “DownloadOnly (ds) 扩展选项”一节第 185 页
- “仅下载发布”一节第 97 页

-e 选项

指定扩展选项。

语法

dbmlsync -e *extended-option*=*value*; ...

extended-option:
adr cd cr ctp dbs dir drs ds eh el ft hrt inc isc lt mem mn mp p pp sa sc sch scn st sv toc tor uo v vn vm vo vr vs vu

参数

可以用长格式或短格式来指定扩展选项。

请参见“MobiLink SQL Anywhere 客户端扩展选项”第 175 页。

注释

在命令行中使用 -e 选项指定的选项作用于在命令行中请求的所有同步。例如，在下面的命令行中，扩展选项 sv=test 作用于 pub1 和 pub2 这两个同步。

```
dbmlsync -e "sv=test" -n pub1 -n pub2
```

您可以在 dbmlsync 消息日志和 SYSSYNC 系统视图中查看扩展选项。

若要为单一上载指定扩展选项，请使用 -eu 选项。

另请参见

- “-eu 选项”一节第 145 页
- “SYSSYNC 系统视图”一节 《SQL Anywhere 服务器 - SQL 参考》
- “sp_hook_dbmlsync_set_extended_options”一节第 278 页

示例

以下 dbmlsync 命令行说明在启动 dbmlsync 时如何设置扩展选项：

```
dbmlsync -e "adr=host=localhost;dir=c:\db\logs"...
```

-eh 选项

忽略挂接函数中出现的错误。

语法

dbmlsync -eh ...

-ek 选项

使您可以直接在命令行中为高度加密的数据库指定加密密钥。

语法

dbmlsync -ek *key* ...

注释

如果您有一个高度加密的数据库，则必须提供加密密钥，才能在任何情况下使用数据库或事务日志，包括脱机事务。对于高度加密数据库，您必须指定 -ek 或 -ep，但不要同时指定这两者。如果不为高度加密的数据库指定密钥，该命令将失败。

-ep 选项

提示输入加密密钥。

语法

dbmlsync -ep ...

注释

使用此选项将显示一个窗口，可以在其中输入加密密钥。这样，加密密钥从不以明文显示，提供了额外的安全保证。对于高度加密数据库，您必须指定 -ek 或 -ep，但不要同时指定这两者。如果不为高度加密的数据库指定密钥，该命令将失败。

-eu 选项

指定扩展上载选项。

语法

dbmlsync -n *publication-name* **-eu** *keyword=value*;...

注释

在命令行中使用 -eu 选项指定的扩展选项仅作用于它们前面的 -n 选项所指定的同步。例如，在以下命令行中，扩展选项 sv=test 仅作用于 pub2 的同步。

```
dbmlsync -n pub1 -n pub2 -eu "sv=test"
```

有关在多处设置了扩展选项时如何处理它们的说明，请参见“MobiLink SQL Anywhere 客户端扩展选项”第 175 页。

有关扩展选项的完整列表，请参见“MobiLink SQL Anywhere 客户端扩展选项”第 175 页。

-is 选项

忽略调度指令，以便立即进行同步。

语法

dbmlsync -is ...

注释

忽略调度同步的扩展选项。

有关调度的信息，请参见“调度同步”一节第 114 页。

-k 选项（不建议使用）

同步完成后关闭 dbmlsync。不建议使用此选项。请改用 -qc。

语法

dbmlsync -k ...

另请参见

- “-qc 选项”一节第 161 页

-l 选项

列出可用的扩展选项。

语法

dbmlsync -l ...

注释

与 dbmlsync 命令行一起使用时，该选项显示可用的扩展选项。

另请参见

- “MobiLink SQL Anywhere 客户端扩展选项”第 175 页

-mn 选项

为正在同步的 MobiLink 用户提供新口令。

语法

dbmlsync -mn *password* ...

注释

更改 MobiLink 用户的口令。

有关详细信息，请参见“MobiLink 用户”第 9 页。

另请参见

- “MobiLinkPwd (mp) 扩展选项”一节第 197 页
- “NewMobiLinkPwd (mn) 扩展选项”一节第 198 页
- “-mp 选项”一节第 150 页

-mp 选项

提供正在同步的 MobiLink 用户的口令。

语法

dbmlsync -mp *password ...*

注释

提供用于 MobiLink 用户验证的口令。

有关详细信息，请参见“MobiLink 用户”第 9 页。

另请参见

- “MobiLinkPwd (mp) 扩展选项”一节第 197 页
- “NewMobiLinkPwd (mn) 扩展选项”一节第 198 页
- “-mn 选项”一节第 149 页

-n 选项

指定要同步的发布。

语法

dbmlsync -n *pubname* ...

注释

同步发布的名称。

可以提供多个 -n 选项来同步多个同步发布，但只能在同步配置文件中指定发布一次。

有两种使用 -n 同步多个发布的方法：

- 指定 `-n pub1,pub2,pub3` 在一个下载前的上载中上载 pub1、pub2 和 pub3。

  这种情况下，如果您已经在发布中设置了扩展选项，将仅使用列表中第一个发布中设置的选项。后续发布中设置的扩展选项将被忽略。

- 指定 `-n pub1 -n pub2 -n pub3` 以在三个连续的同步中同步 pub1、pub2 和 pub3。

  当连续的同步以很快的速度发生（例如指定 `-n pub1 -n pub2`）时，dbmlsync 可能会在服务器仍在处理前一个同步时即开始处理其后的同步。这种情况下，第二个同步会失败并返回一个错误，指示不允许并发同步。如果遇到此情况，可以定义 sp_hook_dbmlsync_delay 存储过程，以在每个同步前创建一个延迟。通常，几秒钟到一分钟的延迟就足够了。

  有关详细信息，请参见“sp_hook_dbmlsync_delay”一节第 241 页。

-o 选项

将输出发送到 dbmlsync 消息日志文件中。

语法

dbmlsync -o *filename* ...

注释

将输出附加到日志文件。缺省情况下会将输出发送到屏幕上。

另请参见

- “-os 选项”一节第 153 页
- “-ot 选项”一节第 154 页

-os 选项

指定 dbmlsync 消息日志文件的最大大小，达到此大小时将重命名日志。

语法

dbmlsync -os *size* [**K** | **M** | **G**]...

注释

size 是记录输出消息的文件大小的最大值，以字节为单位。分别使用 k、m 或 g 将单位指定为千字节、兆字节或千兆字节。缺省情况下无最大值限制。最小值限制为 10K。

在 dbmlsync 实用程序将输出消息记录到文件之前，它会检查当前文件的大小。如果日志消息使文件大小超出了指定大小，dbmlsync 实用程序会将输出文件重命名为 *yymmddxx*.dbr，其中 *yymmdd* 代表年、月和日，*xx* 是从 AA 到 ZZ 的顺序字符。

此选项可用于手工删除旧的日志文件并释放磁盘空间。

另请参见

- “-o 选项”一节第 152 页
- “-ot 选项”一节第 154 页

-ot 选项

删除消息日志文件内容，然后将输出消息记录到日志文件。

语法

dbmlsync -ot *logfile* ...

注释

此选项的功能与 -o 选项相同，只是它在将消息写入日志文件之前（启动 dbmlsync 时）删除日志文件内容。

另请参见

- “-o 选项”一节第 152 页
- “-os 选项”一节第 153 页

-p 选项

禁用日志扫描轮询。

语法

dbmlsync -p ...

注释

为构建上载，dbmlsync 必须扫描事务日志。通常它在同步前执行此操作。然而，在调度同步或使用 sp_hook_dbmlsync_delay 挂接时，缺省情况下，dbmlsync 会在同步前的暂停阶段扫描日志。此行为更为有效，因为在同步开始时，日志至少已部分经过了扫描。此缺省行为称为日志扫描轮询。

日志扫描轮询在缺省情况下启用，但仅在已使用调度选项调度同步或使用 sp_hook_dbmlsync_delay 挂接时有效。有效时，轮询会以设置的时间间隔发生；缺省情况下，此时间间隔为 1 分钟（可通过 dbmlsync -pp 选项更改）。

此选项与扩展选项 DisablePolling=on 相同。

另请参见

- “DisablePolling (p) 扩展选项”一节第 183 页
- “PollingPeriod (pp) 扩展选项”一节第 201 页
- “-pp 选项”一节第 159 页

-pc 选项

在多个同步之间保持与 MobiLink 服务器的持久连接。

语法

dbmlsync -pc+ ...

注释

指定此选项后，dbmlsync 正常连接到 MobiLink 服务器，但会使此连接保持打开状态以便用于后续同步。发生以下任何一种情况时将关闭持久连接：

- 发生导致同步失败的错误。
- 活动检查超时。

  请参见“timeout”一节第 65 页。
- 启动了通信类型或地址不同的同步。这意味着它们的设置不同（例如，指定了不同的主机），或者以不同的方式进行了指定（例如，指定了相同的主机和端口，但其顺序不同）。

关闭持久连接时，将打开一个新连接，此连接也是持久连接。

客户端频繁同步并且与服务器建立连接开销很大时，此选项最有用。

缺省情况下，不维护持久连接。

-pd 选项

为 Windows Mobile 预装载指定的 DLL。

语法

dbmlsync -pd *dllname*;...

注释

在 Windows Mobile 上运行 dbmlsync 时，如果使用加密通信流，则必须使用 -pd 选项来确保在启动时装载相应的 DLL。否则，直到需要时 dbmlsync 才会尝试装载 DLL。由于 Windows Mobile 上的资源限制，以后装载这些 DLL 时容易失败。

以下是需要为每种通信协议装载的 DLL：

协议	DLL
ECC	mlcecc10.dll
RSA	mlcrsa10.dll
FIPS	mlcrsafips10.dll

应以分号分隔的列表来指定多个 DLL。例如：

```
-pd mlcrsafips10.dll;mlcrsa10.dll
```

-pi 选项

强制 MobiLink 服务器回应。

语法

dbmlsync -pi -c *connection_string ...*

注释

使用 -pi 时，dbmlsync 将连接到远程数据库、检索连接到 MobiLink 服务器所需的信息、连接服务器以及验证指定的 MobiLink 用户。当 MobiLink 服务器接收到 ping 请求时，它便会连接到统一数据库，验证用户，然后将正在验证的用户状态和值发送回客户端。如果在 ml_user 系统表中找不到 MobiLink 用户名，同时 MobiLink 服务器正在使用命令行选项 -zu+ 运行，则 MobiLink 服务器会将此用户添加到 ml_user MobiLink 系统表。

要充分测试您的连接，应与所有要使用 dbmlsync 同步的同步选项配合使用 -pi 选项。当包括 -pi 时，dbmlsync 不执行同步。

如果 ping 成功，则 MobiLink 服务器将发出一条信息消息。如果 ping 不成功，它会发出一条错误消息。

通过 -pi 启动 dbmlsync 时，MobiLink 服务器仅会执行以下脚本（如果存在）：

- begin_connection
- authenticate_user
- authenticate_user_hashed
- authenticate_parameters
- end_connection

-pp 选项

指定日志扫描频率。

语法

dbmlsync -pp *number* [**h** | **m** | **s**]...

注释

为构建上载，dbmlsync 必须扫描事务日志。通常它在同步前执行此操作。然而，在调度同步或使用 sp_hook_dbmlsync_delay 挂接时，缺省情况下，dbmlsync 会在同步前的暂停阶段扫描日志。此行为更为有效，因为在同步开始时，日志至少已部分经过了扫描。此缺省行为称为日志扫描轮询。

此选项指定日志扫描间的时间间隔。使用后缀 s、m、h 或 d 分别指定以秒、分钟、小时或天为单位。缺省值为 **1** 分钟。如果不指定后缀，则缺省时间单位为分钟。

另请参见

- “PollingPeriod (pp) 扩展选项”一节第 201 页
- “DisablePolling (p) 扩展选项”一节第 183 页
- “-p 选项”一节第 155 页

-q 选项

以最小化窗口启动 MobiLink 同步客户端。

语法

dbmlsync -q ...

-qc 选项

同步完成后关闭 dbmlsync。

语法

dbmlsync -qc ...

注释

使用时，如果同步成功或者消息日志文件是通过 -o 或 -ot 选项指定的，则 dbmlsync 会在同步完成后退出。

另请参见

- “-o 选项”一节第 152 页
- “-ot 选项”一节第 154 页

-r 选项

指定在远程数据库中的偏移和统一数据库中的偏移不一致时，应使用远程偏移。

-rb 选项表示在远程偏移小于统一偏移时（例如，从备份中恢复远程数据库时），使用远程偏移。-r 选项旨在向后兼容，与 -rb 相同。-ra 选项表示在远程偏移大于统一偏移时，使用远程偏移。此选项仅用于极少数情况下并且可能导致数据丢失。

语法

dbmlsync { **-r** | **-ra** | **-rb** } ...

注释

有关进度偏移的信息，请参见“进度偏移”一节第 92 页。

-rb 如果远程数据库是从备份中恢复的，则缺省行为可能会导致数据丢失。这种情况下，远程数据库恢复后第一次运行 dbmlsync 时，应指定 -rb。使用 -rb 时，只要远程数据库中记录的偏移小于统一数据库中记录的偏移，上载就会从远程数据库记录的偏移位置继续。如果使用 -rb 并且远程数据库中的偏移大于或等于统一数据库中的偏移，则会报告错误并且中止同步。

-rb 选项会导致再次上载一些已经上载的数据。这会在统一数据库中造成冲突，应使用适当的冲突解决脚本进行处理。

-ra -ra 选项应仅用于极少数情况下。使用 -ra 时，如果远程偏移大于从统一数据库中获得的偏移，则会从远程数据库中获得的偏移位置开始重新尝试上载。如果使用 -ra 并且远程数据库中的偏移小于或等于统一数据库中的偏移，则会报告错误并且中止同步。

应谨慎使用 -ra 选项。如果造成偏移不匹配的原因是统一数据库的恢复，则在远程数据库中两个偏移之间发生的更改将丢失。当统一数据库已从备份中恢复而远程数据库事务日志已在与远程偏移相同的位置截断时，-ra 选项可能很有用。在这种情况下，从远程数据库上载的所有数据中会丢失从统一偏移位置到远程偏移位置之间的部分。

-sc 选项

指定 dbmlsync 应在每次同步前重装模式信息。

语法

dbmlsync -sc...

注释

在 9.0 版之前，dbmlsync 在每次同步前都重新从数据库中装载模式信息。重装的信息包括外键关系、发布定义、数据库中存储的扩展选项，以及有关数据库设置的信息。装载此信息非常耗时并且多数情况下同步间的信息并不更改。

从 9.0 版开始，缺省情况下，dbmlsync 仅在启动时装载模式信息。如果希望在每次同步前装载此信息，请指定 -sc。

-sp 选项

当使用 -sp 选项时，可将指定的同步配置文件中的选项添加到在命令行上为同步所指定的选项中。

语法

dbmlsync -sp *sync profile*

注释

如果在命令行和同步配置文件中指定了等同的选项，那么命令行上的选项将会替换在配置文件中所指定的选项。

另请参见

- “CREATE SYNCHRONIZATION PROFILE 语句 [MobiLink]”一节 《SQL Anywhere 服务器 - SQL 参考》
- “ALTER SYNCHRONIZATION PROFILE 语句 [MobiLink]”一节 《SQL Anywhere 服务器 - SQL 参考》
- “DROP SYNCHRONIZATION PROFILE 语句 [MobiLink]”一节 《SQL Anywhere 服务器 - SQL 参考》

-tu 选项

指定远程数据库上的每个事务都应在一次同步中作为一个单独的事务上载。

语法

dbmlsync -tu ...

注释

使用 -tu 时，将创建**事务性上载**：dbmlsync 将远程数据库上的每个事务作为不同的事务上载。接收到事务后，MobiLink 服务器分别应用和提交每个事务。

使用 -tu 时，远程数据库上的事务的顺序始终在统一数据库上保留。不过，事务中的操作顺序可能不会保留，原因有两个：

- MobiLink 始终根据外键关系应用更新。例如，当数据在子表和父表中更改后，MobiLink 会将数据先插入父表然后再插入子表，但会先从子表删除数据然后再从父表删除数据。如果您的远程操作不遵循此顺序，则统一数据库上的操作顺序会有所不同。
- 事务中的操作将被合并。这表示，如果在一个事务中将同一行更改三次，则仅上载行的最后形式。

如果事务性上载中断，则未发送的数据将被发送到下一个同步。多数情况下，此时只发送未成功完成的事务。在某些情况下（例如，首次同步预订期间发生上载失败时），dbmlsync 将重新发送所有事务。

不使用 -tu 时，MobiLink 会将远程数据库上的所有更改合并到上载中的一个事务内。这表示，如果在两次同步之间将同一行更改三次，则无论远程事务数有多少，仅上载行的最后形式。此缺省行为在许多情况下都是有效和理想的。

不过，在某些情况下，您可能希望在统一数据库上保留远程事务。例如，您可能希望在统一数据库上定义触发器，当这些触发器在远程数据库中触发时，会对事务进行处理。

另外，将上载划分为较小的事务有诸多优点。许多统一数据库已经过了优化处理，适用于小型事务，所以发送非常大的事务时，效率不会高或导致太多争用。同时，使用 -tu 时，如果在上载过程中发生通信错误，可能不会丢失整个上载。使用 -tu 且发生上载错误时，所有成功上载的事务都将得以应用。

-tu 选项使 MobiLink 的行为模式非常接近 SQL Remote。主要差别在于 SQL Remote 会将所有更改以其发生顺序复制到远程数据库中，它不进行合并。要模拟此行为，必须每次在远程数据库上执行数据库操作后进行提交。

-tu 不可与 Increment 扩展选项或脚本式上载结合使用。

另请参见

- “-tx 选项”一节 《MobiLink - 服务器管理》
- “从自引用表上载数据”一节 《MobiLink - 服务器管理》

-u 选项

指定 MobiLink 用户名。

语法

dbmlsync -u *ml_username* ...

注释

您可以在 dbmlsync 命令行中指定一个用户，其中 *ml_username* 是在 CREATE SYNCHRONIZATION SUBSCRIPTION 语句的 FOR 子句（对应于要处理的预订）中使用的名称。

此选项应当与 -n *publication* 一起使用，以标识 dbmlsync 将操作的预订。每个预订都由一个 *ml_username*，*publication* 对唯一标识。

只能在命令行中指定一个用户名。在一次单独运行中将要同步的所有预订必须涉及同一个用户。如果命令行中使用 -n 选项指定的每个发布都只有一个预订，则可以省略 -u 选项。

-ui 选项

对于支持 X 窗口服务器的 Linux，如果没有提供可用的显示则以 shell 模式启动 dbmlsync。

语法

mlsrv11 -c "*connection-string***" -ui** ...

注释

使用此选项时，dbmlsync 会尝试使用 X 窗口启动。如果此操作失败，它会以 shell 模式启动。

如果指定 -ui，dbmlsync 会尝试查找可用的显示。如果没有找到，例如因为 X 窗口服务器没有运行，那么 dbmlsync 会以 shell 模式启动。

-uo 选项

指定同步只包括上载。

语法

dbmlsync -uo...

注释

在仅上载同步期间，dbmlsync 会准备并发送一个上载到 MobiLink，所采用的方式与正常的完整同步完全相同。但是，MobiLink 只发送一个确认来指示上载是否成功提交，而不会回发下载。

有关必须为仅上载同步定义的脚本列表，请参见“必需的脚本”一节 《MobiLink - 服务器管理》。

另请参见

- “仅上载同步和仅下载同步”一节 《MobiLink - 服务器管理》
- “DownloadOnly (ds) 扩展选项”一节第 185 页
- “UploadOnly (uo) 扩展选项”一节第 210 页

-urc 选项

指定同步过程中要上载的估计行数。

语法

dbmlsync -urc *row-estimate ...*

注释

为了提高性能，可以指定同步过程中要上载的行的估算行数。上载大量的行时，此项设置尤其有用。估计值越高上载速度越快，但使用的内存越多。

同步继续正确进行，与指定的估计值无关。

另请参见

- “Memory (mem) 扩展选项”一节第 195 页
- “对于较大的上载，估计行数”一节 《MobiLink - 服务器管理》

-ux 选项

在 Linux 上，用于打开显示消息的 dbmlsync 消息窗口。

语法

dbmlsync -ux...

注释

指定 -ux 时，dbmlsync 必须能够找到一个可用显示。如果它找不到一个可用显示（例如，因为没有设置 DISPLAY 环境变量或 X 窗口服务器没有运行），dbmlsync 将无法启动。

若要以安静模式运行 dbmlsync 消息窗口，请使用 -q 选项。

在 Windows 上，dbmlsync 消息窗口将自动出现。

另请参见

- “-q 选项”一节第 160 页

-v 选项

用于指定消息日志文件中记录的信息和同步窗口中显示的信息。高详细程度可能会影响性能，通常应该仅在开发阶段使用高详细程度。

语法

dbmlsync -v [*levels*] ...

注释

-v 选项影响消息日志文件和同步窗口。如果在 dbmlsync 命令行中指定 -o 或 -ot，则仅有一个消息日志。

如果仅指定 -v，将记录少量的信息。

levels 的值如下所示。一次可以使用这些选项中的一个或多个；例如，-vnrsu 或 -v+cp。

- **+** 启用所有记录选项，c 和 p 除外。
- **c** 在日志中显示连接字符串。
- **p** 在日志中显示口令。
- **n** 记录上载和下载的行数。
- **o** 记录有关您指定的命令行选项和扩展选项的信息。
- **r** 记录上载和下载的行值。
- **s** 记录与挂接脚本相关的消息。
- **u** 记录关于上载的信息。

有功能类似于 -v 选项的扩展选项。如果您同时指定 -v 选项和扩展选项，并且它们之间有冲突，则 -v 选项取代扩展选项。如果没有冲突，详细程度记录选项是叠加的—使用您指定的所有选项。如果记录详细程度由扩展选项设置，记录不会立即生效，因此不记录启动信息。到第一次同步时，-v 选项和扩展选项之间的记录行为就完全一样了。

另请参见

- “Verbose (v) 扩展选项”一节第 211 页
- “VerboseHooks (vs) 扩展选项”一节第 212 页
- “VerboseMin (vm) 扩展选项”一节第 213 页
- “VerboseOptions (vo) 扩展选项”一节第 214 页
- “VerboseRowCounts (vn) 扩展选项”一节第 215 页
- “VerboseRowValues (vr) 扩展选项”一节第 216 页
- “-o 选项”一节第 152 页
- “-ot 选项”一节第 154 页

-wc 选项

指定一个窗口类名。

语法

dbmlsync -wc *class-name* ...

注释

此选项指定的类名可用于在 dbmlsync 处于悬停模式的任何时候（例如当启用了调度或正在使用服务器启动的同步时）查找并唤醒它。

另外，窗口类名标识用于 ActiveSync 同步的应用程序。将应用程序注册为在 ActiveSync 同步中使用时必须给出此类名称。

此选项仅适用于 Windows。

另请参见

- “为 ActiveSync 注册 SQL Anywhere 客户端”一节第 112 页
- “使用 ActiveSync 同步”一节第 110 页
- “Schedule (sch) 扩展选项”一节第 202 页中的 INFINITE 关键字
- “调度同步”一节第 114 页

示例

```
dbmlsync -wc dbmlsync_$message_end...
```


-x 选项

在扫描事务日志寻找外传消息后，重命名并重新启动该事务日志。

语法

dbmlsync -x [*size* [**K** | **M** | **G**] ...

注释

可选的 *size* 表示事务日志仅在大于指定的大小后才会被重命名。分别使用 k、m 或 g 将单位指定为千字节、兆字节或千兆字节。缺省大小为 0。

在某些情况下，在数据库服务器关闭时，向统一数据库同步数据可以代替远程数据库备份或事务日志重命名。

如果对远程数据库不执行定期备份，事务日志将会一直增长。要控制事务日志的大小，除了使用 -x 选项，还可以使用 SQL Anywhere 事件处理程序。

另请参见

- “使用调度和事件自动完成任务” 《SQL Anywhere 服务器 - 数据库管理》
- “delete_old_logs 选项 [MobiLink 客户端] [SQL Remote] [复制代理]”一节 《SQL Anywhere 服务器 - 数据库管理》
- “CREATE EVENT 语句”一节 《SQL Anywhere 服务器 - SQL 参考》

MobiLink SQL Anywhere 客户端扩展选项

目录

dbmlsync 扩展选项简介

可以使用 -e 或 -eu 选项在 dbmlsync 命令行中指定扩展选项，也可以将它们存储在数据库中。将扩展选项存储在数据库中的方法是：使用 Sybase Central，或使用 sp_hook_dbmlsync_set_extended_options 事件挂接，或在以下任何语句中使用 OPTION 子句：

- CREATE SYNCHRONIZATION SUBSCRIPTION
- ALTER SYNCHRONIZATION SUBSCRIPTION
- CREATE SYNCHRONIZATION USER
- ALTER SYNCHRONIZATION USER
- CREATE SYNCHRONIZATION SUBSCRIPTION（在不指定同步用户的情况下；这将扩展选项和发布关联起来）

优先级顺序

Dbmlsync 会将存储在数据库中的选项与命令行上指定的选项组合到一起。如果指定的选项有冲突，dbmlsync 将使用以下方法解决这些冲突。在以下列表中，位置靠前的方法所指定的选项优先于位置靠后的方法所指定的选项。

1. 在 sp_hook_dbmlsync_set_extended_options 事件挂接中指定的选项。
2. 在命令行中指定的非扩展选项。（例如，`-ds` 会覆盖 `-e "ds=off"`。
3. 在命令行中使用 -eu 选项指定的选项。
4. 在命令行中使用 -e 选项指定的选项。
5. 为预订指定的选项（通过 SQL 语句或 Sybase Central）。使用 [**部署同步模型向导**] 部署 MobiLink 模型时会设置扩展选项并在预订中指定这些扩展选项。
6. 为 MobiLink 用户指定的选项（通过 SQL 语句或 Sybase Central）。
7. 为发布指定的选项（通过 SQL 语句或 Sybase Central）。

注意
此优先级顺序也影响连接参数，例如上面提到的在 SQL 语句中使用 TYPE 和 ADDRESS 选项指定的连接参数。

您可以在日志和 SYSSYNC 系统视图中查看扩展选项。

有关如何使用扩展选项调整同步的信息，请参见“使用 dbmlsync 扩展选项”一节第 106 页。

另请参见

- “-e 选项”一节第 141 页
- “-eu 选项”一节第 145 页
- “CREATE SYNCHRONIZATION SUBSCRIPTION 语句 [MobiLink]”一节 《SQL Anywhere 服务器 - SQL 参考》
- “ALTER SYNCHRONIZATION SUBSCRIPTION 语句 [MobiLink]”一节 《SQL Anywhere 服务器 - SQL 参考》
- “CREATE SYNCHRONIZATION USER 语句 [MobiLink]”一节 《SQL Anywhere 服务器 - SQL 参考》
- “ALTER SYNCHRONIZATION USER 语句 [MobiLink]”一节 《SQL Anywhere 服务器 - SQL 参考》
- “CREATE PUBLICATION 语句 [MobiLink] [SQL Remote]”一节 《SQL Anywhere 服务器 - SQL 参考》
- “ALTER PUBLICATION 语句 [MobiLink] [SQL Remote]”一节 《SQL Anywhere 服务器 - SQL 参考》
- “SYSSYNC 系统视图”一节 《SQL Anywhere 服务器 - SQL 参考》
- “sp_hook_dbmlsync_set_extended_options”一节第 278 页

示例

以下 dbmlsync 命令行说明在启动 dbmlsync 时如何设置扩展选项：

```
dbmlsync -e "adr=host=localhost;dir=c:\db\logs"...
```

以下 SQL 语句说明如何在数据库中存储扩展选项：

```
CREATE SYNCHRONIZATION SUBSCRIPTION TO mypub
   FOR mluser
   ADDRESS 'host=localhost'
   OPTION schedule='weekday@11:30am-12:30pm', dir='c:\db\logs'
```

以下 dbmlsync 命令行会打开用法屏幕，其中将列出选项及其语法：

```
dbmlsync -l
```

CommunicationAddress (adr) 扩展选项

指定用于连接到 MobiLink 服务器的网络协议选项。

语法

adr=*protocol-option*; ...

参数

请参见“MobiLink 客户端网络协议选项汇总”一节第 33 页。

注释

您必须确保 MobiLink 用户的所有预订仅与一个统一数据库同步。否则，您可能会遇到数据丢失和不可预知的行为。

如果使用的是重定向器，请参见“为重定向器配置 MobiLink 客户端和服务器”一节 《MobiLink - 服务器管理》。

此选项有一个长格式和一个短格式：您可以使用 **adr**，也可以使用 **CommunicationAddress**。

还可以使用创建或变更发布、预订或用户的 SQL 语句将此选项存储在数据库中。有关详细信息，请参见：

- “CREATE SYNCHRONIZATION SUBSCRIPTION 语句 [MobiLink]”一节 《SQL Anywhere 服务器 - SQL 参考》
- “CREATE SYNCHRONIZATION USER 语句 [MobiLink]”一节 《SQL Anywhere 服务器 - SQL 参考》

使用 CommunicationType 扩展选项指定网络协议类型。

请参见“CommunicationType (ctp) 扩展选项”一节第 180 页。

另请参见

- “MobiLink 客户端网络协议选项”第 31 页
- “为重定向器配置 MobiLink 客户端和服务器”一节 《MobiLink - 服务器管理》

示例

以下 dbmlsync 命令行说明在启动 dbmlsync 时如何设置此选项：

```
dbmlsync -e "adr=host=localhost"
```

要在命令行上指定多个网络协议选项，请将其用单引号括起来。例如，

```
dbmlsync -e "adr='host=somehost;port=5001'"
```

要在数据库中存储 Address 或 CommunicationType，可使用扩展选项或 ADDRESS 子句。例如，

```
CREATE SYNCHRONIZATION SUBSCRIPTION
   TO sales_publication
   FOR ml_user1
   ADDRESS 'host=localhost;port=2439'
```

CommunicationType (ctp) 扩展选项

指定用于连接到 MobiLink 服务器的网络协议的类型。

语法

ctp=*network-protocol*; ...

注释

network-protocol 可以是 **tcpip**、**tls**、**http** 或 **https** 之一。缺省值为 **tcpip**。

您必须确保 MobiLink 用户的所有预订仅与一个统一数据库同步。否则，您可能会遇到数据丢失和不可预知的行为。

此选项有一个长格式和一个短格式：您可以使用 **ctp**，也可以使用 **CommunicationType**。

另请参见

- “加密 MobiLink 客户端/服务器通信”一节 《SQL Anywhere 服务器 - 数据库管理》
- “CommunicationAddress (adr) 扩展选项”一节第 179 页

示例

以下 dbmlsync 命令行说明在启动 dbmlsync 时如何设置此选项：

```
dbmlsync -e "ctp=https"
```

要在数据库中存储 CommunicationType，可使用扩展选项或 TYPE 子句。例如，

```
CREATE SYNCHRONIZATION SUBSCRIPTION
   TO sales_publication
   FOR ml_user1
   TYPE 'tcpip'
```


ConflictRetries (cr) 扩展选项

指定在因为冲突而导致下载失败时重试的次数。

语法

cr=*number*; ...

注释

当扩展选项 LockTables 设置为 OFF 时（防止 dbmlsync 获取对正在同步的表的锁定）时，可在建立上载之后和应用下载之前将操作应用于数据库。如果这些更改影响下载所要更改的行，则 dbmlsync 会将其视为冲突，从而不会应用下载流。如果发生这种情况，dbmlsync 会重试整个同步。该选项控制所执行的重试次数。

此选项仅在 LockTables 选项为 OFF（非缺省设置）时有用。

缺省值是 **-1**（重试可无限期地继续）。

此选项有一个长格式和一个短格式：您可以使用 **cr**，也可以使用 **ConflictRetries**。

您还可以在数据库中存储扩展选项。有关 dbmlsync 扩展选项的详细信息，请参见“dbmlsync 扩展选项简介”一节第 177 页。

另请参见

- “冲突处理”一节 《MobiLink - 服务器管理》

示例

以下 dbmlsync 命令行说明在启动 dbmlsync 时如何设置此选项：

```
dbmlsync -e "cr=5"
```

以下 SQL 语句说明如何在数据库中存储此选项：

```
CREATE SYNCHRONIZATION SUBSCRIPTION
   TO sales_publication
   FOR ml_user1
   OPTION cr='5';
```


ContinueDownload (cd) 扩展选项

重新启动以前执行失败的下载。

语法

cd={ ON | OFF }; ...

注释

如果下载过程中 MobiLink 失败，则不会将任何下载数据应用到远程数据库。不过，该选项会将其收到的部分下载存储在远程设备的临时文件中，以便以后能够重新启动它。设置扩展选项 cd=on 时，dbmlsync 重新启动下载，并尝试下载其在先前的下载中未接收到的部分。如果可以下载剩余的数据，则它会将完整的下载应用到远程数据库。

如果在您设置 -cd=on 时有新的数据需要上载，则可重新启动的下载会失败。

也可以使用 -dc 选项或 sp_hook_dbmlsync_end 挂接，为 SQL Anywhere 远程数据库指定可重新启动的下载。

此选项有一个长格式和一个短格式：您可以使用 **cd**，也可以使用 **ContinueDownload**。

您还可以在数据库中存储扩展选项。有关 dbmlsync 扩展选项的详细信息，请参见“dbmlsync 扩展选项简介”一节第 177 页。

另请参见

- “恢复失败的下载”一节 《MobiLink - 服务器管理》
- “sp_hook_dbmlsync_set_extended_options”一节第 278 页
- “-dc 选项”一节第 136 页
- “sp_hook_dbmlsync_end”一节第 261 页

示例

以下 dbmlsync 命令行说明在启动 dbmlsync 时如何设置此选项：

```
dbmlsync -e "cd=on"
```

以下 SQL 语句说明如何在数据库中存储此选项：

```
CREATE SYNCHRONIZATION SUBSCRIPTION
   TO sales_publication
   FOR ml_user1
   OPTION cd='on';
```


DisablePolling (p) 扩展选项

禁用自动日志扫描轮询。

语法

p={ ON | OFF }; ...

注释

为构建上载，dbmlsync 必须扫描事务日志。通常它在同步前执行此操作。不过，如果已调度同步，则缺省情况下，dbmlsync 将在两次已调度的同步之间扫描日志；如果使用 sp_hook_dbmlsync_delay 挂接，则缺省情况下，dbmlsync 将在同步前的暂停阶段扫描日志。此行为更为有效，因为在同步开始时，日志至少已部分地经过了扫描。此缺省行为称为日志扫描轮询。

日志扫描轮询在缺省情况下将会启用，但仅在已调度同步或使用 sp_hook_dbmlsync_delay 挂接时有效。当日志扫描轮询有效时，它会按所设置的时间间隔发生：dbmlsync 扫描到日志的结尾处，等待轮询周期，然后扫描日志中的所有新事务。缺省情况下，轮询周期为 1 分钟，但该周期可使用 dbmlsync -pp 选项或 PollingPeriod 扩展选项进行更改。

缺省设置是不禁用日志扫描轮询 (**OFF**)。

此选项与 **dbmlsync -p** 相同。

此选项有一个长格式和一个短格式：您可以使用 **p**，也可以使用 **DisablePolling**。

您还可以在数据库中存储扩展选项。有关 dbmlsync 扩展选项的详细信息，请参见“dbmlsync 扩展选项简介”一节第 177 页。

另请参见

- “PollingPeriod (pp) 扩展选项”一节第 201 页
- “-p 选项”一节第 155 页
- “-pp 选项”一节第 159 页

示例

以下 dbmlsync 命令行说明在启动 dbmlsync 时如何设置此选项：

```
dbmlsync -e "p=on"
```

以下 SQL 语句说明如何在数据库中存储此选项：

```
CREATE SYNCHRONIZATION SUBSCRIPTION
   TO sales_publication
   FOR ml_user1
   OPTION p='on';
```


DownloadBufferSize (dbs) 扩展选项

指定下载缓冲区的大小。

语法

dbs=*number*[**K** | **M**]; ...

注释

缓冲区大小以字节为单位指定。使用后缀 k 或 m 分别指定以千字节或兆字节为单位。

如果将此选项设置为 0，dbmlsync 不会将下载内容放在缓冲区中。如果该选项大于 0，则会在将下载流应用到远程数据库之前，利用 MobiLink 服务器从通信流中读取整个下载流。如果下载流可全部放在该选项所指定的空间内，则将其全部保存在内存中；否则，将其中的某些内容写入临时文件。

如果设置大于 0 但小于 4 KB，则 dbmlsync 会使用 4 KB 的缓冲区大小并发出警告。在 Windows Mobile 上，其缺省值为 **32k**；而在所有其它操作系统上，其缺省值为 **1m**。

此选项有一个长格式和一个短格式：您可以使用 **dbs**，也可以使用 **DownloadBufferSize**。

您还可以在数据库中存储扩展选项。有关 dbmlsync 扩展选项的详细信息，请参见“dbmlsync 扩展选项简介”一节第 177 页。

示例

以下 dbmlsync 命令行说明在启动 dbmlsync 时如何设置此选项：

```
dbmlsync -e "dbs=32k"
```

以下 SQL 语句说明如何在数据库中存储此选项：

```
CREATE SYNCHRONIZATION SUBSCRIPTION
   TO sales_publication
   FOR ml_user1
   OPTION dbs='32k';
```


DownloadOnly (ds) 扩展选项

指定同步应该为仅下载同步。

语法

ds={ ON | OFF }; ...

注释

当发生仅下载同步时，dbmlsync 不上载任何行操作或数据。但是，它将上载有关模式和进程偏移的信息。

另外，dbmlsync 确保在仅下载同步过程中不会覆盖远程数据库上的更改。这可以通过扫描日志以检测具有等待被上载的操作的行来实现。如果这些行中任何行被下载修改，则将回退下载，而同步将失败。如果同步因此而失败，则您必须执行完全同步来更正该问题。

当具有通过仅下载同步所同步的远程数据库时，您应该定期执行完全同步以减少仅下载同步扫描的日志量。否则，完成仅下载同步所需的时间将越来越长。如果有问题，则可选择使用仅下载发布来避免同步过程中出现的日志问题。

有关必须为仅下载同步定义的脚本列表，请参见“必需的脚本”一节 《MobiLink - 服务器管理》。

缺省值为 **OFF**（上载和下载的完全同步）。

此选项有一个长格式和一个短格式：您可以使用 **ds**，也可以使用 **DownloadOnly**。

您还可以在数据库中存储扩展选项。有关 dbmlsync 扩展选项的详细信息，请参见“dbmlsync 扩展选项简介”一节第 177 页。

另请参见

- “-ds 选项”一节第 140 页
- “仅下载发布”一节第 97 页
- “仅上载同步和仅下载同步”一节 《MobiLink - 服务器管理》

示例

以下 dbmlsync 命令行说明在启动 dbmlsync 时如何设置此选项：

```
dbmlsync -e "ds=on"
```

以下 SQL 语句说明如何在数据库中存储此选项：

```
CREATE SYNCHRONIZATION SUBSCRIPTION
   TO sales_publication
   FOR ml_user1
   OPTION ds='ON';
```


DownloadReadSize (drs) 扩展选项

对于可重新启动的下载，指定发生通信故障后需要重新发送的最大数据量。

语法

drs=*number*[**K**]; ...

注释

DownloadReadSize 选项仅在进行可重新启动的下载时有用。

下载读取大小是以字节为单位指定的。也可以使用后缀 k 来指定以千字节为单位。

Dbmlsync 按块的形式读取下载。DownloadReadSize 定义这些块的大小。发生通信错误时，dbmlsync 会丢失正在处理的整个块。丢失字节数的范围在 0 和 DownloadReadSize -1 之间，具体取决于错误发生的时间。所以，举例来说，如果 DownloadReadSize 为 100 字节，且在读取 497 字节后发生错误，则最后读取的 97 字节将会丢失。以此种方式丢失的字节会在下载重新启动时重新发送。

通常，较大的 DownloadReadSize 值在成功同步时可带来较好的性能，但在发生错误时会导致更多数据需要重新发送。

通信不可靠时，通常会使用此选项来减小缺省大小。

缺省值为 **32767**。如果设置此选项的值大于 32767，则使用值 32767。

此选项有一个长格式和一个短格式：您可以使用 **drs**，也可以使用 **DownloadReadSize**。

您还可以在数据库中存储扩展选项。有关 dbmlsync 扩展选项的详细信息，请参见“dbmlsync 扩展选项简介”一节第 177 页。

另请参见

- “-drs 选项”一节第 139 页
- “恢复失败的下载”一节 《MobiLink - 服务器管理》
- “ContinueDownload (cd) 扩展选项”一节第 182 页
- “sp_hook_dbmlsync_end”一节第 261 页
- “-dc 选项”一节第 136 页

示例

以下 dbmlsync 命令行说明在启动 dbmlsync 时如何设置此选项：

```
dbmlsync -e "drs=100"
```

以下 SQL 语句说明如何在数据库中存储此选项：

```
CREATE SYNCHRONIZATION SUBSCRIPTION
   TO sales_publication
   FOR ml_user1
   OPTION drs='100';
```


ErrorLogSendLimit (el) 扩展选项

指定在发生同步错误时，dbmlsync 应将远程消息日志文件中的多少内容发送给服务器。

语法

el=*number*[**K** | **M**]; ...

注释

此选项是以字节为单位指定的。使用后缀 k 或 m 分别指定以千字节或兆字节为单位。

此选项指定在同步期间发生错误时，dbmlsync 发送给 MobiLink 服务器的消息日志的字节数。如果不想发送任何 dbmlsync 消息日志，请将此选项设置为 **0**。

如果此选项不是 0，则客户端发生错误时会上载错误日志。并非所有客户端错误都会导致日志被发送：当出现以下错误时将不会发送日志：通信错误，或者在 dbmlsync 未连接到 MobiLink 服务器时所发生的错误。如果上载发送之后发生错误，则只有在 SendDownloadAck 扩展选项设为 ON 时才会上载错误日志。

如果将 ErrorLogSendLimit 设置得足够大，则 dbmlsync 会将整个消息日志从当前会话发送到 MobiLink 服务器。例如，如果消息日志消息被附加到旧的消息日志文件，则 dbmlsync 仅发送当前会话中所生成的新消息。如果新消息的总长度大于 ErrorLogSendLimit，则 dbmlsync 将仅记录新生成的错误和日志消息的最后部分，直至达到指定大小。

注意：消息日志的大小将受详细程度设置影响。可使用 dbmlsync -v 选项来调整这些设置，也可使用以 "verbose" 开头的 dbmlsync 扩展选项来调整它们。有关详细信息，请参见“-v 选项”一节第 171 页和 -e verbose 选项：

- “Verbose (v) 扩展选项”一节第 211 页
- “VerboseHooks (vs) 扩展选项”一节第 212 页
- “VerboseMin (vm) 扩展选项”一节第 213 页
- “VerboseOptions (vo) 扩展选项”一节第 214 页
- “VerboseRowCounts (vn) 扩展选项”一节第 215 页
- “VerboseRowValues (vr) 扩展选项”一节第 216 页
- “VerboseUpload (vu) 扩展选项”一节第 217 页

缺省值为 **32K**。

此选项有一个长格式和一个短格式：您可以使用 **el**，也可以使用 **ErrorLogSendLimit**。

您还可以在数据库中存储扩展选项。有关 dbmlsync 扩展选项的详细信息，请参见“dbmlsync 扩展选项简介”一节第 177 页。

示例

以下 dbmlsync 命令行说明在启动 dbmlsync 时如何设置此选项：

```
dbmlsync -e "el=32k"
```

以下 SQL 语句说明如何在数据库中存储此选项：

```
CREATE SYNCHRONIZATION SUBSCRIPTION
   TO sales_publication
```

```
FOR ml_user1
OPTION el='32k';
```


FireTriggers (ft) 扩展选项

指定当应用下载时触发器应在远程数据库上触发。

语法

ft={ ON | OFF }; ...

注释

缺省值为 **ON**。

此选项有一个长格式和一个短格式：您可以使用 **ft**，也可以使用 **FireTriggers**。

您还可以在数据库中存储扩展选项。有关 dbmlsync 扩展选项的详细信息，请参见“dbmlsync 扩展选项简介”一节第 177 页。

示例

以下 dbmlsync 命令行说明在启动 dbmlsync 时如何设置此选项：

```
dbmlsync -e "ft=off"
```

以下 SQL 语句说明如何在数据库中存储此选项：

```
CREATE SYNCHRONIZATION SUBSCRIPTION
   TO sales_publication
   FOR ml_user1
   OPTION ft='off';
```


HoverRescanThreshold (hrt) 扩展选项

当使用调度时，将限制执行重新扫描前允许积累的已放弃内存量。

语法

hrt=*number*[**K** | **M**]; ...

注释

以字节为单位指定内存。使用后缀 k 或 m 分别指定以千字节或兆字节为单位。缺省值为 **1m**。

如果在命令行中指定了多个 -n 选项，dbmlsync 可能会遇到碎片，而碎片会形成被放弃的内存。只能通过重新扫描数据库事务日志来恢复放弃的内存。使用此选项，可指定对日志重新扫描和内存恢复之前所允许积累的已放弃内存量的限制。控制已放弃内存恢复的另一种方法是：执行 sp_hook_dbmlsync_log_rescan 存储过程。

此选项有一个长格式和一个短格式：您可以使用 **hrt**，也可以使用 **HoverRescanThreshold**。

您还可以在数据库中存储扩展选项。有关 dbmlsync 扩展选项的详细信息，请参见“dbmlsync 扩展选项简介”一节第 177 页。

另请参见

- “sp_hook_dbmlsync_log_rescan”一节第 263 页

示例

以下 dbmlsync 命令行说明在启动 dbmlsync 时如何设置此选项：

```
dbmlsync -e "hrt=2m"
```

以下 SQL 语句说明如何在数据库中存储此选项：

```
CREATE SYNCHRONIZATION SUBSCRIPTION
   TO sales_publication
   FOR ml_user1
   OPTION hrt='2m';
```


IgnoreHookErrors (eh) 扩展选项

指定应该忽略挂接函数中所出现的错误。

语法

eh={ ON | OFF }; ...

注释

缺省值为 **OFF**。

此选项有一个长格式和一个短格式：您可以使用 **eh**，也可以使用 **IgnoreHookErrors**。

此选项等效于 dbmlsync -eh 选项。

您还可以在数据库中存储扩展选项。有关 dbmlsync 扩展选项的详细信息，请参见“dbmlsync 扩展选项简介”一节第 177 页。

示例

以下 dbmlsync 命令行说明在启动 dbmlsync 时如何设置此选项：

```
dbmlsync -e "eh=off"
```

以下 SQL 语句说明如何在数据库中存储此选项：

```
CREATE SYNCHRONIZATION SUBSCRIPTION
   TO sales_publication
   FOR ml_user1
   OPTION eh='off';
```


IgnoreScheduling (isc) 扩展选项

指定应该忽略调度设置。

语法

isc={ ON | OFF }; ...

注释

如果设置为 ON，则 dbmlsync 会忽略在扩展选项中指定的所有调度信息而立即进行同步。缺省值为 **OFF**。

此选项等效于 dbmlsync -is 选项。

此选项有一个长格式和一个短格式：您可以使用 **isc**，也可以使用 **IgnoreScheduling**。

您还可以在数据库中存储扩展选项。有关 dbmlsync 扩展选项的详细信息，请参见“dbmlsync 扩展选项简介”一节第 177 页。

另请参见

- “调度同步”一节第 114 页
- “Schedule (sch) 扩展选项”一节第 202 页

示例

以下 dbmlsync 命令行说明在启动 dbmlsync 时如何设置此选项：

```
dbmlsync -e "isc=off"
```

以下 SQL 语句说明如何在数据库中存储此选项：

```
CREATE SYNCHRONIZATION SUBSCRIPTION
   TO sales_publication
   FOR ml_user1
   OPTION isc='off';
```


Increment (inc) 扩展选项

启用增量上载和控制上载增量的大小。

语法

inc=*number*[**K** | **M**]; ...

注释

此选项以字节为单位指定最小扫描增量。使用后缀 k 或 m 分别指定以千字节或兆字节为单位。

指定此选项时，将上载按一个或多个部分发送给 MobiLink。在站点难于将连接维持足够长时间以完成完整上载时，这将非常有用。不设置此选项时，将上载按单一单元发送。

此选项的值所指定的每个上载部分的大小几乎相同。此选项的值控制每个上载部分的大小，如下所示。Dbmlsync 通过扫描数据库事务日志来构建上载。设置此选项时，dbmlsync 会扫描选项中所设置的字节数，然后继续扫描，直至到达没有未完成的部分事务的第一时刻—所有事务均已提交或均已回退的下一时刻。然后，它会将扫描到的内容作为上载部分发送，并从停止扫描的位置重新开始扫描日志。

Increment 扩展选项不可与脚本式上载或事务性上载结合使用。

此选项有一个长格式和一个短格式：您可以使用 **inc**，也可以使用 **Increment**。

您还可以在数据库中存储扩展选项。有关 dbmlsync 扩展选项的详细信息，请参见“dbmlsync 扩展选项简介”一节第 177 页。

示例

以下 dbmlsync 命令行说明在启动 dbmlsync 时如何设置此选项：

```
dbmlsync -e "inc=32000"
```

以下 SQL 语句说明如何在数据库中存储此选项：

```
CREATE SYNCHRONIZATION SUBSCRIPTION
   TO sales_publication
   FOR ml_user1
   OPTION inc='32k';
```


LockTables (lt) 扩展选项

指定在同步前应将被同步的发布中的表锁定。

语法

lt={ ON | OFF | SHARE | EXCLUSIVE }; ...

注释

SHARE 表示 dbmlsync 在共享模式下锁定所有同步表。EXCLUSIVE 表示 dbmlsync 在独占模式下锁定所有同步表。在除 Windows Mobile 之外的所有平台上，ON 等效于 SHARE。在 Windows Mobile 设备上，ON 等效于 EXCLUSIVE。

缺省值是 OFF。这表示在缺省情况下，dbmlsync 不锁定任何同步表，但以下情况除外：

- 如果在当前同步中存在使用基于脚本的上载的发布，或者在远程数据库中定义了 sp_hook_dbmlsync_schema_upgrade 挂接，则 dbmlsync 会使用 ON 来锁定同步表。
- 如果在先前的同步中下载了直通脚本，并且需要在当前同步中自动执行这些脚本，则会使用 SHARE 或 EXCLUSIVE（具体取决于直通脚本的要求）来锁定同步表。

设置为 ON 将阻止在同步过程中进行修改。

有关共享锁和独占锁的详细信息，请参见“锁定的工作方式”一节 《SQL Anywhere 服务器 - SQL 的用法》和“LOCK TABLE 语句”一节 《SQL Anywhere 服务器 - SQL 参考》。

有关在 MobiLink 应用程序中锁定表的详细信息，请参见“同步中的并发”一节第 108 页。

在独占模式下锁定同步表（Windows Mobile 设备的缺省值）时，没有任何其它连接可以访问这些表，因此如果在一个单独的连接上所执行的 dbmlsync 存储过程需要访问任何同步表，则这些存储过程将无法执行。

有关在一个单独连接上执行的挂接的信息，请参见“SQL Anywhere 客户端的事件挂接” 第 225 页。

此选项有一个长格式和一个短格式：您可以使用 **lt**，也可以使用 **LockTables**。

您还可以在数据库中存储扩展选项。有关 dbmlsync 扩展选项的详细信息，请参见“dbmlsync 扩展选项简介”一节第 177 页。

示例

以下 dbmlsync 命令行说明在启动 dbmlsync 时如何设置此选项：

```
dbmlsync -e "lt=on"
```

以下 SQL 语句说明如何在数据库中存储此选项：

```
CREATE SYNCHRONIZATION SUBSCRIPTION
   TO sales_publication
   FOR ml_user1
   OPTION lt='on';
```


Memory (mem) 扩展选项

指定构建上载时 dbmlsync 所使用的高速缓存大小。

语法

mem=*number*[**K** | **M**]; ...

注释

指定用于构建上载的高速缓存大小（以字节为单位）。高速缓存较大表示 dbmlsync 可在内存中保留更多的数据页，减少磁盘读取/写入量并提高性能。

使用后缀 k 或 m 分别指定以千字节或兆字节为单位。缺省值为 **1M**。

此选项有一个长格式和一个短格式：您可以使用 **mem**，也可以使用 **Memory**。

您还可以在数据库中存储扩展选项。有关 dbmlsync 扩展选项的详细信息，请参见“dbmlsync 扩展选项简介”一节第 177 页。

另请参见

- “-urc 选项”一节第 169 页
- “性能提示”一节 《MobiLink - 服务器管理》

示例

以下 dbmlsync 命令行说明在启动 dbmlsync 时如何设置此选项：

```
dbmlsync -e "mem=2M"
```

以下 SQL 语句说明如何在数据库中存储此选项：

```
CREATE SYNCHRONIZATION SUBSCRIPTION
   TO sales_publication
   FOR ml_user1
   OPTION mem='2m';
```


MirrorLogDirectory (mld) 扩展选项

指定旧事务日志镜像文件的位置，以便能够删除它们。

语法

mld=*filename*; ...

注释

此选项使 dbmlsync 能够在出现以下两种情形中的任意一种时，删除旧的事务日志镜像文件：

- 脱机事务日志镜像与事务日志镜像不在同一目录中

  或者

- dbmlsync 与远程数据库服务器不在同一台计算机上运行

在常规设置中，活动事务日志镜像与重命名的事务日志镜像文件位于同一目录中，并且 dbmlsync 与远程数据库在同一台计算机上运行，因此不需要此选项，旧的事务日志镜像文件也会被自动删除。

仅当 delete_old_logs 数据库选项设置为 On、Delay 或 *n* 天时，此目录中的事务日志才会受影响。

此选项有一个长格式和一个短格式：您可以使用 **mld**，也可以使用 **MirrorLogDirectory**。

您还可以在数据库中存储扩展选项。有关 dbmlsync 扩展选项的详细信息，请参见“dbmlsync 扩展选项简介”一节第 177 页。

另请参见

- “delete_old_logs 选项 [MobiLink 客户端] [SQL Remote] [复制代理]”一节 《SQL Anywhere 服务器 - 数据库管理》

示例

以下 dbmlsync 命令行说明在启动 dbmlsync 时如何设置此选项：

```
dbmlsync -e "mld=c:\tmp\file"
```

以下 SQL 语句说明如何在数据库中存储此选项：

```
CREATE SYNCHRONIZATION SUBSCRIPTION
   TO sales_publication
   FOR ml_user1
   OPTION mld='c:\tmp\file';
```


MobiLinkPwd (mp) 扩展选项

指定 MobiLink 口令。

语法

mp=*password*; ...

注释

指定用于连接的口令。此口令应为要同步的预订所属的 MobiLink 用户的正确口令。此用户可使用 dbmlsync -u 选项加以指定。缺省值为空值。

如果 MobiLink 用户已有口令，则使用扩展选项 **-e mn** 来进行更改。

此选项有一个长格式和一个短格式：您可以使用 **mp**，也可以使用 **MobiLinkPwd**。

您还可以在数据库中存储扩展选项。有关 dbmlsync 扩展选项的详细信息，请参见“dbmlsync 扩展选项简介”一节第 177 页。

另请参见

- “NewMobiLinkPwd (mn) 扩展选项”一节第 198 页
- “-mn 选项”一节第 149 页
- “-mp 选项”一节第 150 页

示例

以下 dbmlsync 命令行说明在启动 dbmlsync 时如何设置此选项：

```
dbmlsync -e "mp=password"
```

以下 SQL 语句说明如何在数据库中存储此选项：

```
CREATE SYNCHRONIZATION SUBSCRIPTION
   TO sales_publication
   FOR ml_user1
   OPTION mp='SQL';
```


NewMobiLinkPwd (mn) 扩展选项

指定新的口令。

语法

mn=*new-password*; ...

注释

为要同步的预订所属的 MobiLink 用户指定新口令。在想要更改现有口令时，可使用此选项。缺省设置为不更改口令。

此选项有一个长格式和一个短格式：您可以使用 **mn**，也可以使用 **NewMobiLinkPwd**。

您还可以在数据库中存储扩展选项。有关 dbmlsync 扩展选项的详细信息，请参见“dbmlsync 扩展选项简介”一节第 177 页。

另请参见

- “MobiLinkPwd (mp) 扩展选项”一节第 197 页
- “-mn 选项”一节第 149 页
- “-mp 选项”一节第 150 页

示例

以下 dbmlsync 命令行说明在启动 dbmlsync 时如何设置此选项：

```
dbmlsync -e "mp=oldpassword; mn=newpassword"
```

以下 SQL 语句说明如何在数据库中存储此选项：

```
CREATE SYNCHRONIZATION SUBSCRIPTION
   TO sales_publication
   FOR ml_user1
   OPTION mn='SQL';
```


NoSyncOnStartup (nss) 扩展选项

防止 dbmlsync 在启动时同步，此时将有一个调度选项以其它方式引发同步。

语法

nss={ on | off }; ...

注释

仅当 EVERY 或 INFINITE 子句与调度配合使用时，此选项才会生效。这些调度选项会使 dbmlsync 在启动时自动同步。

缺省值是 off。

如果将 NoSyncOnStartup 设置为 on 并将调度与 INFINITE 子句配合使用，则同步将在收到窗口消息后发生。

如果将 NoSyncOnStartup 设置为 on 并将调度与 EVERY 子句配合使用，则启动后的第一次同步将在 EVERY 子句中所指定的时间之后发生。

除了 dbmlsync 启动以外，此设置在任何方面都不会影响调度的行为。

此选项有一个长格式和一个短格式：您可以使用 **nss**，也可以使用 **NoSyncOnStartup**。

您还可以在数据库中存储扩展选项。有关 dbmlsync 扩展选项的详细信息，请参见“dbmlsync 扩展选项简介”一节第 177 页。

另请参见

- “Schedule (sch) 扩展选项”一节第 202 页
- “调度同步”一节第 114 页

示例

以下部分 dbmlsync 命令行说明在启动 dbmlsync 时如何设置此选项：

```
dbmlsync -e "schedule=EVERY:01:00;nss=off"...
```

以下 SQL 语句说明如何在数据库中存储此选项：

```
CREATE SYNCHRONIZATION SUBSCRIPTION
   TO sales_publication
   FOR ml_user1
   OPTION nss='off', schedule='EVERY:01:00';
```


OfflineDirectory (dir) 扩展选项

指定包含脱机事务日志的路径。

语法

dir=*path*; ...

注释

缺省情况下，dbmlsync 在联机事务日志的目录中检查重命名的日志。仅当重命名的脱机事务日志位于其它目录中时才需要指定此选项。

此选项有一个长格式和一个短格式：您可以使用 **dir**，也可以使用 **OfflineDirectory**。

您还可以在数据库中存储扩展选项。有关 dbmlsync 扩展选项的详细信息，请参见“dbmlsync 扩展选项简介”一节第 177 页。

示例

以下 dbmlsync 命令行说明在启动 dbmlsync 时如何设置此选项：

```
dbmlsync -e "dir=c:\db\logs"
```

以下 SQL 语句说明如何在数据库中存储此选项：

```
CREATE SYNCHRONIZATION SUBSCRIPTION
   TO sales_publication
   FOR ml_user1
   OPTION dir='c:\db\logs';
```


PollingPeriod (pp) 扩展选项

指定日志扫描轮询周期。

语法

pp=*number*[**S** | **M** | **H** | **D**]; ...

注释

此选项指定日志扫描间的时间间隔。使用后缀 s、m、h 或 d 分别指定以秒、分钟、小时或天为单位。缺省值为 **1** 分钟。如果不指定后缀，则缺省时间单位为分钟。

仅当您调度同步或使用 sp_hook_dbmlsync_delay 挂接时，才会发生日志扫描轮询。

有关日志扫描轮询的说明，请参见“DisablePolling (p) 扩展选项”一节第 183 页。

此选项与 **dbmlsync -pp** 相同。

此选项有一个长格式和一个短格式：您可以使用 **pp**，也可以使用 **PollingPeriod**。

您还可以在数据库中存储扩展选项。有关 dbmlsync 扩展选项的详细信息，请参见“dbmlsync 扩展选项简介”一节第 177 页。

另请参见

- “DisablePolling (p) 扩展选项”一节第 183 页
- “-pp 选项”一节第 159 页
- “-p 选项”一节第 155 页
- “sp_hook_dbmlsync_delay”一节第 241 页

示例

以下 dbmlsync 命令行说明在启动 dbmlsync 时如何设置此选项：

```
dbmlsync -e "pp=5"
```

以下 SQL 语句说明如何在数据库中存储此选项：

```
CREATE SYNCHRONIZATION SUBSCRIPTION
   TO sales_publication
   FOR ml_user1
   OPTION pp='5';
```


Schedule (sch) 扩展选项

为同步指定调度。

语法

sch=*schedule*; ...

schedule : { **EVERY**:*hhhh*:*mm* | **INFINITE** | *singleSchedule* }

hhhh : **00** ... **9999**

mm : **00** ... **59**

singleSchedule : *day* **@***hh*:*mm*[**AM** | **PM**] [**-***hh*:*mm*[**AM** | **PM**]] ,...

hh : **00** ... **24**

mm : **00** ... **59**

day :
EVERYDAY | **WEEKDAY** | **MON** | **TUE** | **WED** | **THU** | **FRI** | **SAT** | **SUN** | *dayOfMonth*

dayOfMonth : **0**... **31**

参数

EVERY EVERY 关键字会使同步在启动时发生，然后在指定的时间段后无限期重复进行。如果同步过程比指定周期长，则同步立即再次启动。

要避免在 dbmlsync 启动时发生同步，请使用扩展选项 NoSyncOnStartup。请参见“NoSyncOnStartup (nss) 扩展选项”一节第 199 页。

singleSchedule 如果提供一个或多个单独的调度，则同步只在指定的日期和时间发生。

间隔可以指定为 *@hh*:*mm*-*hh*:*mm*（AM 或 PM 的指定可选）。如果不指定 AM 或 PM，则假定为 24 小时制。24:00 被解释为第二天的 00:00。如果指定间隔，则同步在间隔时间内的随机时间开始。间隔提供了一个同步时间窗口，这样具有相同调度的多个远程数据库不会正好在同一时间同步，从而避免了 MobiLink 服务器上发生拥塞。

间隔的结束时间总是被解释为位于开始时间后。如果时间间隔包含午夜，则它在第二天结束。如果 dbmlsync 在时间间隔的中间启动，则同步在结束时间前的随机时间发生。

EVERYDAY EVERYDAY 指的是一个星期内的所有七天。

WEEKDAY WEEKDAY 指的是从星期一到星期五。

周几指的是周一、周二等等。您还可以使用完整形式的日期，例如 Monday。当您所使用的语言不是英语，不是连接字符串中客户端请求的语言，也不是服务器窗口中出现的语言时，您必须使用完整形式的日期名称。

dayOfMonth 要指定为一个月中的最后一天而不管一个月有多长，请将 *dayOfMonth* 设置为 0。

INFINITE 关键字 INFINITE 可使 dbmlsync 在启动时同步，然后在将窗口消息发送到 dbmlsync 的另一程序启动同步后再次同步。在等待时，dbmlsync 将定期运行和扫描日志。可以使用 dbmlsync 扩展选项 NoSyncOnStartup 来避免初始同步。

有关详细信息，请参见“NoSyncOnStartup (nss) 扩展选项”一节第 199 页。

此选项可与 dbmlsync -wc 选项结合使用，以唤醒 dbmlsync 并执行同步。

有关详细信息，请参见“-wc 选项”一节第 172 页。

注释

如果在调度时间上一次同步仍未完成，则调度的同步将在上一次同步完成时开始。

缺省情况下，没有调度。

此选项有一个长格式和一个短格式：您可以使用 **sch**，也可以使用 **Schedule**。

您还可以在数据库中存储扩展选项。有关 dbmlsync 扩展选项的详细信息，请参见“dbmlsync 扩展选项简介”一节第 177 页。

调度选项语法与同步 SQL 语句和 dbmlsync 命令行中使用的语法相同。

IgnoreScheduling 扩展选项和 -is 选项指示 dbmlsync 忽略调度，从而使同步立即执行。有关详细信息，请参见“IgnoreScheduling (isc) 扩展选项”一节第 192 页。

有关调度的详细信息，请参见“调度同步”一节第 114 页。

示例

以下 dbmlsync 命令行说明在启动 dbmlsync 时如何设置此选项：

```
dbmlsync -e "sch=WEEKDAY@8:00am,SUN@9:00pm"
```

以下 SQL 语句说明如何在数据库中存储此选项：

```
CREATE SYNCHRONIZATION SUBSCRIPTION
   TO sales_publication
   FOR ml_user1
   OPTION sch='WEEKDAY@8:00am,SUN@9:00pm';
```


ScriptVersion (sv) 扩展选项

指定脚本版本。

语法

sv=*version-name*; ...

注释

脚本版本确定在同步期间 MobiLink 在统一数据库上运行的脚本。缺省脚本版本名为 **default**。

此选项有一个长格式和一个短格式：您可以使用 **sv**，也可以使用 **ScriptVersion**。

您还可以在数据库中存储扩展选项。有关 dbmlsync 扩展选项的详细信息，请参见“dbmlsync 扩展选项简介”一节第 177 页。

示例

以下 dbmlsync 命令行说明在启动 dbmlsync 时如何设置此选项：

```
dbmlsync -e "sv=SyaAd001"
```

以下 SQL 语句说明如何在数据库中存储此选项：

```
CREATE SYNCHRONIZATION SUBSCRIPTION
   TO sales_publication
   FOR ml_user1
   OPTION sv='SysAd001';
```


SendColumnNames (scn) 扩展选项

指定应在上载中发送列名，以便在进行行处理时直接使用。

语法

scn={ ON | OFF }; ...

注释

MobiLink 服务器使用列名进行直接行处理。使用行处理 API 按名称而不按索引引用列时，应设置此选项。这是此选项所发送的列名的唯一用途。

请参见“直接行处理” 《MobiLink - 服务器管理》。

缺省值为 **OFF**。

此选项有一个长格式和一个短格式：您可以使用 **scn**，也可以使用 **SendColumnNames**。

您还可以在数据库中存储扩展选项。有关 dbmlsync 扩展选项的详细信息，请参见“dbmlsync 扩展选项简介”一节第 177 页。

示例

以下 dbmlsync 命令行说明在启动 dbmlsync 时如何设置此选项：

```
dbmlsync -e "scn=on"
```

以下 SQL 语句说明如何在数据库中存储此选项：

```
CREATE SYNCHRONIZATION SUBSCRIPTION
   TO sales_publication
   FOR ml_user1
   OPTION scn='on';
```


SendDownloadACK (sa) 扩展选项

指定客户端应该向服务器发送一个下载确认。

语法

sa={ ON | OFF }; ...

注释

建议将 SendDownloadAck 设置为 OFF。如果下载失败，则远程数据库将重新上载相同的时间戳，这样便不会丢失数据。此选项用于更改服务器端脚本的行为。请参见“nonblocking_download_ack 连接事件”一节 《MobiLink - 服务器管理》。

有关通过关闭下载确认提高性能的详细信息，请参见“使用非阻塞下载确认”一节 《MobiLink - 服务器管理》。

注意：当 SendDownloadAck 设置为 ON 且处于详细模式下时，会在客户端日志中写入一个确认行。

缺省值为 **OFF**。

此选项有一个长格式和一个短格式：您可以使用 **sa**，也可以使用 **SendDownloadACK**。

您还可以在数据库中存储扩展选项。有关 dbmlsync 扩展选项的详细信息，请参见“dbmlsync 扩展选项简介”一节第 177 页。

示例

以下 dbmlsync 命令行说明在启动 dbmlsync 时如何设置此选项：

```
dbmlsync -e "sa=off"
```

以下 SQL 语句说明如何在数据库中存储此选项：

```
CREATE SYNCHRONIZATION SUBSCRIPTION
   TO sales_publication
   FOR ml_user1
   OPTION sa='off';
```


SendTriggers (st) 扩展选项

指定上载时应发送触发器操作。

语法

st={ ON | OFF }; ...

注释

级联删除同样被视为触发器操作。

缺省值为 **OFF**。

此选项有一个长格式和一个短格式：您可以使用 **st**，也可以使用 **SendTriggers**。

如果两个发布重叠，即它们包含了一个或多个相同的表，则必须使用相同的 SendTriggers 选项设置来同步这两个发布。

您还可以在数据库中存储扩展选项。有关 dbmlsync 扩展选项的详细信息，请参见“dbmlsync 扩展选项简介”一节第 177 页。

示例

以下 dbmlsync 命令行说明在启动 dbmlsync 时如何设置此选项：

```
dbmlsync -e "st=on"
```

以下 SQL 语句说明如何在数据库中存储此选项：

```
CREATE SYNCHRONIZATION SUBSCRIPTION
   TO sales_publication
   FOR ml_user1
   OPTION st='on';
```


TableOrder (tor) 扩展选项

指定上载中的表顺序。

语法

tor=*tables*; ...

tables = *table-name* [,*table-name*], ...

注释

此选项允许您在远程数据库上指定即将上载的表的顺序。以逗号分隔的列表形式指定 *tables*。必须指定所有要上载的表。如果所包含的表未包括在同步中，则会忽略这些表。

指定的表顺序必须确保参照完整性。这意味着，如果 Table1 具有对 Table2 的外键引用，则 Table2 必须在 Table1 前上载。如果不按适当的顺序指定表，则会出错，但以下两种情况除外：

- 设置 TableOrderChecking=OFF。
- 您的表具有循环外键关系。（在这种情况下，没有任何顺序符合规则，因此可按任意顺序上载循环中所涉及的表。）

如果未指定 TableOrder，则 dbmlsync 将选择一个满足参照完整性的顺序。

下载中的表顺序与上载中的相同。控制上载表顺序可使编写服务器端脚本变得更加简单，在远程数据库和统一数据库具有不同外键约束时尤其如此。

不存在任何必须使用此选项的情况。此选项可供那些想要确保表按特定顺序上载的用户使用。

此选项有一个长格式和一个短格式：您可以使用 **tor**，也可以使用 **TableOrder**。

您还可以在数据库中存储扩展选项。有关 dbmlsync 扩展选项的详细信息，请参见“dbmlsync 扩展选项简介”一节第 177 页。

另请参见

- “TableOrderChecking (toc) 扩展选项”一节第 209 页
- “如何处理上载”一节 《MobiLink - 入门》
- “参照完整性与同步”一节 《MobiLink - 入门》

示例

以下 dbmlsync 命令行说明在启动 dbmlsync 时如何设置此选项：

```
dbmlsync -e "tor=admin,parent,child"
```

以下 SQL 语句说明如何在数据库中存储此选项：

```
CREATE SYNCHRONIZATION SUBSCRIPTION
   TO sales_publication
   FOR ml_user1
   OPTION tor='admin,parent,child';
```


TableOrderChecking (toc) 扩展选项

在您指定 TableOrder 时，确定 dbmlsync 是否应检查在上载另一个具有外键的表之前未上载任何表。

语法

tor={ OFF | ON }; ...

注释

在大多数应用程序中，远程数据库和统一数据库上的表具有相同的外键关系。这种情况下，您应将 TableOrderChecking 设为缺省值 ON，这样 dbmlsync 会确保在上载另一个具有外键的表之前未上载任何表。这可确保参照完整性。

在统一和远程数据库具有不同的外键关系时，该选项非常有用。将该选项与 TableOrder 扩展选项配合使用，以指定不遵循在上载另一个具有外键的表之前未上载任何表这一规则的表的顺序。

此选项仅在指定了 TableOrder 扩展选项时才会有用。

缺省值为 **ON**。

此选项有一个长格式和一个短格式：您可以使用 **toc**，也可以使用 **TableOrderChecking**。

您还可以在数据库中存储扩展选项。有关 dbmlsync 扩展选项的详细信息，请参见“dbmlsync 扩展选项简介”一节第 177 页。

另请参见

- “TableOrder (tor) 扩展选项”一节第 208 页
- “如何处理上载”一节 《MobiLink - 入门》
- “参照完整性与同步”一节 《MobiLink - 入门》

示例

以下 dbmlsync 命令行说明在启动 dbmlsync 时如何设置此选项：

```
dbmlsync -e "toc=OFF"
```

以下 SQL 语句说明如何在数据库中存储此选项：

```
CREATE SYNCHRONIZATION SUBSCRIPTION
   TO sales_publication
   FOR ml_user1
   OPTION toc='Off';
```


UploadOnly (uo) 扩展选项

指定同步应只包括上载。

语法

uo={ ON | OFF }; ...

注释

在仅上载同步期间，dbmlsync 会准备并发送一个上载到 MobiLink 服务器，所采用的方式与正常的完全同步完全相同。但是，MobiLink 只发送一个确认来指示上载是否成功提交，而不会回发下载。

有关必须为仅上载同步定义的脚本列表，请参见“必需的脚本”一节 《MobiLink - 服务器管理》。

缺省值为 **OFF**。

此选项有一个长格式和一个短格式：您可以使用 **uo**，也可以使用 **UploadOnly**。

您还可以在数据库中存储扩展选项。有关 dbmlsync 扩展选项的详细信息，请参见“dbmlsync 扩展选项简介”一节第 177 页。

另请参见

- “仅上载同步和仅下载同步”一节 《MobiLink - 服务器管理》
- “DownloadOnly (ds) 扩展选项”一节第 185 页

示例

以下 dbmlsync 命令行说明在启动 dbmlsync 时如何设置此选项：

```
dbmlsync -e "uo=on"
```

以下 SQL 语句说明如何在数据库中存储此选项：

```
CREATE SYNCHRONIZATION SUBSCRIPTION
   TO sales_publication
   FOR ml_user1
   OPTION uo='on';
```


Verbose (v) 扩展选项

指定最高详细程度。

语法

v={ ON | OFF }; ...

注释

此选项指定的高详细程度可能会影响性能，通常应该仅在开发阶段使用。

此选项与 **dbmlsync -v+** 相同。如果您同时指定 -v 选项和扩展选项，并且它们之间有冲突，则 -v 选项将覆盖扩展选项。如果没有冲突，详细程度记录选项是叠加的—使用您指定的所有选项。如果记录详细程度由扩展选项设置，记录不会立即生效，因此不记录启动信息。到第一次同步时，-v 选项和扩展选项之间的记录行为就完全一样了。

有关详细信息，请参见“-v 选项”一节第 171 页。

缺省值为 **OFF**。

此选项有一个长格式和一个短格式：您可以使用 **v**，也可以使用 **Verbose**。

您还可以在数据库中存储扩展选项。有关 dbmlsync 扩展选项的详细信息，请参见“dbmlsync 扩展选项简介”一节第 177 页。

另请参见

- “VerboseHooks (vs) 扩展选项”一节第 212 页
- “VerboseMin (vm) 扩展选项”一节第 213 页
- “VerboseOptions (vo) 扩展选项”一节第 214 页
- “VerboseRowCounts (vn) 扩展选项”一节第 215 页
- “VerboseRowValues (vr) 扩展选项”一节第 216 页
- “VerboseUpload (vu) 扩展选项”一节第 217 页

示例

以下 dbmlsync 命令行说明在启动 dbmlsync 时如何设置此选项：

```
dbmlsync -e "v=on"
```

以下 SQL 语句说明如何在数据库中存储此选项：

```
CREATE SYNCHRONIZATION SUBSCRIPTION
   TO sales_publication
   FOR ml_user1
   OPTION v='on';
```


VerboseHooks (vs) 扩展选项

指定应记录与挂接脚本有关的消息。

语法

vs={ ON | OFF }; ...

注释

此选项与 **dbmlsync -vs** 相同。如果您同时指定 -v 选项和扩展选项，并且它们之间有冲突，则 -v 选项取代扩展选项。如果没有冲突，详细程度记录选项是叠加的—使用您指定的所有选项。如果记录详细程度由扩展选项设置，记录不会立即生效，因此不记录启动信息。到第一次同步时，-v 选项和扩展选项之间的记录行为就完全一样了。

有关详细信息，请参见“-v 选项”一节第 171 页。

缺省值为 **OFF**。

此选项有一个长格式和一个短格式：您可以使用 **vs**，也可以使用 **VerboseHooks**。

您还可以在数据库中存储扩展选项。有关 dbmlsync 扩展选项的详细信息，请参见“dbmlsync 扩展选项简介”一节第 177 页。

另请参见

- “SQL Anywhere 客户端的事件挂接”第 225 页
- “Verbose (v) 扩展选项”一节第 211 页
- “VerboseMin (vm) 扩展选项”一节第 213 页
- “VerboseOptions (vo) 扩展选项”一节第 214 页
- “VerboseRowCounts (vn) 扩展选项”一节第 215 页
- “VerboseRowValues (vr) 扩展选项”一节第 216 页
- “VerboseUpload (vu) 扩展选项”一节第 217 页

示例

以下 dbmlsync 命令行说明在启动 dbmlsync 时如何设置此选项：

```
dbmlsync -e "vs=on"
```

以下 SQL 语句说明如何在数据库中存储此选项：

```
CREATE SYNCHRONIZATION SUBSCRIPTION
   TO sales_publication
   FOR ml_user1
   OPTION vs='on';
```


VerboseMin (vm) 扩展选项

指定应该记录少量信息。

语法

vm={ ON | OFF }; ...

注释

此选项与 **dbmlsync -v** 相同。如果您同时指定 -v 选项和扩展选项，并且它们之间有冲突，则 -v 选项取代扩展选项。如果没有冲突，详细程度记录选项是叠加的—使用您指定的所有选项。如果记录详细程度由扩展选项设置，记录不会立即生效，因此不记录启动信息。到第一次同步时，-v 选项和扩展选项之间的记录行为就完全一样了。

有关详细信息，请参见“-v 选项”一节第 171 页。

缺省值为 **OFF**。

此选项有一个长格式和一个短格式：您可以使用 **vm**，也可以使用 **VerboseMin**。

您还可以在数据库中存储扩展选项。有关 dbmlsync 扩展选项的详细信息，请参见“dbmlsync 扩展选项简介”一节第 177 页。

另请参见

- “Verbose (v) 扩展选项”一节第 211 页
- “Verbose (v) 扩展选项”一节第 211 页
- “VerboseOptions (vo) 扩展选项”一节第 214 页
- “VerboseRowCounts (vn) 扩展选项”一节第 215 页
- “VerboseRowValues (vr) 扩展选项”一节第 216 页
- “VerboseUpload (vu) 扩展选项”一节第 217 页

示例

以下 dbmlsync 命令行说明在启动 dbmlsync 时如何设置此选项：

```
dbmlsync -e "vm=on"
```

以下 SQL 语句说明如何在数据库中存储此选项：

```
CREATE SYNCHRONIZATION SUBSCRIPTION
   TO sales_publication
   FOR ml_user1
   OPTION vm='on';
```


VerboseOptions (vo) 扩展选项

指定应记录与指定的命令行选项（包括扩展选项）有关的信息。

语法

vo={ ON | OFF }; ...

注释

此选项与 **dbmlsync -vo** 相同。如果您同时指定 -v 选项和扩展选项，并且它们之间有冲突，则 -v 选项取代扩展选项。如果没有冲突，详细程度记录选项是叠加的—使用您指定的所有选项。如果记录详细程度由扩展选项设置，记录不会立即生效，因此不记录启动信息。到第一次同步时，-v 选项和扩展选项之间的记录行为就完全一样了。

有关详细信息，请参见“-v 选项”一节第 171 页。

缺省值为 **OFF**。

此选项有一个长格式和一个短格式：您可以使用 **vo**，也可以使用 **VerboseOptions**。

您还可以在数据库中存储扩展选项。有关 dbmlsync 扩展选项的详细信息，请参见“dbmlsync 扩展选项简介”一节第 177 页。

另请参见

- “Verbose (v) 扩展选项”一节第 211 页
- “Verbose (v) 扩展选项”一节第 211 页
- “VerboseMin (vm) 扩展选项”一节第 213 页
- “VerboseRowCounts (vn) 扩展选项”一节第 215 页
- “VerboseRowValues (vr) 扩展选项”一节第 216 页
- “VerboseUpload (vu) 扩展选项”一节第 217 页

示例

以下 dbmlsync 命令行说明在启动 dbmlsync 时如何设置此选项：

```
dbmlsync -e "vo=on"
```

以下 SQL 语句说明如何在数据库中存储此选项：

```
CREATE SYNCHRONIZATION SUBSCRIPTION
   TO sales_publication
   FOR ml_user1
   OPTION vo='on';
```


VerboseRowCounts (vn) 扩展选项

指定应记录上载和下载的行的数目。

语法

vn={ ON | OFF }; ...

注释

此选项与 **dbmlsync -vn** 相同。如果您同时指定 -v 选项和扩展选项，并且它们之间有冲突，则 -v 选项取代扩展选项。如果没有冲突，详细程度记录选项是叠加的—使用您指定的所有选项。如果记录详细程度由扩展选项设置，记录不会立即生效，因此不记录启动信息。到第一次同步时，-v 选项和扩展选项之间的记录行为就完全一样了。

有关详细信息，请参见“-v 选项”一节第 171 页。

缺省值为 **OFF**。

此选项有一个长格式和一个短格式：您可以使用 **vn**，也可以使用 **VerboseRowCounts**。

您还可以在数据库中存储扩展选项。有关 dbmlsync 扩展选项的详细信息，请参见“dbmlsync 扩展选项简介”一节第 177 页。

另请参见

- “Verbose (v) 扩展选项”一节第 211 页
- “Verbose (v) 扩展选项”一节第 211 页
- “VerboseMin (vm) 扩展选项”一节第 213 页
- “VerboseOptions (vo) 扩展选项”一节第 214 页
- “VerboseRowValues (vr) 扩展选项”一节第 216 页
- “VerboseUpload (vu) 扩展选项”一节第 217 页

示例

以下 dbmlsync 命令行说明在启动 dbmlsync 时如何设置此选项：

```
dbmlsync -e "vn=on"
```

以下 SQL 语句说明如何在数据库中存储此选项：

```
CREATE SYNCHRONIZATION SUBSCRIPTION
   TO sales_publication
   FOR ml_user1
   OPTION vn='on';
```


VerboseRowValues (vr) 扩展选项

指定应记录上载和下载的行的值。

语法

vr={ ON | OFF }; ...

注释

此选项与 **dbmlsync -vr** 相同。如果您同时指定 -v 选项和扩展选项，并且它们之间有冲突，则 -v 选项取代扩展选项。如果没有冲突，详细程度记录选项是叠加的—使用您指定的所有选项。如果记录详细程度由扩展选项设置，记录不会立即生效，因此不记录启动信息。到第一次同步时，-v 选项和扩展选项之间的记录行为就完全一样了。

有关详细信息，请参见“-v 选项”一节第 171 页。

缺省值为 **OFF**。

此选项有一个长格式和一个短格式：您可以使用 **vr**，也可以使用 **VerboseRowValues**。

您还可以在数据库中存储扩展选项。有关 dbmlsync 扩展选项的详细信息，请参见“dbmlsync 扩展选项简介”一节第 177 页。

另请参见

- “Verbose (v) 扩展选项”一节第 211 页
- “Verbose (v) 扩展选项”一节第 211 页
- “VerboseMin (vm) 扩展选项”一节第 213 页
- “VerboseOptions (vo) 扩展选项”一节第 214 页
- “VerboseRowCounts (vn) 扩展选项”一节第 215 页
- “VerboseUpload (vu) 扩展选项”一节第 217 页

示例

以下 dbmlsync 命令行说明在启动 dbmlsync 时如何设置此选项：

```
dbmlsync -e "vr=on"
```

以下 SQL 语句说明如何在数据库中存储此选项：

```
CREATE SYNCHRONIZATION SUBSCRIPTION
   TO sales_publication
   FOR ml_user1
   OPTION vr='on';
```


VerboseUpload (vu) 扩展选项

指定有关应该记录的上载流的信息。

语法

vu={ ON | OFF }; ...

注释

此选项与 **dbmlsync -vu** 相同。如果您同时指定 -v 选项和扩展选项，并且它们之间有冲突，则 -v 选项取代扩展选项。如果没有冲突，详细程度记录选项是叠加的—使用您指定的所有选项。如果记录详细程度由扩展选项设置，记录不会立即生效，因此不记录启动信息。到第一次同步时，-v 选项和扩展选项之间的记录行为就完全一样了。

有关详细信息，请参见“-v 选项”一节第 171 页。

缺省值为 **OFF**。

此选项有一个长格式和一个短格式：您可以使用 **vu**，也可以使用 **VerboseUpload**。

您还可以在数据库中存储扩展选项。有关 dbmlsync 扩展选项的详细信息，请参见“dbmlsync 扩展选项简介”一节第 177 页。

另请参见

- “Verbose (v) 扩展选项”一节第 211 页
- “Verbose (v) 扩展选项”一节第 211 页
- “VerboseMin (vm) 扩展选项”一节第 213 页
- “VerboseOptions (vo) 扩展选项”一节第 214 页
- “VerboseRowCounts (vn) 扩展选项”一节第 215 页
- “VerboseRowValues (vr) 扩展选项”一节第 216 页

示例

以下 dbmlsync 命令行说明在启动 dbmlsync 时如何设置此选项：

```
dbmlsync -e "vu=on"
```

以下 SQL 语句说明如何在数据库中存储此选项：

```
CREATE SYNCHRONIZATION SUBSCRIPTION
   TO sales_publication
   FOR ml_user1
   OPTION vu='on';
```


MobiLink SQL 语句

目录

MobiLink 语句

以下是用于配置和运行 MobiLink SQL Anywhere 客户端的 SQL 语句：

- “ALTER PUBLICATION 语句 [MobiLink] [SQL Remote]”一节 《SQL Anywhere 服务器 - SQL 参考》
- “ALTER SYNCHRONIZATION PROFILE 语句 [MobiLink]”一节 《SQL Anywhere 服务器 - SQL 参考》
- “ALTER SYNCHRONIZATION SUBSCRIPTION 语句 [MobiLink]”一节 《SQL Anywhere 服务器 - SQL 参考》
- “ALTER SYNCHRONIZATION USER 语句 [MobiLink]”一节 《SQL Anywhere 服务器 - SQL 参考》
- “CREATE PUBLICATION 语句 [MobiLink] [SQL Remote]”一节 《SQL Anywhere 服务器 - SQL 参考》
- “CREATE SYNCHRONIZATION PROFILE 语句 [MobiLink]”一节 《SQL Anywhere 服务器 - SQL 参考》
- “CREATE SYNCHRONIZATION SUBSCRIPTION 语句 [MobiLink]”一节 《SQL Anywhere 服务器 - SQL 参考》
- “CREATE SYNCHRONIZATION USER 语句 [MobiLink]”一节 《SQL Anywhere 服务器 - SQL 参考》
- “DROP PUBLICATION 语句 [MobiLink] [SQL Remote]”一节 《SQL Anywhere 服务器 - SQL 参考》
- “DROP SYNCHRONIZATION PROFILE 语句 [MobiLink]”一节 《SQL Anywhere 服务器 - SQL 参考》
- “DROP SYNCHRONIZATION SUBSCRIPTION 语句 [MobiLink]”一节 《SQL Anywhere 服务器 - SQL 参考》
- “DROP SYNCHRONIZATION USER 语句 [MobiLink]”一节 《SQL Anywhere 服务器 - SQL 参考》
- “START SYNCHRONIZATION DELETE 语句 [MobiLink]”一节 《SQL Anywhere 服务器 - SQL 参考》
- “STOP SYNCHRONIZATION DELETE 语句 [MobiLink]”一节 《SQL Anywhere 服务器 - SQL 参考》

UltraLite 客户端

请参见“UltraLite SQL 语句” 《UltraLite - 数据库管理和参考》。

MobiLink 同步配置文件

目录

MobiLink 同步配置文件

同步配置文件允许您将某些 dbmlsync 选项放置在数据库中。您创建的同步配置文件可以包含各种同步选项。

创建一个同步配置文件后，如果想使用该文件，请使用 dbmlsync -sp 选项并指定该同步配置文件的名称。同步配置文件中所指定的选项将与您所指定的命令行选项合并。如果在命令行和同步配置文件中指定了等同的选项，那么命令行上的选项将会替换在配置文件中所指定的选项。请参见“-sp 选项”一节第 164 页。

可使用以下语句创建、更改和删除同步配置文件：

- “ALTER SYNCHRONIZATION PROFILE 语句 [MobiLink]”一节 《SQL Anywhere 服务器 - SQL 参考》
- “CREATE SYNCHRONIZATION PROFILE 语句 [MobiLink]”一节 《SQL Anywhere 服务器 - SQL 参考》
- “DROP SYNCHRONIZATION PROFILE 语句 [MobiLink]”一节 《SQL Anywhere 服务器 - SQL 参考》

有关在 UltraLite 中使用同步配置文件的信息，请参见“同步配置文件选项”一节 《UltraLite - 数据库管理和参考》。

可在同步配置文件中指定以下选项：

长选项名	简称	有效值	说明
AuthParms	ap	String	将参数提供给 authenticate_parameters 脚本和验证参数。请参见“-ap 选项”一节第 129 页。
ApplyDnldFile	ba	String	应用一个下载文件。请参见“-ba 选项”一节第 130 页。
ContinueDownload	dc	Boolean	重新启动以前执行失败的下载。请参见“-dc 选项”一节第 136 页。
CreateDnldFile	bc	String	创建一个下载文件。请参见“-bc 选项”一节第 131 页。
DnldFileExtra	be	String	创建下载文件时，此选项用于指定要包含在文件中的额外字符串。请参见“-be 选项”一节第 132 页。
DownloadOnly	ds	Boolean	执行仅下载同步。请参见“-ds 选项”一节第 140 页。
DownloadReadSize	drs	Integer	对于可重新启动的下载，指定发生通信故障后需要重新发送的最大数据量。请参见“-drs 选项”一节第 139 页。
ExtOpt	e	String	指定扩展选项。请参见“-e 选项”一节第 141 页。

长选项名	简称	有效值	说明
IgnoreHookErrors	eh	Boolean	忽略挂接函数中出现的错误。请参见“-eh 选项”一节第 142 页。
IgnoreScheduling	is	Boolean	忽略调度指令，以便立即进行同步。请参见“-is 选项”一节第 146 页。
KillConnections	d	Boolean	删除对远程数据库的冲突锁定。请参见“-d 选项”一节第 135 页。
LogRenameSize	x	后跟 K 或 M 的一个整数。	在扫描事务日志获取上载数据后，重命名并重新启动该事务日志。请参见“-x 选项”一节第 173 页。
MobiLinkPwd	mp	String	提供 MobiLink 用户的口令。请参见“-mp 选项”一节第 150 页。
MLUser	u	String	指定 MobiLink 用户名。请参见“-u 选项”一节第 166 页。
NewMLPassword	mn	String	为 MobiLink 用户提供新口令。在想要更改现有口令时，可使用此选项。请参见“-mn 选项”一节第 149 页。
Ping	pi	Boolean	强制 MobiLink 服务器回应，以确认客户端与 MobiLink 之间的通信。请参见“-pi 选项”一节第 158 页。
Publication	n	String	指定要同步的发布。注意，在同步配置文件中，发布只能指定一次，但可以多次指定命令行选项。请参见“-n 选项”一节第 151 页。
RemoteProgressGreater	ra	Boolean	指定远程偏移大于统一偏移时，使用远程偏移。此选项等同于 -ra 选项。请参见“-r 选项”一节第 162 页。
RemoteProgressLess	rb	Boolean	指定远程偏移小于统一偏移时（例如，通过备份恢复远程数据库时），使用远程偏移。此选项等同于 -rb 选项。请参见“-r 选项”一节第 162 页。
TransactionalUpload	tu	Boolean	指定远程数据库上的每个事务都应在一次同步中作为一个单独的事务上载。请参见“-tu 选项”一节第 165 页。
UpdateGenNum	bg	Boolean	创建下载文件时，此选项可创建能用于尚未同步的远程数据库的文件。请参见“-bg 选项”一节第 133 页。
UploadOnly	uo	Boolean	指定同步仅包含上载，不进行任何下载。请参见“-uo 选项”一节第 168 页。

长选项名	简称	有效值	说明
UploadRowCnt	urc	Integer	指定同步过程中估计要上载的行数。请参见“-urc 选项”一节第 169 页。
Verbosity		String（用逗号分隔的选项列表）	控制 dbmlsync 详细程度。与“-v 选项”一节第 171 页类似。其值必须为下列其中一个或多个选项的用逗号分隔的列表，每个选项对应于一个现有 -v 选项，如下所述： ● BASIC - 等同于 -v ● HIGH - 等同于 -v+ ● CONNECT_STR - 等同于 -vc ● ROW_CNT - 等同于 -vn ● OPTIONS - 等同于 -vo ● ML_PASSWORD - 等同于 -vp ● ROW_DATA - 等同于 -vr ● HOOK - 等同于 -vs

SQL Anywhere 客户端的事件挂接

目录

dbmlsync 挂接简介

SQL Anywhere 同步客户端 dbmlsync 提供了一组可选的事件挂接，您可以使用这些挂接自定义同步过程。如果执行挂接，则将在同步过程中的特定点调用该挂接。

可以通过创建具有特定名称的 SQL 存储过程来执行事件挂接。大部分事件挂接存储过程与同步本身使用同一连接来执行。

可使用事件挂接记录和处理同步事件。例如，可根据逻辑事件调度同步、重试连接故障或处理错误和参照完整性违规。

此外，您还可以使用事件挂接对发布中无法轻松定义的数据子集进行同步。例如，您可以用以下方法对临时表中的数据进行同步：编写一个事件挂接过程在同步之前将数据从临时表复制到一个永久性的表中，再编写另外一个事件挂接过程在同步之后将数据复制回来。

注意

同步过程的完整性取决于一系列内置事务。不得在事件挂接过程中执行隐式或显式的提交或回退。

另外，在挂接中更改连接设置可能会产生意想不到的结果。如果您需要在挂接中更改连接设置，此挂接应在挂接完成前恢复旧值。

dbmlsync 接口

可以将客户端事件挂接与 dbmlsync 命令行实用程序或任何用于同步 SQL Anywhere 客户端的编程接口（包括 dbmlsync API 和 dbmlsync 的 DBTools 接口）配合使用。

请参见“自定义 dbmlsync 同步”一节第 116 页。

同步事件挂接序列

下列伪代码显示可用的事件以及在同步过程期间将在哪一点调用这些事件。例如，sp_hook_dbmlsync_abort 是第一个将要调用的事件挂接。

每个挂接都随实现事件挂接过程时使用的参数值一起提供。在某些情况下，您可以修改此值以返回一个新值，而有些值是只读的。这些参数不是存储过程参数。没有任何参数会传递给任何事件挂接存储过程。实际上这些参数的交换通过读取并修改 #hook_dict 表中的行进行。

例如，sp_hook_dbmlsync_begin 过程有一个 MobiLink 用户参数，其为要同步的 MobiLink 用户。您可以从 #hook_dict 表中检索到此值。

尽管此序列与 MobiLink 服务器的事件序列有相似之处，但在您希望添加到统一数据库与远程数据库的逻辑类型方面几乎没有重叠。因此这两种接口独立且不重复。

如果成功执行了 *_begin 挂接，不管执行 *_begin 挂接后发生任何错误，系统都将调用相应的 *_end 挂接。如果未定义 *_begin 挂接而已定义 *_end 挂接，则除非在通常调用 *_begin 挂接的时间点之前发生错误，否则将调用 *_end 挂接。

如果此挂接更改数据库中的数据，则直到并包括 sp_hook_dbmlsync_logscan_begin 在内的所有的更改将在当前的同步会话中进行同步；之后，将在下一个会话中同步更改。

```
sp_hook_dbmlsync_abort
sp_hook_dbmlsync_set_extended_options
loop until return codes direct otherwise (
   sp_hook_dbmlsync_abort
   sp_hook_dbmlsync_delay
)
sp_hook_dbmlsync_abort
// start synchronization
sp_hook_dbmlsync_begin
// upload events
for each upload segment
// a normal synchronization has one upload segment
// a transactional upload has one segment per transaction
// an incremental upload has one segment per upload piece
 sp_hook_dbmlsync_logscan_begin  //not called for scripted upload
 sp_hook_dbmlsync_logscan_end  //not called for scripted upload
 sp_hook_dbmlsync_set_ml_connect_info //only called during first upload
 sp_hook_dbmlsync_upload_begin
 sp_hook_dbmlsync_set_upload_end_ progress  //only called for scripted upload
 sp_hook_dbmlsync_upload_end
next upload segment
// download events
sp_hook_dbmlsync_validate_download_file (only called
   when -ba option is used)
for each download segment
sp_hook_dbmlsync_download_begin
for each table
   sp_hook_dbmlsync_download_table_begin
   sp_hook_dbmlsync_download_table_end
next table
sp_hook_dbmlsync_download_end

sp_hook_dbmlsync_schema_upgrade
// end synchronization
sp_hook_dbmlsync_end
sp_hook_dbmlsync_process_exit_code
sp_hook_dbmlsync_log_rescan
```

有关上载选项的详细信息，请参见“-tu 选项”一节第 165 页和“Increment (inc) 扩展选项”一节第 193 页。

使用事件挂接过程

本部分介绍设计和使用事件挂接过程的一些考虑事项。

注意

- 不要在事件挂接过程中执行任何 COMMIT 或 ROLLBACK 操作。这些过程与同步在相同的连接上执行，COMMIT 或 ROLLBACK 操作可能会干扰同步。
- 不要更改连接设置。在挂接中更改连接设置可能会产生意想不到的结果。如果您需要在挂接中更改连接设置，此挂接应在挂接完成前恢复旧值。
- 事件挂接连接在调用存储过程时不根据其所有者进行限定。因此，必须由在 dbmlsync 连接上使用的用户名（通常为具有 REMOTE DBA 权限的用户）或包含 dbmlsync 用户作为成员的组 ID 拥有此存储过程。
- 远程数据库对于每个挂接只能有一个实例。不要对一个挂接创建具有不同拥有者的多个实例。

- 挂接程序必须由具有 DBA 权限的用户创建。
- 如果成功执行了 *_begin 挂接，不管执行 *_begin 挂接后发生任何错误，系统都将调用相应的 *_end 挂接。如果未定义 *_begin 挂接而已定义 *_end 挂接，则除非在通常调用 *_begin 挂接的时间点之前发生错误，否则将调用 *_end 挂接。

#hook_dict 表

在即将调用挂接时，dbmlsync 会使用以下的 CREATE 语句在远程数据库中创建 #hook_dict 表。表名前的 # 表示此表为临时表。

```
CREATE TABLE #hook_dict(
name VARCHAR(128) NOT NULL UNIQUE,
value VARCHAR(10240) NOT NULL)
```

dbmlsync 实用程序使用 #hook_dict 表将值传递给挂接函数，而挂接函数使用 #hook_dict 表将值传递回 dbmlsync。

每个挂接都接收参数值。在某些情况下，您可以修改此值以返回一个新值，而有些值是只读的。表中的每一行都包含一个参数值。

例如，对于以下的 dbmlsync 命令行，在调用 sp_hook_dbmlsync_abort 挂接时，#hook_dict 表将包含以下行：

```
dbmlsync -c 'dsn=MyDsn' -n pub1,pub2 -u MyUser
```

#hook_dict 行	值
publication_0	pub1
publication_1	pub2
MobiLink user	MyUser
Abort synchronization	false

中止挂接可以从 #hook_dict 表中检索值，并使用检索到的值对行为进行自定义。例如，您可以使用如下 SELECT 语句检索 MobiLink 用户：

```
SELECT value
FROM #hook_dict
WHERE name = 'MobiLink user'
```

您可以使用挂接更新输入/输出参数，从而修改 dbmlsync 的行为。例如，通过使用如下语句更新表的 abort synchronization 行，您可以使用挂接指示 dbmlsync 中止同步：

```
UPDATE #hook_dict
SET value='true'
WHERE name='abort synchronization'
```

每个挂接的描述都列出了 #hook_dict 表中的行。

示例

以下的示例 sp_hook_dbmlsync_delay 过程说明如何使用 #hook_dict 表来传递参数。此过程只允许在 MobiLink 系统的调度关机时间（18:00 至 19:00）以外进行同步。

```
CREATE PROCEDURE sp_hook_dbmlsync_delay()
BEGIN
   DECLARE delay_val integer;
 SET delay_val=DATEDIFF(
   second, CURRENT TIME, '19:00');
 IF (delay_val>0 AND
     delay_val<3600)
 THEN
 UPDATE #hook_dict SET value=delay_val
   WHERE name='delay duration';
 END IF;
END
```

以下过程将于同步开始时在远程数据库中执行。它将检索当前的 MobiLink 用户名（sp_hook_dbmlsync_begin 事件的可用参数之一），并将其显示在 SQL Anywhere 消息窗口中。

```
CREATE PROCEDURE sp_hook_dbmlsync_begin()
BEGIN
   DECLARE syncdef VARCHAR(150);
   SELECT '>>>syncdef = ' || value INTO syncdef
      FROM #hook_dict
      WHERE name ='MobiLink user name';
   MESSAGE syncdef TYPE INFO TO CONSOLE;
END
```

事件挂接过程的连接

每个事件挂接过程都与同步本身在同一连接上执行。但以下过程例外：

- sp_hook_dbmlsync_all_error
- sp_hook_dbmlsync_communication_error
- sp_hook_dbmlsync_download_com_error
- sp_hook_dbmlsync_download_fatal_sql_error
- sp_hook_dbmlsync_download_log_ri_violation
- sp_hook_dbmlsync_misc_error
- sp_hook_dbmlsync_sql_error

这些过程在同步失败之前调用。失败时，同步操作将回退。通过在单独的连接上执行操作，您可以使用这些过程记录有关失败的信息，该记录操作不随同步操作一起回退。

处理事件挂接过程中的错误和警告

可以创建事件挂接存储过程来处理同步错误、MobiLink 连接故障和参照完整性违规。本节介绍用来处理错误和警告的事件挂接过程。一旦实施了这些过程，它们将在指定类型的错误发生时自动执行。

处理 RI 违规

当下载中的行违反了远程数据库的外键关系时，会发生参照完整性违规。使用以下的事件挂接来记录和处理参照完整性违规：

- “sp_hook_dbmlsync_download_log_ri_violation”一节第 251 页
- “sp_hook_dbmlsync_download_ri_violation”一节第 253 页

处理 MobiLink 连接故障

使用 sp_hook_dbmlsync_ml_connect_failed 事件挂接，您可以使用不同的通信类型或地址重试对 MobiLink 服务器失败的连接。如果连接最终失败，dbmlsync 将调用 sp_hook_dbmlsync_communication_error 和 sp_hook_dbmlsync_all_error 挂接。

请参见“sp_hook_dbmlsync_ml_connect_failed”一节第 271 页。

处理 dbmlsync 错误

每次生成 dbmlsync 错误消息时，都将调用以下挂接：

- 首先，根据错误的类型，将调用以下的挂接之一：sp_hook_dbmlsync_communication_error、sp_hook_dbmlsync_misc_error 或 sp_hook_dbmlsync_sql_error。这些挂接包含错误类型的特定信息；例如，针对 SQL 错误提供的 sqlcode 和 sqlstate。
- 其次，调用 sp_hook_dbmlsync_all_error。此挂接对于记录所有发生的错误非常有用。

请参见：

- “sp_hook_dbmlsync_communication_error”一节第 239 页
- “sp_hook_dbmlsync_sql_error”一节第 282 页
- “sp_hook_dbmlsync_misc_error”一节第 268 页
- “sp_hook_dbmlsync_all_error”一节第 234 页

如果您要重新启动响应某错误的一个同步，可以在 sp_hook_dbmlsync_end 中使用用户状态参数。

请参见“sp_hook_dbmlsync_end”一节第 261 页。

忽略错误

缺省情况下，当在事件挂接过程中遇到错误时同步将停止。通过提供 -eh 选项，您可以指示 dbmlsync 实用程序忽略事件挂接过程中出现的错误。

请参见“IgnoreHookErrors (eh) 扩展选项”一节第 191 页。

sp_hook_dbmlsync_abort

使用此存储过程可取消同步过程。

#hook_dict 表中各行

名称	值	说明
abort synchronization (in|out)	**true** | **false**	如果将 #hook_dict 表的 abort synchronization 行设置为 **true**，则 dbmlsync 将在该事件结束后立即终止。
publication_*n* (in)	发布	正被同步的发布，其中 *n* 是一个整数。每个正在上载的发布都有一个 publication_*n* 条目。*n* 的编号从零开始。
MobiLink user (in)	MobiLink 用户名	您正在为其进行同步的 MobiLink 用户。
exit code (in|out)	数字	如果将 abort synchronization 设置为 TRUE，可以使用此值设置已中止同步的退出代码。0 指示成功的同步。任何其它数字都表示同步失败。
script version (in|out)	脚本版本名称	将用于同步的 MobiLink 脚本版本。

注释

如果以该名称命名的过程存在，则它将在 dbmlsync 启动时被调用，并且将在每次 sp_hook_dbmlsync_delay 挂接导致同步延迟后被再次调用。

如果挂接通过将 abort synchronization 的值设置为 true 来请求中止，则此退出代码将传递给 sp_hook_dbmlsync_process_exit_code 挂接。如果未定义 sp_hook_dbmlsync_process_exit_code 挂接，则此退出代码会用作程序的退出代码。

此过程的操作将在执行后立即被提交。

另请参见

- “同步事件挂接序列”一节第 227 页
- “sp_hook_dbmlsync_process_exit_code”一节第 274 页

示例

以下过程将阻止在计划的维护时间（每天 19:00 到 20:00）内进行同步。

```
CREATE PROCEDURE sp_hook_dbmlsync_abort()
BEGIN
  DECLARE down_time_start TIME;
  DECLARE is_down_time VARCHAR(128);
  SET down_time_start='19:00';
  IF datediff( hour,down_time_start,now(*) ) < 1
  THEN
    set is_down_time='true';
```



```
  ELSE
    SET is_down_time='false';
  END IF;
  UPDATE #hook_dict
  SET value = is_down_time
  WHERE name = 'abort synchronization'
END;
```

假定您有一个中止挂接，它可能因为以下两个原因之一中止同步。原因之一是同步正常完成，因此您希望 dbmlsync 的退出代码为 0。另一个原因是出现错误，因此您希望 dbmlsync 具有非 0 的退出代码。可以使用定义如下的 sp_hook_dbmlsync_abort 挂接来实现。

```
BEGIN
   IF [condition that defines the normal abort case] THEN
      UPDATE #hook_dict SET value =  '0'
   WHERE name = 'exit code';
      UPDATE #hook_dict SET value =  'TRUE'
   WHERE name = 'abort synchronization';
   ELSEIF [condition that defines the error abort case] THEN
      UPDATE #hook_dict SET value =  '1'
   WHERE name = 'exit code';
      UPDATE #hook_dict SET value =  'TRUE'
   WHERE name = 'abort synchronization';
   END IF;
END;
```

sp_hook_dbmlsync_all_error

您可以使用此存储过程处理所有类型的 dbmlsync 错误消息。例如，您可以实现 sp_hook_dbmlsync_all_error 挂接，从而在发生特定错误时记录错误或执行特定操作。

#hook_dict 表中各行

名称	值	说明
publication_*n* (in)	发布	正被同步的发布，其中 *n* 是一个整数。每个正在上载的发布都有一个 publication_*n* 条目。*n* 的编号从零开始。
MobiLink user (in)	MobiLink 用户名	您正在为其进行同步的 MobiLink 用户。
script version (in)	脚本版本名	用于同步的 MobiLink 脚本版本。
error message (in)	错误消息文本	这与在 dbmlsync 日志中显示的文本相同。
error id (in)	整数	唯一标识消息的 ID。使用此行来标识错误消息，因为错误消息文本可能发生变化。
error hook user state (in|out)	整数	此值可由挂接设置，以便将状态信息传递到对 sp_hook_dbmlsync_all_error、sp_hook_dbmlsync_communication_error、sp_hook_dbmlsync_misc_error、sp_hook_dbmlsync_sql_error 或 sp_hook_dbmlsync_end 挂接的未来调用。第一次调用这些挂接的其中之一时，该行的值为 0。如果挂接更改了行值，则将在下一个挂接调用中使用新值。 使用此挂接将状态信息传递给 sp_hook_dbmlsync_end 挂接时，可以导致 sp_hook_dbmlsync_end 挂接执行某些操作（如重试同步）。

注释

每次生成 dbmlsync 错误消息时，都将调用以下挂接：

- 首先，根据错误的类型，将调用以下的挂接之一：sp_hook_dbmlsync_communication_error、sp_hook_dbmlsync_misc_error 或 sp_hook_dbmlsync_sql_error。这些挂接包含错误类型的特定信息；例如，针对 SQL 错误提供的 sqlcode 和 sqlstate。
- 其次，调用 sp_hook_dbmlsync_all_error。此挂接对于记录所有已发生的错误非常有用。

如果在启动过程中还没有初始化同步的时候发生了错误，用于 Mobilink 用户和脚本版本的 #hook_dict 条目将被设置为一个空字符串，并且在 #hook_dict 表中不设置 publication_*n* 行。

此过程将在单独的连接上执行以确保它执行的操作不会在同步连接执行回退时丢失。如果 dbmlsync 无法建立单独的连接，则不调用该过程。

缺省情况下，在 Windows Mobile 设备上，同步表以独占模式被锁定，这意味着如果此挂接需要访问任何同步表，它就不能成功执行。如果它需要访问同步表而您将 dbmlsync 扩展选项 LockTables 设置为 EXCLUSIVE，则它也无法执行。请参见“LockTables (lt) 扩展选项”一节第 194 页。

此过程的操作将在挂接完成后立即被提交。

另请参见

- “处理事件挂接过程中的错误和警告”一节第 231 页
- “sp_hook_dbmlsync_communication_error”一节第 239 页
- “sp_hook_dbmlsync_misc_error”一节第 268 页
- “sp_hook_dbmlsync_sql_error”一节第 282 页

示例

假定您使用下表记录远程数据库中的错误。

```
CREATE TABLE error_log
(
 pk INTEGER DEFAULT AUTOINCREMENT PRIMARY KEY,
 err_id INTEGER,
 err_msg VARCHAR(10240),
);
```

以下示例设置 sp_hook_dbmlsync_all_error 以记录错误。

```
CREATE PROCEDURE sp_hook_dbmlsync_all_error()
BEGIN
 DECLARE msg VARCHAR(10240);
 DECLARE id INTEGER;

 // get the error message text
 SELECT value INTO msg
  FROM #hook_dict
  WHERE name ='error message';

 // get the error id
 SELECT value INTO id
  FROM #hook_dict
  WHERE name = 'error id';

 // log the error information
 INSERT INTO error_log(err_msg, err_id)
  VALUES (msg, id);
END;
```

要查看可能存在的错误 ID 值，可测试运行 dbmlsync。例如，如果 dbmlsync 返回错误 [无法连接到 MobiLink 服务器]，sp_hook_dbmlsync_all_error 将在 error_log 中插入以下的行。

```
1,14173,
 'Unable to connect to MobiLink server'
```

现在，您就可以将错误 "无法连接到 MobiLink 服务器" 同错误 ID 14173 相关联。

以下的示例设置挂接，以便在每当发生错误 14173 时，都能够重试该同步。

```
CREATE PROCEDURE sp_hook_dbmlsync_all_error()
BEGIN
 IF EXISTS( SELECT value FROM #hook_dict
    WHERE name = 'error id' AND value = '14173' )
 THEN
    UPDATE #hook_dict SET value = '1'
       WHERE name = 'error hook user state';
   END IF;
END;

CREATE PROCEDURE sp_hook_dbmlsync_end()
BEGIN
 IF EXISTS( SELECT value FROM #hook_dict
    WHERE name='error hook user state' AND value='1')
 THEN
    UPDATE #hook_dict SET value = 'sync'
       WHERE name='restart';
   END IF;
END;
```

请参见“sp_hook_dbmlsync_end”一节第 261 页。

sp_hook_dbmlsync_begin

您可以使用此存储过程在同步过程开始时添加自定义操作。

#hook_dict 表中各行

名称	值	说明
publication_*n* (in)	发布	正被同步的发布，其中 *n* 是一个整数。每个正在上载的发布都有一个 publication_*n* 条目。*n* 的编号从零开始。
MobiLink user (in)	MobiLink 用户名	您正在为其进行同步的 MobiLink 用户。
script version (in)	脚本版本名	将用于同步的 MobiLink 脚本版本。

注释

如果以该名称命名的过程存在，则它将在同步过程开始时被调用。

该过程的操作将在执行后立即被提交。

另请参见

- “同步事件挂接序列”一节第 227 页

示例

假定您使用下面的表记录远程数据库上的同步事件。

```
CREATE TABLE SyncLog
(
 "event_id"           integer NOT NULL DEFAULT autoincrement ,
   "event_name"        varchar(128) NOT NULL ,
   "ml_user"             varchar(128) NULL ,
   "event_time"          timestamp NULL,
   "table_name"          varchar(128) NULL ,
   "upsert_count"        varchar(128) NULL ,
   "delete_count"        varchar(128) NULL ,
   "exit_code"           integer NULL ,
   "status_retval"       varchar(128) NULL ,
   "pubs"                 varchar(128) NULL ,
   "sync_descr "          varchar(128) NULL ,
    PRIMARY KEY ("event_id"),
)
```

以下示例编译一个发布列表。它在同步过程开始时记录发布列表和其它同步信息。

```
CREATE PROCEDURE sp_hook_dbmlsync_begin ()
BEGIN

    DECLARE pubs_list VARCHAR(1024);
    DECLARE temp_str VARCHAR(128);
    DECLARE qry VARCHAR(128);

-- insert publication list into pubs_list
    SELECT LIST(value) INTO pubs_list
```

```
        FROM #hook_dict
        WHERE name LIKE 'publication_%';

    -- log publication and synchronization information
        INSERT INTO SyncLog(event_name,ml_user,pubs,event_time)
        SELECT 'dbmlsync_begin',#hook_dict.value,pubs_list,CURRENT TIMESTAMP
        FROM #hook_dict
        WHERE name='MobiLink user';
    END
```

sp_hook_dbmlsync_communication_error

您可以使用此存储过程处理通信错误。

#hook_dict 表中各行

名称	值	说明
publication_*n* (in)	发布	正被同步的发布，其中 *n* 是一个整数。每个正在上载的发布都有一个 publication_*n* 条目。*n* 的编号从零开始。
MobiLink user (in)	MobiLink 用户名	您正在为其进行同步的 MobiLink 用户。
script version (in)	脚本版本名称	用于同步的 MobiLink 脚本版本。
error message (in)	错误消息文本	这与在 dbmlsync 日志中显示的文本相同。
error id (in)	数字	唯一标识消息的 ID。使用此行来标识错误消息，因为错误消息文本可能发生变化。
error hook user state (in|out)	整数	此值可由挂接设置，以便将状态信息传递到对 sp_hook_dbmlsync_all_error、sp_hook_dbmlsync_communication_error、sp_hook_dbmlsync_misc_error、sp_hook_dbmlsync_sql_error 或 sp_hook_dbmlsync_end 挂接的未来调用。第一次调用这些挂接的其中之一时，该行的值为 0。如果挂接更改了行值，则将在下一个挂接调用中使用新值。 使用此挂接将状态信息传递给 sp_hook_dbmlsync_end 挂接时，可以导致此 _end 挂接执行某些操作（如重试同步）。
stream error code (in)	整数	流报告的错误。
system error code (in)	整数	系统特定的错误代码。

注释

如果在启动过程中还没有初始化同步的时候发生了错误，用于 Mobilink 用户和脚本版本的 #hook_dict 条目将被设置为一个空字符串，并且在 #hook_dict 表中不设置 publication_*n* 行。

当在 dbmlsync 和 MobiLink 服务器间发生通信错误时，使用此挂接可以访问特定流的错误信息。

stream error code 参数是一个整数，用于指示通信错误的类型。

有关可能存在的错误代码值的列表，请参见“MobiLink 通信错误消息” 《错误消息》。

此过程将在单独的连接上执行以确保它执行的操作不会在同步连接执行回退时丢失。如果 dbmlsync 无法建立单独的连接，则不调用该过程。

缺省情况下，在 Windows Mobile 设备上，同步表以独占模式被锁定，这意味着如果此挂接需要访问任何同步表，它就不能成功执行。如果它需要访问同步表而您将 dbmlsync 扩展选项 LockTables 设置为 EXCLUSIVE，则它也无法执行。请参见“LockTables (lt) 扩展选项”一节第 194 页。

该过程的操作将在执行后立即被提交。

另请参见

- “处理事件挂接过程中的错误和警告”一节第 231 页
- “sp_hook_dbmlsync_all_error”一节第 234 页
- “sp_hook_dbmlsync_misc_error”一节第 268 页
- “sp_hook_dbmlsync_sql_error”一节第 282 页

示例

假定您使用下表记录远程数据库中的通信错误。

```
CREATE TABLE communication_error_log
(
  error_msg VARCHAR(10240),
  error_code VARCHAR(128)
);
```

以下示例设置 sp_hook_dbmlsync_communication_error 以记录通信错误。

```
CREATE PROCEDURE sp_hook_dbmlsync_communication_error()
BEGIN
 DECLARE msg VARCHAR(255);
 DECLARE code INTEGER;

 // get the error message text
 SELECT value INTO msg
  FROM #hook_dict
  WHERE name ='error message';

 // get the error code
 SELECT value INTO code
  FROM #hook_dict
  WHERE name = 'stream error code';

 // log the error information
 INSERT INTO communication_error_log(error_code,error_msg)
  VALUES (code,msg);
END
```


sp_hook_dbmlsync_delay

您可以使用此存储过程控制同步发生的时间。

#hook_dict 表中各行

名称	值	说明
delay duration (in|out)	秒数	如果该过程将 delay duration 的值设置为零，则 dbmlsync 同步将继续。非零的 delay duration 值指定再次调用延迟挂接前的秒数。
maximum accumulated delay (in|out)	秒数	最大积累延迟指定每次同步前的最大秒数。Dbmlsync 跟踪从上一次同步以来所有延迟挂接调用产生的总延迟。如果从 dbmlsync 开始运行以来没有同步发生，则总延迟从 dbmlsync 开始的时刻起计算。总延迟超过最大积累延迟的值时，将开始执行同步而不调用延迟挂接。
publication_*n* (in)	发布	正被同步的发布，其中 *n* 是一个整数。每个正在上载的发布都有一个 publication_*n* 条目。*n* 的编号从零开始。
MobiLink user (in)	MobiLink 用户名	您正在为其进行同步的 MobiLink 用户。
script version (in)	脚本版本名称	将用于同步的 MobiLink 脚本版本。

注释

如果存在以该名称命名的过程，则会在同步过程开始时在 **sp_hook_dbmlsync_begin** 之前调用该过程。

该过程的操作将在执行后立即被提交。

另请参见

- “同步事件挂接序列”一节第 227 页
- “使用事件挂接启动同步”一节第 115 页
- “sp_hook_dbmlsync_download_end”一节第 247 页

示例

假定您使用下面的表记录远程数据库中的优先次序。

```
CREATE TABLE OrdersTable(
  "id" INTEGER PRIMARY KEY DEFAULT AUTOINCREMENT,
  "priority" VARCHAR(128)
);
```

以下示例对同步进行最大积累延迟时间为 1 小时的延迟。每 10 秒钟再次调用挂接并查找 OrdersTable 中高优先级的行。如果存在高优先级的行，则将延迟持续时间设置为 0 以开始同步过程。

```
CREATE PROCEDURE sp_hook_dbmlsync_delay()
BEGIN
      -- Set the maximum delay between synchronizations
    -- or before the first synchronization starts to 1 hour
    UPDATE #hook_dict SET value = '3600'  // 3600 seconds
     WHERE name = 'maximum accumulated delay';

    -- check if a high priority order exists in OrdersTable
    IF EXISTS (SELECT * FROM OrdersTable where priority='high') THEN
     -- start the synchronization to process the high priority row
     UPDATE #hook_dict
      SET value = '0'
      WHERE name='delay duration';
    ELSE
     -- set the delay duration to call this procedure again
     -- following a 10 second delay
     UPDATE #hook_dict
      SET value = '10'
      WHERE name='delay duration';
    END IF;
END;
```

在 sp_hook_dbmlsync_end 挂接中，可以将高优先级的行标记为已处理：

```
CREATE PROCEDURE sp_hook_dbmlsync_upload_end()
     BEGIN
         IF EXISTS( SELECT value FROM #hook_dict
      WHERE name = 'Upload status'
      AND value = 'committed' ) THEN
      UPDATE OrderTable SET priority = 'high-processed'
     WHERE priority = 'high';
         END IF;
     END;
```

请参见“sp_hook_dbmlsync_end”一节第 261 页。

sp_hook_dbmlsync_download_begin

您可以使用此存储过程在同步过程的下载阶段开始时添加自定义操作。

#hook_dict 表中各行

名称	值	说明
publication_*n* (in)	发布	正被同步的发布，其中 *n* 是一个整数。每个正在上载的发布都有一个 publication_*n* 条目。*n* 的编号从零开始。
MobiLink user (in)	MobiLink 用户名	您正在为其进行同步的 MobiLink 用户。
script version (in)	脚本版本名称	将用于同步的 MobiLink 脚本版本。

注释

如果以该名称命名的过程存在，那么它将在同步过程的下载阶段开始时被调用。

此过程的操作将在下载被提交或被回退时被提交或被回退。

另请参见

- “同步事件挂接序列”一节第 227 页

示例

假定您使用下面的表记录远程数据库上的同步事件。

```
CREATE TABLE SyncLog
(
 "event_id"           INTEGER NOT NULL DEFAULT AUTOINCREMENT ,
   "event_name"        VARCHAR(128) NOT NULL ,
   "ml_user"             VARCHAR(128) NULL ,
   "event_time"          TIMESTAMP NULL,
   "table_name"          VARCHAR(128) NULL ,
   "upsert_count"        VARCHAR(128) NULL ,
   "delete_count"        VARCHAR(128) NULL ,
   "exit_code"           INTEGER NULL ,
   "status_retval"       VARCHAR(128) NULL ,
   "pubs"                 VARCHAR(128) NULL ,
   "sync_descr "          VARCHAR(128) NULL ,
    PRIMARY KEY ("event_id"),
);
```

以下示例编译一个发布列表。它在同步的下载阶段开始时记录发布列表和其它同步信息。

```
CREATE PROCEDURE sp_hook_dbmlsync_download_begin ()
BEGIN

    DECLARE pubs_list VARCHAR(1024);
    DECLARE temp_str VARCHAR(128);
    DECLARE qry VARCHAR(128);

-- insert publication list into pubs_list
    SELECT LIST(value) INTO pubs_list
```

```
        FROM #hook_dict
        WHERE name LIKE 'publication_%';

    -- log publication and synchronization information
       INSERT INTO SyncLog(event_name,ml_user,pubs,event_time)
        SELECT 'dbmlsync_download_begin',#hook_dict.value,
         pubs_list,CURRENT TIMESTAMP
        FROM #hook_dict
        WHERE name='MobiLink user';
    END;
```

sp_hook_dbmlsync_download_com_error（不建议使用）

您可以使用此存储过程添加自定义操作，以便在读取 MobiLink 服务器发送的下载过程中发生通信错误时执行。

不建议使用此挂接。请参见“处理事件挂接过程中的错误和警告”一节第 231 页。

#hook_dict 表中各行

名称	值	说明
table name (in)	表名	错误发生时正在对其应用操作的表。如果 dbmlsync 无法识别该表，则该值为空字符串。
publication_*n* (in)	发布	正被同步的发布，其中 *n* 是一个整数。每个正在上载的发布都有一个 publication_*n* 条目。*n* 的编号从零开始。
MobiLink user (in)	MobiLink 用户名	您正在为其进行同步的 MobiLink 用户。
script version (in)	脚本版本名称	将用于同步的 MobiLink 脚本版本。

注释

如果以该名称命名的过程存在，那么它将在同步过程的下载阶段检测到通信错误时被调用。然后终止下载。

此过程应在单独的连接上执行，以便记录故障。否则记录操作将与同步操作一同被回退。如果 dbmlsync 无法建立单独的连接，则不调用该过程。

缺省情况下，在 Windows Mobile 设备上，同步表以独占模式被锁定，这意味着如果此挂接需要访问任何同步表，它就不能成功执行。如果它需要访问同步表而您将 dbmlsync 扩展选项 LockTables 设置为 EXCLUSIVE，则它也无法执行。请参见“LockTables (lt) 扩展选项”一节第 194 页。

该过程的操作将在执行后立即被提交。

另请参见

- “同步事件挂接序列”一节第 227 页

示例

假定您使用下面的表记录通信错误。

```
CREATE TABLE SyncLogComErrorTable
(
   " user_name "  VARCHAR(255) NOT NULL ,
   " event_time "   TIMESTAMP NOT NULL ,
);
```

以下示例记录当读取由 MobiLink 服务器发送的下载过程中发生通信错误时的 MobiLink 用户和当前时间戳。信息存储在远程数据库的 SyncLogComErrorTable 表中。

```
CREATE PROCEDURE sp_hook_dbmlsync_download_com_error ()
BEGIN
 INSERT INTO SyncLogComErrorTable (user_name, event_time)
  SELECT #hook_dict.value, CURRENT TIMESTAMP
  FROM #hook_dict
  WHERE name = 'MobiLink user';
END;
```


sp_hook_dbmlsync_download_end

您可以使用此存储过程在同步过程的下载阶段结束时添加自定义操作。

#hook_dict 表中各行

名称	值	说明
publication_*n* (in)	发布	正被同步的发布，其中 *n* 是一个整数。每个正在上载的发布都有一个 publication_*n* 条目。*n* 的编号从零开始。
MobiLink user (in)	MobiLink 用户名	您正在为其进行同步的 MobiLink 用户。
script version (in)	脚本版本名称	将用于同步的 MobiLink 脚本版本。

注释

如果以该名称命名的过程存在，那么它将在同步过程的下载阶段结束时被调用。

此过程的操作将在下载被提交或被回退时被提交或被回退。

另请参见

- “同步事件挂接序列”一节第 227 页
- “使用事件挂接启动同步”一节第 115 页
- “sp_hook_dbmlsync_delay”一节第 241 页

示例

假定您使用下面的表记录远程数据库上的同步事件。

```
CREATE TABLE SyncLog
(
 "event_id"           INTEGER NOT NULL DEFAULT AUTOINCREMENT ,
   "event_name"        VARCHAR(128) NOT NULL ,
   "ml_user"             VARCHAR(128) NULL ,
   "event_time"          TIMESTAMP NULL,
   "table_name"          VARCHAR(128) NULL ,
   "upsert_count"        VARCHAR(128) NULL ,
   "delete_count"        VARCHAR(128) NULL ,
   "exit_code"           INTEGER NULL ,
   "status_retval"       VARCHAR(128) NULL ,
   "pubs"                 VARCHAR(128) NULL ,
   "sync_descr "          VARCHAR(128) NULL ,
    PRIMARY KEY ("event_id"),
)
```

以下示例编译一个发布列表。它在同步的下载阶段结束时记录发布列表和其它同步信息。

```
CREATE PROCEDURE sp_hook_dbmlsync_download_end ()
BEGIN

    DECLARE pubs_list VARCHAR(1024);
    DECLARE temp_str VARCHAR(128);
    DECLARE qry VARCHAR(128);
```

```
-- insert publication list into pubs_list
    SELECT LIST(value) INTO pubs_list
     FROM #hook_dict
     WHERE name LIKE 'publication_%';

-- log publication and synchronization information
    INSERT INTO SyncLog(event_name,ml_user,pubs,event_time)
     SELECT 'dbmlsync_download_end',#hook_dict.value,
      pubs_list,CURRENT TIMESTAMP
     FROM #hook_dict
     WHERE name='MobiLink user';
END
```

sp_hook_dbmlsync_download_fatal_sql_error（不建议使用）

在同步下载由于发生数据库错误即将被回退时执行操作。

不建议使用此挂接。请参见“处理事件挂接过程中的错误和警告”一节第 231 页。

#hook_dict 表中各行

名称	值	说明
table name (in)	表名	错误发生时正在对其应用操作的表。如果 dbmlsync 无法识别该表，则该值为空字符串。
SQL error code (in)	SQL 错误代码	在操作失败时标识由数据库返回的 SQL 错误代码。
publication_*n* (in)	发布	正被同步的发布，其中 *n* 是一个整数。每个正在上载的发布都有一个 publication_*n* 条目。*n* 的编号从零开始。
MobiLink user (in)	MobiLink 用户名	您正在为其进行同步的 MobiLink 用户。
script version (in)	脚本版本名称	将用于同步的 MobiLink 脚本版本。

注释

如果以该名称命名的过程存在，那么将在同步下载由于发生数据库错误而被回退之前立即调用它。每当遇到不可忽略的 SQL 错误或者当已经调用 sp_hook_dbmlsync_download_SQL_error 挂接并选择了不忽略错误时，将发生这种情况。

此过程应在单独的连接上执行，以便记录故障。否则记录操作将与同步操作一同被回退。如果 dbmlsync 无法建立单独的连接，则不调用该过程。

缺省情况下，在 Windows Mobile 设备上，同步表以独占模式被锁定，这意味着如果此挂接需要访问任何同步表，它就不能成功执行。如果它需要访问同步表而您将 dbmlsync 扩展选项 LockTables 设置为 EXCLUSIVE，则它也无法执行。请参见“LockTables (lt) 扩展选项”一节第 194 页。

该过程的操作将在执行后立即被提交。

另请参见

- “同步事件挂接序列”一节第 227 页
- “sp_hook_dbmlsync_download_sql_error（不建议使用）”一节第 255 页

示例

假定您使用下面的表记录 SQL 错误。

```
CREATE TABLE "DBA"."SyncLogComErrorTable"
(
   " error_code "       VARCHAR(255) NOT NULL ,
   " event_time "       TIMESTAMP NOT NULL ,
);
```

以下示例记录在读取下载的过程中发生 SQL 错误时的 SQL 错误代码和当前时间戳。信息存储在远程数据库的 SyncLogSQLErrorTable 中。

```
CREATE PROCEDURE sp_hook_dbmlsync_download_fatal_sql_error ()
BEGIN
 INSERT INTO SyncLogSQLErrorTable (error_code, event_time)
  SELECT #hook_dict.value, CURRENT TIMESTAMP
  FROM #hook_dict
  WHERE name = 'SQL error code';
END;
```


sp_hook_dbmlsync_download_log_ri_violation

在下载过程中记录参照完整性违规。

#hook_dict 表中各行

名称	值	说明
publication_*n* (in)	发布	正被同步的发布，其中 *n* 是一个整数。每个正在上载的发布都有一个 publication_*n* 条目。*n* 的编号从零开始。
MobiLink user (in)	MobiLink 用户名	您正在为其进行同步的 MobiLink 用户。
foreign key table (in)	表名	包含为其调用挂接的外键列的表。
primary key table (in)	表名	为其调用挂接的外键所引用的表。
role name (in)	角色名	为其调用挂接的外键的角色名。
script version (in)	脚本版本名称	将用于同步的 MobiLink 脚本版本。

注释

当下载中的行违反了远程数据库的外键关系时，会发生下载 RI 违规。使用此挂接可以在发生下载 RI 违规时记录 RI 违规，以便在以后调查违规原因。

下载完成后，但在提交之前，dbmlsync 将检查 RI 违规。如果 dbmlsync 发现任何 RI 违规，它将确定有 RI 违规的外键并调用 sp_hook_dbmlsync_download_log_ri_violation（如果实现了此存储过程）。然后调用 sp_hook_dbmlsync_download_ri_conflict（如果实现了此存储过程）。如果仍有冲突，dbmlsync 会删除违反外键约束的行。对剩下的有 RI 违规的外键重复此过程。

只有在 RI 违规涉及当前正在同步的表时，才调用此挂接。如果 RI 违规涉及的表当前没有同步，则不调用此挂接并且同步失败。

此挂接在 dbmlsync 用于下载的一个单独的连接上被调用。此挂接所使用的连接的隔离级别是 0，从而此挂接可以看到从尚未提交的下载中应用的行。此挂接的操作在其完成后立即提交，以便保留此挂接所做的更改，不论下载是被提交还是被回退。

缺省情况下，在 Windows Mobile 设备上，同步表以独占模式被锁定，这意味着如果此挂接需要访问任何同步表，它就不能成功执行。如果它需要访问同步表而您将 dbmlsync 扩展选项 LockTables 设置为 EXCLUSIVE，则它也无法执行。请参见“LockTables (lt) 扩展选项”一节第 194 页。

不要试图用此挂接修正 RI 违规问题。此挂接应该只用于记录。使用 sp_hook_dbmlsync_download_ri_violation 来解决 RI 违规。

另请参见

- “sp_hook_dbmlsync_download_ri_violation”一节第 253 页
- “同步事件挂接序列”一节第 227 页

示例

假定您使用下面的表记录参照完整性违规。

```
CREATE TABLE DBA.LogRIViolationTable
(
    entry_time  TIMESTAMP,
    pk_table  VARCHAR( 255 ),
    fk_table  VARCHAR( 255 ),
    role_name  VARCHAR( 255 )
);
```

以下示例记录当在远程数据库上检测到参照完整性违规时的外键表名、主键表名和角色名。信息存储在远程数据库的 LogRIErrorTable 中。

```
CREATE PROCEDURE sp_hook_dbmlsync_download_log_ri_violation()
BEGIN
 INSERT INTO DBA.LogRIViolationTable VALUES(
 CURRENT_TIMESTAMP,
 (SELECT value FROM #hook_dict WHERE name = 'Primary key table'),
 (SELECT value FROM #hook_dict WHERE name = 'Foreign key table'),
 (SELECT value FROM #hook_dict WHERE name = 'Role name' ) );
END;
```


sp_hook_dbmlsync_download_ri_violation

允许您在下载过程中解决参照完整性违规。

#hook_dict 表中各行

名称	值	说明
publication_*n* (in)	发布	正被同步的发布，其中 *n* 是一个整数。每个正在上载的发布都有一个 publication_*n* 条目。*n* 的编号从零开始。
MobiLink user (in)	MobiLink 用户名	您正在为其进行同步的 MobiLink 用户。
foreign key table (in)	表名	包含为其调用挂接的外键列的表。
primary key table (in)	表名	为其调用挂接的外键所引用的表。
role name (in)	角色名	为其调用挂接的外键的角色名。
script version (in)	脚本版本名称	将用于同步的 MobiLink 脚本版本。

注释

当下载中的行违反了远程数据库的外键关系时，会发生下载 RI 违规。使用此挂接可以尝试在 dbmlsync 删除导致冲突的行之前解决 RI 违规。

下载完成后，但在提交之前，dbmlsync 将检查 RI 违规。如果 dbmlsync 发现任何 RI 违规，它将确定有 RI 违规的外键并调用 sp_hook_dbmlsync_download_log_ri_violation（如果实现了此存储过程）。然后调用 sp_hook_dbmlsync_download_ri_conflict（如果实现了此存储过程）。如果仍然有冲突，dbmlsync 将删除行。对剩下的有 RI 违规的外键重复此过程。

只有在 RI 违规涉及当前正在同步的表时，才调用此挂接。如果 RI 违规涉及的表当前没有同步，则不调用此挂接并且同步失败。

此挂接在 dbmlsync 用于下载的同一连接上被调用。此挂接不应该包含任何隐式或显式提交，因为这些提交可能导致数据库中的数据不一致。此挂接的操作将在下载被提交或被回退时被提交或被回退。

与其它挂接操作不同，在此挂接期间执行的操作不在下一个同步期间上载。

另请参见

- “sp_hook_dbmlsync_download_log_ri_violation”一节第 251 页

示例

此示例使用如下所示的 Department 和 Employee 表：

```
CREATE TABLE Department(
 "department_id"  INTEGER primary key
);
```

```
CREATE TABLE Employee(
  "employee_id"      INTEGER PRIMARY KEY,
  "department_id" INTEGER,
  FOREIGN KEY EMPLOYEE_FK1 (department_id) REFERENCES Department
);
```

以下的 sp_hook_dbmlsync_download_ri_violation 定义清除 Department 和 Employee 表之间的参照完整性违规。它校验外键的角色名，并将缺少的 department_id 值插入 Department 表。

```
CREATE PROCEDURE sp_hook_dbmlsync_download_ri_violation()
BEGIN

IF EXISTS (SELECT * FROM #hook_dict WHERE name = 'role name'
 AND value = 'EMPLOYEE_FK1') THEN

 -- update the Department table with missing department_id values
 INSERT INTO Department
  SELECT distinct department_id FROM Employee
  WHERE department_id NOT IN (SELECT department_id FROM Department)

END IF;
```


sp_hook_dbmlsync_download_sql_error（不建议使用）

处理在应用由 MobiLink 服务器发送的下载时发生的数据库错误。

不建议使用此挂接。请参见“处理事件挂接过程中的错误和警告”一节第 231 页。

#hook_dict 表中各行

名称	值	说明
table name (in)	表名	错误发生时正在对其应用操作的表。如果 dbmlsync 无法识别该表，则该值为空字符串。
continue (in|out)	**true** | **false**	指示是否应该忽略该错误并继续同步。该参数应设置为 **false**，以调用 sp_hook_dbmlsync_download_fatal_sql_error 挂接并停止同步。如果将此参数设置为 **true**，dbmlsync 会忽略错误并继续执行同步，这可能导致数据丢失。
SQL error code (in)	SQL 错误代码	在操作失败时标识由数据库返回的 SQL 错误代码。
publication_*n* (in)	发布	正被同步的发布，其中 *n* 是一个整数。每个正在上载的发布都有一个 publication_*n* 条目。*n* 的编号从零开始。
MobiLink user (in)	MobiLink 用户名	您正在为其进行同步的 MobiLink 用户。
script version (in)	脚本版本名称	将用于同步的 MobiLink 脚本版本。

注释

如果以该名称命名的过程存在，那么它将在同步的下载阶段检测到数据库错误时被调用。该过程仅在有可能忽略错误并继续同步的情况下被调用。如果发生致命错误，则需要调用 sp_hook_dbmlsync_download_fatal_SQL_error 过程。

小心

如果将 continue 设置为 TRUE，dbmlsync 会忽略数据库错误并继续执行同步。而不会重新尝试失败的操作。这样一来，可能会丢失某些或全部下载。丢失的数据量取决于遇到的错误类型及发生错误时挂接采取的恢复步骤。很难预测哪些数据会丢失，因此使用此功能时必须非常谨慎。建议大多数用户在遇到 SQL 错误时最好不要尝试继续。

此过程的操作将在下载被提交或被回退时被提交或被回退。

另请参见

- “同步事件挂接序列”一节第 227 页
- “sp_hook_dbmlsync_download_fatal_sql_error（不建议使用）”一节第 249 页

sp_hook_dbmlsync_download_table_begin

您可以使用此存储过程在每个表下载前添加自定义操作。

#hook_dict 表中各行

名称	值	说明
table name (in)	表名	即将对其应用操作的表。
publication_*n* (in)	发布	正被同步的发布，其中 *n* 是一个整数。每个正在上载的发布都有一个 publication_*n* 条目。*n* 的编号从零开始。
MobiLink user (in)	MobiLink 用户名	您正在为其进行同步的 MobiLink 用户。
script version (in)	脚本版本名称	将用于同步的 MobiLink 脚本版本。

注释

如果以该名称命名的过程存在，则将在下载操作应用到每个表之前立即为该表调用此过程。此过程的操作将在下载被提交或被回退时被提交或被回退。

另请参见

- “同步事件挂接序列”一节第 227 页

示例

假定您使用下面的表记录远程数据库上的同步事件。

```
CREATE TABLE SyncLog
(
 "event_id"           INTEGER NOT NULL DEFAULT AUTOINCREMENT ,
   "event_name"        VARCHAR(128) NOT NULL ,
   "ml_user"             VARCHAR(128) NULL ,
   "event_time"          TIMESTAMP NULL,
   "table_name"          VARCHAR(128) NULL ,
   "upsert_count"        VARCHAR(128) NULL ,
   "delete_count"        VARCHAR(128) NULL ,
   "exit_code"           INTEGER NULL ,
   "status_retval"       VARCHAR(128) NULL ,
   "pubs"                 VARCHAR(128) NULL ,
   "sync_descr "          VARCHAR(128) NULL ,
    PRIMARY KEY ("event_id"),
);
```

以下示例记录即将下载表时的 MobiLink 用户、表名和当前时间戳。

```
CREATE PROCEDURE sp_hook_dbmlsync_download_table_begin()
BEGIN
    DECLARE tbl VARCHAR(255);

    -- load the table name from #hook_dict
  SELECT #hook_dict.value
   INTO tbl
```

```
    FROM #hook_dict
    WHERE #hook_dict.name = 'table name';

  INSERT INTO SyncLog (event_name, ml_user, table_name
    ,event_time)
    SELECT 'download_table_begin', #hook_dict.value, tbl
    ,CURRENT TIMESTAMP
    FROM #hook_dict
    WHERE name = 'MobiLink user' ;
END;
```


sp_hook_dbmlsync_download_table_end

您可以使用此存储过程在每个表下载后添加自定义操作。

#hook_dict 表中各行

名称	值	说明
table name (in)	表名	刚刚对其应用过操作的表。
delete count (in)	行数	此表中由下载删除的行数。
upsert count (in)	行数	此表中由下载更新或插入的行数。
publication_*n* (in)	发布	正被同步的发布，其中 *n* 是一个整数。每个正在上载的发布都有一个 publication_*n* 条目。*n* 的编号从零开始。
MobiLink user (in)	MobiLink 用户名	您正在为其进行同步的 MobiLink 用户。
script version (in)	脚本版本名称	将用于同步的 MobiLink 脚本版本。

注释

如果以该名称命名的过程存在，则将在对表应用完下载中的所有操作之后立即调用此过程。

此过程的操作将在下载被提交或被回退时被提交或被回退。

另请参见

- “同步事件挂接序列”一节第 227 页

示例

假定您使用下面的表记录远程数据库上的同步事件。

```
CREATE TABLE SyncLog
(
 "event_id"           INTEGER NOT NULL DEFAULT AUTOINCREMENT ,
   "event_name"        VARCHAR(128) NOT NULL ,
   "ml_user"             VARCHAR(128) NULL ,
   "event_time"          TIMESTAMP NULL,
   "table_name"          VARCHAR(128) NULL ,
   "upsert_count"        VARCHAR(128) NULL ,
   "delete_count"        VARCHAR(128) NULL ,
   "exit_code"           INTEGER NULL ,
   "status_retval"       VARCHAR(128) NULL ,
   "pubs"                 VARCHAR(128) NULL ,
   "sync_descr "          VARCHAR(128) NULL ,
    PRIMARY KEY ("event_id"),
);
```

以下示例在下载表之后立即记录 MobiLink 用户、表名和插入或更新的行数。

```
CREATE PROCEDURE sp_hook_dbmlsync_download_table_end()
BEGIN
```

```
        -- declare variables
        DECLARE tbl VARCHAR(255);
        DECLARE upsertCnt VARCHAR(255);
        DECLARE deleteCnt VARCHAR(255);

        -- load the table name from #hook_dict
        SELECT #hook_dict.value
         INTO tbl
         FROM #hook_dict
         WHERE #hook_dict.name = 'table name';

        -- load the upsert count from #hook_dict
        SELECT #hook_dict.value
         INTO upsertCnt
         FROM #hook_dict
         WHERE #hook_dict.name = 'upsert count';

        -- load the delete count from #hook_dict
        SELECT #hook_dict.value
         INTO deleteCnt
         FROM #hook_dict
         WHERE #hook_dict.name = 'delete count';

        INSERT INTO SyncLog (event_name, ml_user, table_name,
          upsert_count, delete_count, event_time)
         SELECT 'download_table_end', #hook_dict.value, tbl,
          upsertCnt, deleteCnt, CURRENT TIMESTAMP
         FROM #hook_dict
         WHERE name = 'MobiLink user' ;
    END;
```


sp_hook_dbmlsync_end

您可以使用此存储过程在同步完成之前添加自定义操作。

#hook_dict 表中各行

名称	值	说明
restart (out)	**sync** | **download** | **none**	如果设置为 **sync**，dbmlsync 会重试它刚完成的同步。值 **sync** 替换 **true**，虽然后者与前者作用相同，但不建议使用后者。 如果设置为 **none**（缺省值），则 dbmlsync 将根据其命令行参数关闭或重新启动。值 **none** 替换 **false**，虽然后者与前者作用相同，但不建议使用后者。 如果设置为 **download** 并且 restartable download 参数为 **true**，则 dbmlsync 会尝试重新启动刚才失败的下载。
exit code (in)	数字	如果设置为任意非零值（零为缺省值），则表示同步错误。
publication_*n* (in)	发布	正被同步的发布，其中 *n* 是一个整数。每个正在上载的发布都有一个 publication_*n* 条目。*n* 的编号从零开始。
MobiLink user (in)	MobiLink 用户名	您正在为其进行同步的 MobiLink 用户。
upload status (in)	**not sent** | **committed** | **failed**	指定在 dbmlsync 试图验证是否接收到上载时 MobiLink 服务器所返回的状态。状态可以是： ● **not sent** - 未向 MobiLink 服务器发送上载，因为某个错误阻止了上载或是因为请求的同步不需要上载，不发送上载会在诸如仅下载同步、重新启动的下载或基于文件的下载这些情况下发生。 ● **committed** - MobiLink 服务器已接收到上载并已提交。 ● **failed** - MobiLink 服务器未提交上载。对于事务性上载，当一些事务但不是所有事务由服务器成功上载并确认后，上载状态为 '失败'。
script version (in)	脚本版本名称	将用于同步的 MobiLink 脚本版本。

名称	值	说明
restartable download (in)	**true**|**false**	如果为 **true**，则当前同步的下载已失败并且可以重新启动。如果为 **false**，则下载已成功或无法重新启动。
restartable download size (in)	整数	当 restartable download 参数为 **true** 时，此参数指示下载失败前收到的字节数。当 restartable download 为 **false** 时，此值无意义。
error hook user state (in)	整数	该值包含关于错误的信息，可从挂接 sp_hook_dbmlsync_all_error、sp_hook_dbmlsync_communication_error、sp_hook_dbmlsync_misc_error 或 sp_hook_dbmlsync_sql_error 发送。

注释

如果以该名称命名的过程存在，那么它将在每个同步结束时被调用。

该过程的操作将在执行后立即被提交。

如果定义 sp_hook_dbmlsync_end 挂接，使其始终将 restart 参数设置为 **sync**，并且在 dbmlsync 命令行上以 -n pub1、-n pub2、……，这种格式指定了多个发布，则 dbmlsync 会重复同步第一个发布而永远不同步第二个发布。

另请参见

- “dbmlsync 挂接简介”一节第 227 页
- “同步事件挂接序列”一节第 227 页
- “恢复失败的下载”一节 《MobiLink - 服务器管理》
- “处理事件挂接过程中的错误和警告”一节第 231 页

示例

在下面的示例中，如果当前同步的下载失败并可以重新启动，则手工重新启动下载。

```
CREATE PROCEDURE sp_hook_dbmlsync_end()
BEGIN
  -- Restart the download if the download for the current sync
  --  failed and can be restarted
  IF EXISTS (SELECT * FROM #hook_dict
    WHERE name = 'restartable download' AND value='true')
      THEN
   UPDATE #hook_dict SET value ='download' WHERE name='restart';
  END IF;
END;
```


sp_hook_dbmlsync_log_rescan

您可以使用此存储过程以编程方式来确定何时需要重新扫描。

#hook_dict 表中各行

名称	值	说明
publication_*n* (in)	发布	正被同步的发布，其中 *n* 是一个整数。每个正在上载的发布都有一个 publication_*n* 条目。*n* 的编号从零开始。
MobiLink user (in)	MobiLink 用户名	您正在为其进行同步的 MobiLink 用户。
discarded storage (in)	数字	最后一次同步后放弃的内存的字节数。
rescan (in|out)	**true** | **false**	如果被挂接设置为 True，dbmlsync 会在下次同步前执行完整的重新扫描。开始时，此值被设置为 False。
script version (in)	脚本版本名称	将用于同步的 MobiLink 脚本版本。

注释

如果在命令行中指定了多个 -n 选项，dbmlsync 可能会遇到碎片，而碎片会形成被放弃的内存。可以通过重新扫描数据库事务日志来恢复放弃的内存。使用此挂接可以决定 dbmlsync 是否应重新扫描数据库事务日志以恢复内存。

如果未遇到将强制重新扫描的其它情况，则在 sp_hook_dbmlsync_process_exit_code 挂接后立即调用此挂接。

另请参见

- “HoverRescanThreshold (hrt) 扩展选项”一节第 190 页

示例

下面的示例在放弃的存储大于 1000 字节时将 #hook_dict 表中的 rescan 字段设置为 TRUE。

```
CREATE PROCEDURE sp_hook_dbmlsync_log_rescan ()
BEGIN
 IF EXISTS(SELECT * FROM #hook_dict
  WHERE name = 'Discarded storage' AND value>1000)
 THEN
  UPDATE #hook_dict SET value ='true' WHERE name='Rescan';
 END IF;
END;
```

sp_hook_dbmlsync_logscan_begin

您可以使用此存储过程在为上载而扫描事务日志之前添加自定义操作。

#hook_dict 表中各行

名称	值	说明
starting log offset_*n* (in)	数字	扫描即将开始处的日志偏移值。每个正在上载的发布都有一个相应值。*n* 的编号从零开始。此值与 Publication-*n* 相匹配。例如，log offset_0 是 publication_0 的偏移。
log scan retry (in)	**true** | **false**	如果这是为此同步进行的第一次事务日志扫描，则该值为 false，否则为 true。如果 MobiLink 服务器与 dbmlsync 中关于从何处开始扫描的信息不同，则进行两次日志扫描。
publication_*n* (in)	发布	正被同步的发布，其中 *n* 是一个整数。每个正在上载的发布都有一个 publication_*n* 条目。*n* 的编号从零开始。
MobiLink user (in)	MobiLink 用户名	您正在为其进行同步的 MobiLink 用户。
script version (in)	脚本版本名称	将用于同步的 MobiLink 脚本版本。

注释

如果以该名称命名的过程存在，那么它将在 dbmlsync 即将扫描事务日志以汇编上载时被调用。

该过程的操作将在执行后立即被提交。

另请参见

- “同步事件挂接序列”一节第 227 页

示例

假定您使用下面的表记录远程数据库上的同步事件。

```
CREATE TABLE SyncLog
(
 "event_id"           INTEGER NOT NULL DEFAULT AUTOINCREMENT ,
   "event_name"        VARCHAR(128) NOT NULL ,
   "ml_user"             VARCHAR(128) NULL ,
   "event_time"          TIMESTAMP NULL,
   "table_name"          VARCHAR(128) NULL ,
   "upsert_count"        VARCHAR(128) NULL ,
   "delete_count"        VARCHAR(128) NULL ,
   "exit_code"           INTEGER NULL ,
   "status_retval"       VARCHAR(128) NULL ,
   "pubs"                 VARCHAR(128) NULL ,
   "sync_descr "          VARCHAR(128) NULL ,
```



```
    PRIMARY KEY ("event_id"),
);
```

以下示例记录即将为上载扫描事务日志时的 MobiLink 用户和当前时间戳。

```
CREATE PROCEDURE sp_hook_dbmlsync_logscan_begin ()
BEGIN
 -- log the synchronization event
 INSERT INTO SyncLog (event_name, ml_user,event_time)
  SELECT 'logscan_begin', #hook_dict.value, CURRENT TIMESTAMP
  FROM #hook_dict
  WHERE name = 'MobiLink user' ;
END;
```

sp_hook_dbmlsync_logscan_end

您可以使用此存储过程在为上载而进行事务日志扫描之后添加自定义操作。

#hook_dict 表中各行

名称	值	说明
ending log offset (in)	数字	扫描结束处的日志偏移值。
starting log offset_*n* (in)	数字	您同步的每个预订的初始进度值。此 *n* 值对应于 Publication_*n* 中的值。例如，Starting log offset_1 是 Publication_1 的偏移。
log scan retry (in)	**true** | **false**	如果这是为此同步进行的第一次事务日志扫描，则该值为 false，否则为 true。如果 MobiLink 服务器与 dbmlsync 中关于从何处开始扫描的信息不同，则进行两次日志扫描。
publication_*n* (in)	发布	正被同步的发布，其中 *n* 是一个整数。每个正在上载的发布都有一个 publication_*n* 条目。*n* 的编号从零开始。
MobiLink user (in)	MobiLink 用户名	您正在为其进行同步的 MobiLink 用户。
script version (in)	脚本版本名称	将用于同步的 MobiLink 脚本版本。

注释

如果以该名称命名的过程存在，那么它将在 dbmlsync 扫描过事务日志以汇编上载之后立即被调用。

该过程的操作将在执行后立即被提交。

另请参见

- “同步事件挂接序列”一节第 227 页

示例

假定您使用下面的表记录远程数据库上的同步事件。

```
CREATE TABLE SyncLog
(
 "event_id"           INTEGER NOT NULL DEFAULT AUTOINCREMENT ,
   "event_name"        VARCHAR(128) NOT NULL ,
   "ml_user"             VARCHAR(128) NULL ,
   "event_time"          TIMESTAMP NULL,
   "table_name"          VARCHAR(128) NULL ,
   "upsert_count"        VARCHAR(128) NULL ,
   "delete_count"        VARCHAR(128) NULL ,
   "exit_code"           INTEGER NULL ,
   "status_retval"       VARCHAR(128) NULL ,
```



```
    "pubs"                VARCHAR(128) NULL ,
    "sync_descr "         VARCHAR(128) NULL ,
     PRIMARY KEY ("event_id"),
);
```

以下示例在为上载扫描过事务日志后立即记录 MobiLink 用户和当前时间戳。#hook_dict 日志扫描重试参数指示是否扫描了多次事务日志。

```
CREATE PROCEDURE sp_hook_dbmlsync_logscan_end ()
BEGIN

 DECLARE scan_retry VARCHAR(128);

 -- load the scan retry parameter from #hook_dict
 SELECT #hook_dict.value
  INTO scan_retry
  FROM #hook_dict
  WHERE #hook_dict.name = 'log scan retry';

 -- Determine if the log is being rescanned
 -- and log the synchronization event

 IF scan_retry='true' THEN
   INSERT INTO SyncLog (event_name, ml_user,event_time,sync_descr)
    SELECT 'logscan_end', #hook_dict.value, CURRENT TIMESTAMP,
     'Transaction log rescanned'
    FROM #hook_dict
    WHERE name = 'MobiLink user' ;
 ELSE
   INSERT INTO SyncLog (event_name, ml_user,event_time,sync_descr)
    SELECT 'logscan_end', #hook_dict.value, CURRENT TIMESTAMP,
     'Transaction log scanned normally'
    FROM #hook_dict
    WHERE name = 'MobiLink user' ;
 END IF;
END;
```

sp_hook_dbmlsync_misc_error

您可以使用此存储过程处理未被分类为数据库错误或通信错误的 dbmlsync 错误。例如，您可以实现 sp_hook_dbmlsync_misc_error 挂接，从而在发生特定错误时记录错误或执行特定操作。

#hook_dict 表中各行

名称	值	说明
publication_*n* (in)	发布	正被同步的发布，其中 *n* 是一个整数。每个正在上载的发布都有一个 publication_*n* 条目。*n* 的编号从零开始。
MobiLink user (in)	MobiLink 用户名	您正在为其进行同步的 MobiLink 用户。
script version (in)	脚本版本名称	将用于同步的 MobiLink 脚本版本。
error message (in)	错误消息文本	这与在 dbmlsync 日志中显示的文本相同。
error id (in)	整数	唯一标识消息的 ID。使用此行来标识错误消息，因为错误消息文本可能发生变化。
error hook user state (in|out)	整数	此值可由挂接设置，以便将状态信息传递到对 sp_hook_dbmlsync_all_error、sp_hook_dbmlsync_communication_error、sp_hook_dbmlsync_misc_error、sp_hook_dbmlsync_sql_error 或 sp_hook_dbmlsync_end 挂接的未来调用。第一次调用这些挂接的其中之一时，该行的值为 0。如果挂接更改了行值，则将在下一个挂接调用中使用新值。 使用此挂接将状态信息传递给 sp_hook_dbmlsync_end 挂接时，可以导致此 _end 挂接执行某些操作（如重试同步）。

注释

如果在启动过程中还没有初始化同步的时候发生了错误，用于 Mobilink 用户和脚本版本的 #hook_dict 条目将被设置为一个空字符串，并且在 #hook_dict 表中不设置 publication_*n* 行。

此过程将在单独的连接上执行以确保它执行的操作不会在同步连接执行回退时丢失。如果 dbmlsync 无法建立单独的连接，则不调用该过程。

缺省情况下，在 Windows Mobile 设备上，同步表以独占模式被锁定，这意味着如果此挂接需要访问任何同步表，它就不能成功执行。如果它需要访问同步表而您将 dbmlsync 扩展选项 LockTables 设置为 EXCLUSIVE，则它也无法执行。请参见“LockTables (lt) 扩展选项”一节第 194 页。

该过程的操作将在执行后立即被提交。

另请参见

- “处理事件挂接过程中的错误和警告”一节第 231 页
- “sp_hook_dbmlsync_communication_error”一节第 239 页
- “sp_hook_dbmlsync_all_error”一节第 234 页
- “sp_hook_dbmlsync_sql_error”一节第 282 页

示例

假定您使用下表记录远程数据库中的错误。

```
CREATE TABLE error_log
(
 pk INTEGER DEFAULT AUTOINCREMENT PRIMARY KEY,
 err_id INTEGER,
 err_msg VARCHAR(10240),
);
```

以下示例设置 sp_hook_dbmlsync_misc_error 以记录所有错误消息的类型。

```
CREATE PROCEDURE sp_hook_dbmlsync_misc_error()
BEGIN
 DECLARE msg VARCHAR(10240);
 DECLARE id INTEGER;

 // get the error message text
 SELECT value INTO msg
  FROM #hook_dict
  WHERE name ='error message';

 // get the error id
 SELECT value INTO id
  FROM #hook_dict
  WHERE name = 'error id';

 // log the error information
 INSERT INTO error_log(err_msg,err_id)
  VALUES (msg,id);
END;
```

要查看可能存在的错误 ID 值，可测试运行 dbmlsync。例如，以下的 dbmlsync 命令行引用了一个无效的发布。

```
dbmlsync -c eng=custdb;uid=DBA;pwd=sql -n test
```

现在 error_log 表包含以下行，从而将错误与错误 ID 9931 相关联。

```
1,9931,
 'There is no synchronization subscription to publication "test"'
```

若要进行自定义错误处理，请检查 sp_hook_dbmlsync_misc_error 中的错误 ID 9931。

```
ALTER PROCEDURE sp_hook_dbmlsync_misc_error()
BEGIN
 DECLARE msg VARCHAR(10240);
 DECLARE id INTEGER;

 // get the error message text
 SELECT value INTO msg
  FROM #hook_dict
  WHERE name ='error message';
```

```
  // get the error id
  SELECT value INTO id
   FROM #hook_dict
   WHERE name = 'error id';

  // log the error information
  INSERT INTO error_log(err_msg,err_id)
   VALUES (msg,id);

  IF id = 9931 THEN
   // handle invalid publication
  END IF;

END;
```


sp_hook_dbmlsync_ml_connect_failed

使用此存储过程可以使用不同的通信类型或地址重试对 MobiLink 服务器失败的连接。

#hook_dict 表中各行

名称	值	说明
publication_*n* (in)	发布	正被同步的发布，其中 *n* 是一个整数。每个正在上载的发布都有一个 publication_*n* 条目。*n* 的编号从零开始。
MobiLink user (in)	MobiLink 用户名	您正在为其进行同步的 MobiLink 用户。
script version (in)	脚本版本名称	将用于同步的 MobiLink 脚本版本。
connection address (in|out)	连接地址	调用挂接时，这是在最近失败的通信尝试中使用的地址。可以将该值设置为要尝试的新的连接地址。如果将重试设置为 true，则该值将用于下一次通信尝试。有关协议选项的列表，请参见“MobiLink 客户端网络协议选项汇总”一节第 33 页。
connection type (in|out)	网络协议	调用挂接时，这是在最近失败的通信尝试中使用的网络协议（如 TCPIP）。可以将此值设置为要尝试的新的网络协议。如果将重试设置为 true，则该值将用于下一次通信尝试。有关网络协议的列表，请参见“CommunicationType (ctp) 扩展选项”一节第 180 页。
user data (in|out)	用户定义的数据	下次连接尝试失败时将使用的状态信息。例如，您可能会发现存储已重试次数很有用。缺省值为空字符串。
allow remote ahead (in|out)	**true** | **false**	仅当使用 -ra 选项启动了 dbmlsync 时为 true。该行只能用于读取或更改当前同步的 -ra 选项。请参见“-r 选项”一节第 162 页。
allow remote behind (in|out)	**true** | **false**	仅当使用 -rb 选项启动了 dbmlsync 时为 true。该行只能用于读取或更改当前同步的 -rb 选项。请参见“-r 选项”一节第 162 页。
retry (in|out)	**true** | **false**	如果要重试失败的连接尝试，请将该值设置为 true。缺省值为 FALSE。

注释

如果以该名称命名的过程存在，那么它将在 dbmlsync 未能连接到 MobiLink 服务器时被调用。

此挂接只适用于对 MobiLink 服务器（而非数据库）的连接尝试。

当出现进度偏移不匹配时，dbmlsync 会断开与 MobiLink 服务器的连接，稍后重新连接。在这种重新连接中，不调用此挂接，重新连接失败导致同步失败。

该过程的操作将在执行后立即被提交。

示例

此示例最多可五次使用 sp_hook_dbmlsync_ml_connect_failed 挂接重试连接。

```
CREATE PROCEDURE sp_hook_dbmlsync_ml_connect_failed ()
BEGIN
    DECLARE idx integer;

    SELECT value
      INTO buf
      FROM #hook_dict
      WHERE name = 'user data';

  IF idx <= 5 THEN
        UPDATE #hook_dict
        SET value = idx
        WHERE name = 'user data';

        UPDATE #hook_dict
        SET value = 'TRUE'
        WHERE name = 'retry';
  END IF;
END;
```

下一示例使用一个包含连接信息的表。当尝试连接失败时，挂接会尝试列表中的下一个服务器。

```
CREATE TABLE conn_list (
    label   INTEGER PRIMARY KEY,
    addr  VARCHAR( 128 ),
    type  VARCHAR( 64 )
);
INSERT INTO conn_list
 VALUES ( 1, 'host=server1;port=91', 'tcpip' );
INSERT INTO conn_list
 VALUES ( 2, 'host=server2;port=92', 'http' );
INSERT INTO conn_list
 VALUES ( 3, 'host=server3;port=93', 'tcpip' );
COMMIT;

CREATE PROCEDURE sp_hook_dbmlsync_ml_connect_failed ()
BEGIN
 DECLARE idx INTEGER;
 DECLARE cnt INTEGER;

 SELECT value
  INTO idx
  FROM #hook_dict
  WHERE name = 'user data';

 SELECT COUNT( label ) INTO cnt FROM conn_list;

 IF idx <= cnt THEN
```



```
    UPDATE #hook_dict
        SET value = ( SELECT addr FROM conn_list WHERE label = idx )
        WHERE name = 'connection address';
             UPDATE #hook_dict
     SET value = (SELECT type FROM conn_list WHERE label=idx)
     WHERE name = 'connection type';

    UPDATE #hook_dict
     SET value = idx
         WHERE name = 'user data';

        UPDATE #hook_dict
         SET value = 'TRUE'
         WHERE name = 'retry';
 END IF;
END;
```

sp_hook_dbmlsync_process_exit_code

您可以使用此存储过程管理退出代码。

#hook_dict 表中各行

名称	值	说明
publication_*n* (in)	发布	正被同步的发布，其中 *n* 是一个整数。每个正在上载的发布都有一个 publication_*n* 条目。*n* 的编号从零开始。
MobiLink user (in)	MobiLink 用户名	您正在为其进行同步的 MobiLink 用户。
fatal error (in)	**true** | **false**	因为将导致 dbmlsync 终止的错误而调用此挂接时为 True。
aborted synchronization (in)	**true** | **false**	因为 sp_hook_dbmlsync_abort 挂接的中止请求而调用此挂接时为 True。
exit code (in)	数字	最近同步尝试的退出代码。0 指示成功的同步。任何其它值都表示同步失败。该值可在 sp_hook_dbmlsync_abort 用于中止同步时由该挂接设置。
last exit code (in)	数字	上次调用此挂接时存储在 #hook_dict 表 **new exit code** 行中的值，或为 0（如果这是第一次调用此挂接）。
new exit code (in|out)	数字	您为进程选择的退出代码。当 dbmlsync 退出时，其 **exit code** 为上次调用该挂接时存储在该行中的值。该值必须在 -32768 至 32767 之间。
script version (in)	脚本版本名称	将用于同步的 MobiLink 脚本版本。

注释

当命令行中指定多次 -n 选项，或使用调度，或使用 sp_hook_dbmlsync_end 中的 restart 参数时，一个 dbmlsync 会话可以运行多个同步。在这些情况下，如果一个或多个同步失败，缺省的退出代码不能表示哪个同步失败了。您可以使用此挂接根据同步的退出代码来定义 dbmlsync 过程的退出代码。此挂接还可用于记录退出代码。

如果在启动过程中还没有初始化同步的时候发生了错误，用于 Mobilink 用户和脚本版本的 #hook_dict 条目将被设置为一个空字符串，并且在 #hook_dict 表中不设置 publication_*n* 行。

示例

假定您运行 dbmlsync 来执行五个同步，并且希望退出代码指示失败的同步的个数，其中退出代码 0 指示无失败，退出代码 1 指示有一个同步失败，依次类推。您可以通过定义 sp_hook_dbmlsync_process_exit_code 挂接来实现此目的，如下所示。在此情况下，如果三个同步失败了，则新退出代码为 3。

```
CREATE PROCEDURE sp_hook_dbmlsync_process_exit_code()
BEGIN
   DECLARE  rc INTEGER;

   SELECT value INTO rc FROM #hook_dict WHERE name = 'exit code';
   IF rc <> 0 THEN
      SELECT value INTO rc FROM #hook_dict WHERE name = 'last exit code';
      UPDATE #hook_dict SET value = rc + 1 WHERE name = 'new exit code';
   END IF;
END;
```

另请参见

- “同步事件挂接序列”一节第 227 页
- “sp_hook_dbmlsync_abort”一节第 232 页

sp_hook_dbmlsync_schema_upgrade

您可以使用此存储过程运行修订模式的 SQL 脚本。

#hook_dict 表中各行

名称	值	说明
publication_*n* (in)	发布	正被同步的发布，其中 *n* 是一个整数。每个正在上载的发布都有一个 publication_*n* 条目。*n* 的编号从零开始。
MobiLink user (in)	MobiLink 用户名	您正在为其进行同步的 MobiLink 用户。
script version (in)	脚本版本名称	用于同步的脚本版本。
drop hook (out)	**never** | **always** | **on success**	其值可以是： **never** -（缺省值）不从数据库删除 sp_hook_dbmlsync_schema_upgrade 挂接。 **always** - 尝试运行此挂接后，从数据库删除 sp_hook_dbmlsync_schema_upgrade 挂接。 **on success** - 如果此挂接成功运行，则从数据库删除 sp_hook_dbmlsync_schema_upgrade 挂接。如果使用 dbmlsync -eh 选项，或 dbmlsync 扩展选项 IgnoreHookErrors 设置为 true，则 On success 与 always 相同。

注释

此存储过程旨在对部署的远程数据库进行模式更改。使用该挂接进行模式升级可以确保远程数据库上的所有更改在模式升级前得以同步，这确保了数据库可以继续同步。当使用此挂接时，不应将 dbmlsync 扩展选项 LockTables 设置为 off（LockTables 缺省为 on）。

在任何上载已成功应用并已得到 MobiLink 确认的同步过程中，此挂接会在 sp_hook_dbmlsync_download_end 挂接之后和 sp_hook_dbmlsync_end 挂接之前被调用。在仅下载同步过程中或在创建或应用基于文件的下载时，不会调用此挂接。

此挂接中执行的操作将在挂接完成后立即提交。

另请参见

- “远程客户端中的模式更改”第 75 页

示例

以下的示例使用 sp_hook_dbmlsync_schema_upgrade 过程向远程数据库的 Dealer 表中添加一列。如果升级成功，则删除 sp_hook_dbmlsync_schema_upgrade 挂接。

```
CREATE PROCEDURE sp_hook_dbmlsync_schema_upgrade()
BEGIN
```



```
 -- Upgrade the schema of the Dealer table. Add a column:
 ALTER TABLE Dealer
  ADD dealer_description VARCHAR(128);

 -- If the schema upgrade is successful, drop this hook:
 UPDATE #hook_dict
  SET value = 'on success'
  WHERE name = 'drop hook';
END;
```

sp_hook_dbmlsync_set_extended_options

您可以使用此存储过程，以编程方式自定义即将发生的同步的行为，方法是指定应用于该同步的扩展选项。

#hook_dict 表中各行

名称	值	说明
publication_*n* (in)	发布	正被同步的发布，其中 *n* 是一个整数。每个正在上载的发布都有一个 publication_*n* 条目。*n* 的编号从零开始。
MobiLink user (in)	MobiLink 用户名	您正在为其进行同步的 MobiLink 用户。
extended options (out)	*opt*=*val*;...	要为下一同步添加的扩展选项。

注释

如果以该名称命名的过程存在，则它将在每个同步之前被调用一次或多次。

此挂接指定的扩展选项仅应用于发布和 MobiLink 用户条目标识的同步，并且直到下次为该同步调用此挂接时才应用。

不能使用此挂接指定调度选项。

该过程的操作将在执行后立即被提交。

另请参见

- “同步事件挂接序列”一节第 227 页
- “MobiLink SQL Anywhere 客户端扩展选项”第 175 页
- “优先级顺序”一节第 177 页

示例

以下示例使用 sp_hook_dbmlsync_set_extended_options 指定 SendColumnNames 扩展选项。仅当 pub1 进行同步时应用此扩展选项。

```
CREATE PROCEDURE sp_hook_dbmlsync_set_extended_options ()
BEGIN
 IF exists(SELECT * FROM #hook_dict
  WHERE name LIKE 'publication_%' AND value='pub1')
 THEN
   -- specify the SendColumnNames=on extended option
   UPDATE #hook_dict
          SET value = 'SendColumnNames=on'
    WHERE name = 'extended options';
 END IF;
END;
```


sp_hook_dbmlsync_set_ml_connect_info

使用此存储过程设置网络协议和网络协议选项。

名称	值	说明
publication_*n* (in)	发布	正被同步的发布，其中 *n* 是一个整数。每个正在上载的发布都有一个 publication_*n* 条目。*n* 的编号从零开始。
MobiLink user (in)	MobiLink 用户名	您正在为其进行同步的 MobiLink 用户。
script version (in)	脚本版本名称	将用于同步的 MobiLink 脚本版本。
connection type (in/out)	tcpip、tls、http 或 https	用于连接到 MobiLink 服务器的网络协议。
connection address (in/out)	协议选项	用于连接到 MobiLink 服务器的通信地址。请参见“MobiLink 客户端网络协议选项汇总”一节第 33 页。

注释

可以使用此挂接设置网络协议和网络协议选项。

也可以在同步开始时调用的 sp_hook_dbmlsync_set_extended_options 挂接中设置协议和选项。在 dbmlsync 即将开始尝试连接到 MobiLink 服务器之前调用 sp_hook_dbmlsync_set_ml_connect_info。

如果您想在挂接中设置选项，但是想在 sp_hook_dbmlsync_set_extended_options 之后在同步过程中进行设置，此挂接很有用。例如，如果应根据正在使用的网络的信号强度的可用性设置选项。

另请参见

- “dbmlsync 挂接简介”一节第 227 页
- “同步事件挂接序列”一节第 227 页
- “CommunicationType (ctp) 扩展选项”一节第 180 页
- “MobiLink 客户端网络协议选项汇总”一节第 33 页
- “sp_hook_dbmlsync_set_extended_options”一节第 278 页

示例

```
CREATE PROCEDURE sp_hook_dbmlsync_set_ml_connect_info()
begin
    UPDATE #hook_dict
    SET VALUE = 'tcpip'
      WHERE name = 'connection type';

    UPDATE #hook_dict
    SET VALUE = 'host=localhost'
      WHERE name = 'connection address';
end
```

sp_hook_dbmlsync_set_upload_end_progress

此存储过程可用于在同步脚本式上载预订时定义结束进度。仅在正在同步脚本式上载预订时调用此过程。

#hook_dict 表中各行

名称	值	说明
generating download exclusion list (in)	TRUE | FALSE	如果在同步期间（例如，在仅下载同步中或应用基于文件的下载时）不发送上载，则为 TRUE。在这些情况下仍然调用上载脚本，并且生成的操作用于标识下载操作，这些操作将更改需要被上载的行。找到这样的操作后，将不会应用上载。
publication_*n* (in)	发布	正被同步的发布，其中 *n* 是一个整数。每个正在上载的发布都有一个 publication_*n* 条目。*n* 的编号从零开始。
start progress as timestamp_*n*	进度为时间戳	每个正在被同步的发布的开始进度表示为一个时间戳，其中 *n* 是用于标识发布的同一整数。
开始进度为 bigint_*n*	进度为 bigint	每个正在被同步的发布的开始进度表示为一个 bigint，其中 *n* 是用于标识发布的同一整数。
script version (n)	脚本版本名称	将用于同步的 MobiLink 脚本版本。
MobiLink user (in)	MobiLink 用户名	您正在为其进行同步的 MobiLink 用户。
end progress is bigint (in|out)	TRUE | FALSE	当此行被设置为 TRUE 时，结束进度值被认为是一个表示为字符串的无符号的 bigint（例如，'12345'）。 当此行被设置为 FALSE 时，结束进度值被认为是一个表示为字符串的时间戳（例如，'1900/01/01 12:00:00.000'）。 缺省值为 FALSE。

名称	值	说明
end progress (in|out)	时间戳	此挂接可以修改此行以更改传递到上载脚本的 "end progress as bigint" 和 "end progress as timestamp" 的值。这些值定义了直到所有的操作都包含在正在生成的上载中的时间点。 此行的值可根据 "progress is bigint" 行的设置被设置为无符号的 bigint 或一个时间戳。此行的缺省值为当前时间戳。

注释

对于脚本式上载，每次调用上载过程时它都会传递一个开始进度值和一个结束进度值。上载过程必须返回所有在由这两个值定义的期间发生的相应操作。开始进度值始终与上一次成功同步的结束进度值相同，除非这是一个第一次的同步，在此情况下开始进度值为 January 1, 1900, 00:00:00.000。缺省情况下，结束进度值是 dbmlsync 开始构建上载的时间。

此挂接允许覆盖缺省的结束进度值。您可以为上载定义更短的时期，或根据除时间戳以外的某些内容（例如，世代号）来实施进度跟踪模式。

如果 "end progress is bigint" 被设置为 true，则结束进度必须为一个小于或等于从 1900-01-01 00:00:00 至 9999-12-31 23:59:59:9999 之间的毫秒数（即 255,611,203,259,999）的一个整数。

另请参见

- “脚本式上载中的自定义进度值”一节第 363 页
- “同步事件挂接序列”一节第 227 页
- “脚本式上载”第 355 页

sp_hook_dbmlsync_sql_error

您可以使用此存储过程来处理同步期间出现的数据库错误。例如，您可以实现 sp_hook_dbmlsync_sql_error 挂接，从而在发生 SQL 错误时执行特定操作。

#hook_dict 表中各行

名称	值	说明
publication_*n* (in)	发布	正被同步的发布，其中 *n* 是一个整数。每个正在上载的发布都有一个 publication_*n* 条目。*n* 的编号从零开始。
MobiLink user (in)	MobiLink 用户名	您正在为其进行同步的 MobiLink 用户。
script version (in)	脚本版本名称	将用于同步的 MobiLink 脚本版本。
error message (in)	错误消息文本	这与在 dbmlsync 日志中显示的文本相同。
error id (in)	数字	唯一标识消息的 ID。使用此行来标识错误消息，因为错误消息文本可能发生变化。
error hook user state (in|out)	整数	此值可由挂接设置，以便将状态信息传递到对 sp_hook_dbmlsync_all_error、sp_hook_dbmlsync_communication_error、sp_hook_dbmlsync_misc_error、sp_hook_dbmlsync_sql_error 或 sp_hook_dbmlsync_end 挂接的未来调用。第一次调用这些挂接的其中之一时，该行的值为 0。如果挂接更改了行值，则将在下一个挂接调用中使用新值。 使用此挂接将状态信息传递给 sp_hook_dbmlsync_end 挂接时，可以导致此 _end 挂接执行某些操作（如重试同步）。
SQL code (in)	SQL 错误代码	操作失败时由数据库返回的 SQL 错误代码。这些值在 SQL Anywhere 11 安装目录的 *SDK\Include* 子目录下的 *sqlerr.h* 中定义。
SQL state (in)	SQLSTATE 值	操作失败时由数据库返回的 SQL 状态。

注释

如果在启动过程中还没有初始化同步的时候发生了错误，用于 Mobilink 用户和脚本版本的 #hook_dict 条目将被设置为一个空字符串，并且在 #hook_dict 表中不设置 publication_*n* 行。

您可以使用 SQL Anywhere SQLCODE 或 ANSI SQL 标准 SQLSTATE 来标识 SQL 错误。有关 SQLCODE 或 SQLSTATE 值的列表，请参见“SQL Anywhere 错误消息” 《错误消息》。

此过程将在单独的连接上执行以确保它执行的操作不会在同步连接执行回退时丢失。如果 dbmlsync 无法建立单独的连接，则不调用该过程。

缺省情况下，在 Windows Mobile 设备上，同步表以独占模式被锁定，这意味着如果此挂接需要访问任何同步表，它就不能成功执行。如果它需要访问同步表而您将 dbmlsync 扩展选项 LockTables 设置为 EXCLUSIVE，则它也无法执行。请参见“LockTables (lt) 扩展选项”一节第 194 页。

该过程的操作将在执行后立即被提交。

另请参见

- “处理事件挂接过程中的错误和警告”一节第 231 页
- “sp_hook_dbmlsync_all_error”一节第 234 页
- “sp_hook_dbmlsync_communication_error”一节第 239 页
- “sp_hook_dbmlsync_misc_error”一节第 268 页
- “SQL Anywhere 错误消息” 《错误消息》

sp_hook_dbmlsync_upload_begin

您可以使用此存储过程在上载开始传输之前添加自定义操作。

#hook_dict 表中各行

名称	值	说明
Publication_*n* (in)	发布	正被同步的发布，其中 *n* 是一个整数。每个正在上载的发布都有一个 publication_*n* 条目。*n* 的编号从零开始。
MobiLink user (in)	MobiLink 用户名	您正在为其进行同步的 MobiLink 用户。
Script version (in)	脚本版本名称	将用于同步的 MobiLink 脚本版本。

注释

如果以该名称命名的过程存在，那么它将在 dbmlsync 即将发送上载时被调用。

该过程的操作将在执行后立即被提交。

另请参见

- “同步事件挂接序列”一节第 227 页

示例

假定您使用下面的表记录远程数据库上的同步事件。

```
CREATE TABLE SyncLog
(
 "event_id"           INTEGER NOT NULL DEFAULT AUTOINCREMENT ,
   "event_name"        VARCHAR(128) NOT NULL ,
   "ml_user"             VARCHAR(128) NULL ,
   "event_time"          TIMESTAMP NULL,
   "table_name"          VARCHAR(128) NULL ,
   "upsert_count"        VARCHAR(128) NULL ,
   "delete_count"        VARCHAR(128) NULL ,
   "exit_code"           INTEGER NULL ,
   "status_retval"       VARCHAR(128) NULL ,
   "pubs"                 VARCHAR(128) NULL ,
   "sync_descr "          VARCHAR(128) NULL ,
    PRIMARY KEY ("event_id"),
);
```

以下示例记录即将传输上载时的 MobiLink 用户和当前时间戳。

```
CREATE PROCEDURE sp_hook_dbmlsync_upload_begin ()
BEGIN
 INSERT INTO SyncLog (event_name, ml_user,event_time)
  SELECT 'upload_begin', #hook_dict.value, CURRENT TIMESTAMP
  FROM #hook_dict
  WHERE name = 'MobiLink user';
END;
```


sp_hook_dbmlsync_upload_end

可以使用此存储过程在 dbmlsync 已验证 MobiLink 服务器收到上载后添加自定义操作。

#hook_dict 表中各行

名称	值	说明
failure cause (in)	请参见下面的注释部分中的值范围。	导致上载失败的原因。有关详细信息，请参见说明。
upload status (in)	**retry** | **committed** | **failed** | **unknown**	指定在 dbmlsync 试图验证是否接收到上载时 MobiLink 服务器所返回的状态。 **retry** - MobiLink 服务器与 dbmlsync 中的上载开始位置的日志偏移值不同。MobiLink 服务器未提交上载。dbmlsync 实用程序会尝试发送从新的日志偏移处开始的另一个上载。 **committed** - MobiLink 服务器已接收到上载并已提交。 **failed** - MobiLink 服务器未提交上载。 **unknown** - 启动 dbmlsync 时使用了 -tu 选项，从而引发了事务级上载。对于每个上载的事务，都调用了 sp_hook_dbmlsync_upload_begin 和 sp_hook_dbmlsync_upload_end 挂接，且 upload status 的值为 **unknown** - 除最后一次外，每次都如此。
publication_*n* (in)	发布	正被同步的发布，其中 *n* 是一个整数。每个正在上载的发布都有一个 publication_*n* 条目。*n* 的编号从零开始。
MobiLink user (in)	MobiLink 用户名	您正在为其进行同步的 MobiLink 用户。
script version (in)	脚本版本名称	将用于同步的 MobiLink 脚本版本。
authentication value (in)	值	此值由服务器上的 authenticate_user、authenticate_user_hashed 或 authenticate_parameters 脚本生成。当 upload status 为 unknown，或者由于远程数据库和统一数据库中存储的日志偏移之间存在冲突而重发上载之后调用 upload_end 挂接时，此值为空字符串。

注释

如果以该名称命名的过程存在，那么它将在 dbmlsync 已发送上载并收到来自 MobiLink 服务器的确认后立即被调用。

该过程的操作将在执行后立即被提交。

#hook_dict 表中 failure cause 行参数的可选值为：

- **UPLD_ERR_ABORTED_UPLOAD** 上载由于在远程数据库上发生的错误而失败。失败的原因通常包括通信错误和内存不足情况。
- **UPLD_ERR_COMMUNICATIONS_FAILURE** 发生通信错误。
- **UPLD_ERR_LOG_OFFSET_MISMATCH** 由于远程数据库与统一数据库中存储的日志偏移发生冲突导致上载失败。
- **UPLD_ERR_GENERAL_FAILURE** 未知原因导致上载失败。
- **UPLD_ERR_INVALID_USERID_OR_PASSWORD** 用户 ID 或口令错误。
- **UPLD_ERR_USERID_OR_PASSWORD_EXPIRED** 用户 ID 或口令失效。
- **UPLD_ERR_USERID_ALREADY_IN_USE** 用户 ID 已经在使用中。
- **UPLD_ERR_DOWNLOAD_NOT_AVAILABLE** 上载已经提交到统一数据库，但出现错误导致 MobiLink 无法生成下载。
- **UPLD_ERR_PROTOCOL_MISMATCH** dbmlsync 从 MobiLink 服务器收到意外数据。
- **UPLD_ERR_SQLCODE_n** 其中，*n* 是整数。统一数据库中发生 SQL 错误。指定的整数是遇到的错误的 SQLCODE。

另请参见

- “同步事件挂接序列”一节第 227 页

示例

假定您使用下面的表记录远程数据库上的同步事件。

```
CREATE TABLE SyncLog(
   "event_id"          INTEGER NOT NULL DEFAULT AUTOINCREMENT ,
   "event_name"       VARCHAR(128) NOT NULL ,
   "ml_user"            VARCHAR(128) NULL ,
   "event_time"         TIMESTAMP NULL,
   "table_name"         VARCHAR(128) NULL ,
   "upsert_count"       VARCHAR(128) NULL ,
   "delete_count"       VARCHAR(128) NULL ,
   "exit_code"          INTEGER NULL ,
   "status_retval"      VARCHAR(128) NULL ,
   "pubs"                VARCHAR(128) NULL ,
   "sync_descr "         VARCHAR(128) NULL ,
     PRIMARY KEY ("event_id"),
);
```

以下示例在 dbmlsync 验证 MobiLink 服务器已收到上载之后记录 MobiLink 用户和当前时间戳。

```
CREATE PROCEDURE sp_hook_dbmlsync_upload_end ()
BEGIN
```



```
 DECLARE status_return_value VARCHAR(255);

 -- store status_return_value
  SELECT #hook_dict.value
  INTO status_return_value
  FROM #hook_dict
  WHERE #hook_dict.name = 'upload status';

   INSERT INTO SyncLog (event_name, ml_user,
     status_retval, event_time)
   SELECT 'upload_end', #hook_dict.value,
     status_return_value, CURRENT TIMESTAMP
  FROM #hook_dict
  WHERE name = 'MobiLink user';
END;
```

sp_hook_dbmlsync_validate_download_file

可以使用此挂接实现自定义逻辑，以决定下载文件是否可应用于远程数据库。仅在基于文件的下载被应用时才调用此挂接。

#hook_dict 表中各行

名称	值	说明
publication_*n* (in)	发布	正被同步的发布，其中 *n* 是一个整数。每个正在上载的发布都有一个 publication_*n* 条目。publication_*n* 和 generation number_*n* 中的 *n* 匹配。*n* 的编号从零开始。
MobiLink user (in)	MobiLink 用户名	您正在为其进行同步的 MobiLink 用户。
file last download time (in)		下载文件的上次下载时间。（下载文件中包含在其上次下载时间和更早一次下载时间之间发生更改的所有行。）
file next last download time (in)		下载文件的下一个上次下载时间。（下载文件中包含在其上次下载时间和更早一次下载时间之间发生更改的所有行。）
file creation time (in)		创建下载文件的时间。
file generation number_*n* (in)	数字	来自下载文件的世代号。每个 publication_*n* 条目对应一个 file generation number_*n*。publication_*n* 和 generation number_*n* 中的 *n* 匹配。*n* 的编号从零开始。
user data (in)	字符串	创建下载文件时由 dbmlsync -be 选项指定的字符串。
apply file (in|out)	**True**|**False**	如果为 true（缺省值），则仅当下载文件通过 dbmlsync 的其它校验检查时才应用下载文件。如果为 false，下载文件将不会应用于远程数据库。
check generation number (in|out)	**True**|**False**	如果为 true（缺省值），则 dbmlsync 校验世代号。如果下载文件的世代号与远程数据库的不匹配，则 dbmlsync 不应用下载文件。如果为 false，则 dbmlsync 不检查世代号。
setting generation number (in)	**true** | **false**	如果创建下载文件时使用了 -bg 选项，则为 True。如果使用了 -bg，则远程数据库中的世代号会根据下载文件更新，并且不会执行一般的世代号检查。

注释

您可以使用此存储过程实现自定义的检查，以确定是否能够应用下载文件。

如果要将文件中包含的世代号或时间戳与存储在远程数据库中的世代号或时间戳进行比较，您可以从 SYSSYNC 和 SYSPUBLICATION 系统视图中查询这些值。

当指定 -ba 选项时，将调用此挂接。在下载文件应用于远程数据库之前调用此挂接。

该挂接的操作将在完成后立即被提交。

另请参见

- “-be 选项”一节第 132 页
- “-bg 选项”一节第 133 页
- “MobiLink 基于文件的下载” 《MobiLink - 服务器管理》

示例

下面的示例阻止应用不包含用户字符串 'sales manager data' 的下载文件。

```
CREATE PROCEDURE sp_hook_dbmlsync_validate_download_file ()
BEGIN
  IF NOT exists(SELECT * FROM #hook_dict
   WHERE name = 'User data' AND value='sales manager data')
  THEN
    UPDATE #hook_dict
            SET value = 'false' WHERE name = 'Apply file';
  END IF;
END;
```


Dbmlsync API

目录

Dbmlsync API 简介

Dbmlsync API 提供了一个编程接口，允许使用 C++ 或 .NET 来编写 MobiLink 客户端应用程序，以启动同步并接收有关这些应用程序所请求的同步进度的反馈。API 用于无缝地将同步集成到您的应用程序中。

这种新的编程接口便于您访问同步结果的更多信息，您也可排列同步，使其更易于管理。

体系结构

使用 Dbmlsync API 时，客户端应用程序会实例化并调用 DbmlsyncClient 类中的方法。此类使用 TCP/IP 与独立进程 dbmlsync 服务器进行通信，该服务器通过连接 MobiLink 服务器和远程数据库实际执行同步。由同步生成的状态信息通过 DbmlsyncClient 类的 GetEvent 方法返回到客户端应用程序。

多个客户端可共享同一个 dbmlsync 服务器。但是，每个 dbmlsync 服务器仅能同步单个远程数据库，并且每个远程数据库仅能有一个同步它的 dbmlsync 服务器。

dbmlsync 服务器每次执行一个同步。如果在执行同步时收到同步请求，那么它就会将此请求排队并在稍后执行。

Dbmlsync API 接口

Dbmlsync API 有 2 个版本，一个适用于 C++，另一个适用于 .NET。

C++ 版本是在 DLL dbmlsynccli11.dll 中实现的。想要使用此版本的客户端，应在其 C++ 代码中包含标头 dbmlsynccli.hpp 并链接到导入库 dbmlsynccli11.lib。之后客户端必须使用其应用程序分发 dbmlsynccli11.dll 文件。请参见“适用于 C++ 的 Dbmlsync API”一节第 293 页。

API 的 .NET 版本是在 DLL iAnywhere.MobiLink.Client.dll. 中实现的。.NET 应用程序可通过包含对 DLL 的引用使用接口。请参见“用于 .NET 的 Dbmlsync API”一节第 305 页。

适用于 C++ 的 Dbmlsync API

本节介绍在 DbmlsyncClient 类的 C++ 实现中的方法。

以下示例显示了一个使用 C++ 版本的 Dbmlsync API 执行同步以接收输出事件的典型应用程序。为清晰起见，示例省略了错误处理。检查每个 API 调用的返回值始终是良好的习惯。

Dbmlsync C++ 示例

```
#include <stdio.h>
#include "dbmlsynccli.h"


int main( void ) {
    DbmlsyncClient *client;
    DBSC_SyncHdl    syncHdl;
    DBSC_Event     *ev1;

    client = DbmlsyncClient::InstantiateClient();
    if( client == NULL ) return( 1 );
    client->Init();

    // Setting the "server path" is usually required on Windows Mobile/CE.
    // In other environments the server path is usually not required unless
    // you SA install is not in your path or you have multiple versions of
the
    // product installed
    client->SetProperty( "server path", "C:\\SQLAnywhere\\bin32" );

    client->StartServer( 3426,
                "-c eng=remote;dbn=rem1;uid=dba;pwd=sql -v+ -ot c:\
\dbsync1.txt",
                5000, NULL );
    client->Connect( NULL, 3426, "dba", "sql");
    syncHdl = client->Sync( "my_sync_profile", "" );
    while( client->GetEvent( &ev1, 5000) == DBSC_GETEVENT_OK ) {
        if( ev1->hdl == syncHdl ) {
            //
            // Process events that interest you here
            //

            if( ev1->type == DBSC_EVENTTYPE_SYNC_DONE ) {
                client->FreeEventInfo( ev1 );
                break;
            }
            client->FreeEventInfo( ev1 );
        }
     }
     client->ShutdownServer( DBSC_SHUTDOWN_ON_EMPTY_QUEUE );
     client->WaitForServerShutdown( 10000 );
     client->Disconnect();
     client->Fini();
     delete client;

     return( 0 );
}
```

下面介绍了在 API 的 C++ 版本中的 DbmlsyncClient 类公共方法。

InstantiateClient 方法

调用 InstantiateClient 方法以实例化 DbmlsyncClient 类。

语法

static DbmlsyncClient ***InstantiateClient( );**

注释

此方法返回的指针可用于调用此类中的其余方法。当您使用完由 InstantiateClient 返回的实例之后，可通过调用指针上的删除将其取消。

返回值

将指针返回到已创建的新实例。

如发生错误则返回空值。

Init 方法

初始化类的一个实例。

语法

bool **Init( );**

注释

此方法必须是在使用 InstantiateClient 方法实例化 DbmlSyncClient 类之后所调用的第一个方法。

返回值

如果类实例已成功初始化，则返回 true。

如果类实例未成功初始化，则返回 false。返回 false 时，您可调用 GetErrorInfo 方法获取有关失败的详细信息。只有成功初始化实例后，才能调用其它方法。请参见“GetErrorInfo 方法”一节 第 301 页。

Fini 方法

释放类的实例所使用的所有资源。

语法

bool **Fini**

注释

删除此类的实例前必须调用 Fini 方法。

调用 Fini 方法前，如果实例已连接到服务器，则您必须调用 Disconnect 方法。

返回值

如果方法成功，则返回 true。

如果方法不成功，则返回 false。返回 false 时，您可调用 GetErrorInfo 方法获取有关失败的详细信息。请参见“GetErrorInfo 方法”一节第 301 页。

StartServer 方法

此方法首先检查是否有 dbmlsync 服务器在指定的端口上监听。如果存在服务器，该方法将 starttype 参数设置为 DBSC_SS_ALREADY_RUNNING 并且不执行进一步动作即返回。如果未找到服务器，则此方法使用由 cmdline 参数所指定的选项启动一个新的服务器并在返回前等待其开始接受请求。

注意

在 Windows Mobile 设备中，在成功调用 StartServer 之前，通常需要设置服务器路径属性。在以下实例中不需要设置服务器路径属性：

- 应用程序与 *dbmlsync.exe* 位于同一目录。
- *dbmlsync.exe* 位于 Windows 目录。

语法

bool **StartServer(** unsigned port, const char *cmdline, unsigned timeout, DBSC_Starttype *starttype **);**

参数

- **port** 检查是否已有 dbmlsync 服务器的 TCP 端口。如果启动新服务器，则它会被设置为监听此端口。
- **cmdline** 用于启动 dbmlsync 服务器的有效命令行。此命令行仅能包含以下选项（这些选项对于 dbmlsync 实用程序具有相同的含义）：
  - -a、-c、-dl、-do、-ek、-ep、-k、-l、-o、-os、-ot、-p、-pc+、-pc-、-pd、-pp、-q、-qi、-qc、-sc、-sp、-uc、-ud、-ui、-um、-un、-ux、-v[cnoprsut]、-wc、-wh。请参见“dbmlsync 语法”一节第 123 页。

  必须指定 -c 选项。
- **timeout** dbmlsync 服务器启动后到其可接受请求所需的最长等待时间，以毫秒为单位。使用 DBSC_INFINITY 可一直等待。
- **starttype** 这是一个 OUT 参数。如果 starttype 为非空条目并且 StartServer 返回 true，则退出时 starttype 指向的变量被设置为以下值之一：
  - **DBSC_SS_STARTED** 表示已启动了新的 dbmlsync 服务器。
  - **DBSC_SS_ALREADY_RUNNING** 表示已找到一个现有 dbmlsync 服务器，所以未启动新的服务器。

返回值

如果服务器已经运行或成功启动，则返回 true。

如果服务器未成功启动或者在超时到期之前未开始处理请求，则返回 false。返回 false 时，您可调用 GetErrorInfo 方法获取有关失败的详细信息。请参见“GetErrorInfo 方法”一节第 301 页。

Connect 方法

打开与已在此计算机上运行的 dbmlsync 服务器的连接。

语法

bool **Connect(** const char *host, unsigned port, const char *uid, const char *pwd **);**

注释

传递的数据库用户 id 和口令用来校验此客户端是否有足够的权限来同步数据库。执行同步时，会使用在 dbmlsync 服务器启动时由 -c 选项所指定的用户 id。

参数

- **host** 保留。使用 NULL。
- **port** 服务器所监听的 TCP 端口。使用在启动服务器时通过 StartServer 方法所指定的同一端口值。
- **uid** 在将要被同步的远程数据库上的具有 DBA 或 REMOTE DBA 权限的有效数据库用户 id。
- **pwd** 由 uid 指定的用户的数据库口令。

返回值

如果已建立了到此服务器的连接，则返回 true。

如果无法建立到此服务器的连接，则返回 false。返回 false 时，您可调用 GetErrorInfo 方法获取有关失败的详细信息。请参见“GetErrorInfo 方法”一节第 301 页。

Disconnect 方法

断开使用 Connect 方法创建的、与 dbmlsync 服务器的连接。

语法

bool **Disconnect( )**

注释

当您用完连接之后，一定要调用 Disconnect。

返回值

如果成功断开了到此服务器的连接，则返回 true。

如果无法断开到此服务器的连接，则返回 false。返回 false 时，您可调用 GetErrorInfo 方法获取有关失败的详细信息。请参见“GetErrorInfo 方法”一节第 301 页。

Ping 方法

向 dbmlsync 服务器发送一个 ping 请求来检查连接是否完好以及服务器是否活动并响应请求。

语法

bool **Ping(** unsigned timeout **)**

注释

只有连接到某个服务器才能调用此方法。

参数

- **timeout** 等待服务器响应 ping 请求的最大毫秒数。使用 DBSC_INFINITY 可一直等待。

返回值

如果从服务器收到了对 ping 请求的响应，则返回 true。

如果未收到对 ping 请求的响应，则返回 false。返回 false 时，您可调用 GetErrorInfo 方法获取有关失败的详细信息。请参见“GetErrorInfo 方法”一节第 301 页。

Sync 方法

请求 dbmlsync 服务器执行同步。

语法

DBSC_SyncHdl **Sync(** const char *profile_name, const char *extra_opts **)**

注释

只有连接到服务器才能调用此方法。

profile_name 和 extra_opts 中至少一个必须为非空值。

参数

- **profile_name** 在远程数据库中定义的包含同步选项的同步配置文件的名称。如果 profile_name 为空值，则不会使用配置文件并且 extra_opts 参数应包含所有用于同步的选项。
- **extra_opts** 根据为同步配置文件定义选项字符串所使用的相同规则形成的字符串。如果 profile_name 为非空值，则由 extra_opts 指定的选项将被添加到由 profile_name 指定的同步配置文件中的选项中。如果字符串中的选项已在配置文件中存在，那么字符串的值会替换已存储在配置文件中的值。如果 profile_name 为空值，则 extra_opts 应为同步指定所有选项。

返回值

返回唯一标识此同步请求的 DBSC_SyncHdl 值。返回的句柄只在客户端与服务器断开连接前有效。

如果错误阻止了对同步请求的创建，则返回 NULL_SYNCHDL。返回 NULL_SYNCHDL 时，您可调用 GetErrorInfo 方法获取有关失败的详细信息。请参见“GetErrorInfo 方法”一节第 301 页。

ShutdownServer 方法

关闭与客户端连接的 dbmlsync 服务器。

语法

bool **ShutdownServer(** DBSC_ShutdownType how **)**

注释

Shutdown 方法立即返回，但是在服务器真正关闭前可能会有一些延迟。

调用 ShutdownServer 后仍必须调用 Disconnect。

可使用 WaitForServerShutdown 方法等待，直到服务器真正关闭。请参见“WaitForServerShutdown 方法”一节第 298 页。

参数

- **how** 表示应关闭服务器的迫切程度。支持以下值：
  - **DBSC_SHUTDOWN_ON_EMPTY_QUEUE** 表示服务器应完成所有未完成的同步请求然后关闭。一旦服务器收到关闭请求，它就无法再接受同步请求了。
  - **DBSC_SHUTDOWN_CLEANLY** 表示服务器应尽快完全关闭。不会执行未完成的同步请求，并且可能会中断正在运行的同步。

返回值

如果关闭请求已成功发送到服务器，则返回 true。

如果无法发送关闭请求，则返回 false。返回 false 时，您可调用 GetErrorInfo 方法获取有关失败的详细信息。请参见“GetErrorInfo 方法”一节第 301 页。

WaitForServerShutdown 方法

当服务器已关闭或者当超时到期时（无论何种情况首先出现），WaitForServerShutdown 方法将返回。

语法

bool **WaitForServerShutdown(** unsigned timeout **)**

注释

只有在调用了 ShutdownServer 方法之后才能调用 WaitForServerShutdown。请参见“ShutdownServer 方法”一节第 298 页。

参数

- **timeout** 以毫秒为单位表示等待服务器关闭所需的最长时间。使用 DBSC_INFINITY 可一直等待。

返回值

如果方法由于服务器关闭返回，则返回 true。

否则返回 false。返回 false 时，您可调用 GetErrorInfo 方法获取有关失败的详细信息。请参见“GetErrorInfo 方法”一节第 301 页。

CancelSync 方法

允许客户端取消之前使用 Sync 方法做出的同步请求。

语法

bool **CancelSync(** DBSC_SyncHdl hdl **)**

注释

仅能取消正在等待被执行的同步请求。一旦同步开始，服务器就无法将其取消。

必须建立与服务器的连接才能使用此方法。如果从调用 Sync 方法时开始客户端与服务器已断开连接，则不能使用此方法。

参数

- **hdl** 请求同步时，Sync 方法返回的同步句柄。

返回值

如果已成功取消同步请求，则返回 true。

如果未取消同步请求，则返回 false。返回 false 时，您可调用 GetErrorInfo 方法获取有关失败的详细信息。请参见“GetErrorInfo 方法”一节第 301 页。

GetEvent 方法

当 dbmlsync 服务器运行同步时，它会生成一系列包含同步进度信息的事件。这些事件会从服务器发送到 DbmlsyncClient 类，DbmlsyncClient 类会对其排序。调用 GetEvent 方法时，如果有一个正在等待的事件，则将返回队列中的下一个事件。

语法

DBSC_GetEventRet **GetEvent(** DBSC_Event **event, unsigned timeout **)**

注释

如果队列中没有等待的事件，则此方法将等待事件变为可用或者指定的超时到期后再返回。

可使用属性控制为同步所生成的事件的类型。请参见“SetProperty 方法”一节第 302 页。

参数

- **event** 如果返回值为 DBSC_GETEVENT_OK，则此事件参数是由指向 DBSC_Event 结构（包含关于已检索事件的信息）的指针填充的。当您用完事件结构时，必须调用 FreeEventInfo 方法以释放与其关联的内存。请参见“DBSC_Event 结构”一节第 304 页和“FreeEventInfo 方法”一节第 300 页。
- **timeout** 如果没有可立即返回的事件，则指定等待的最大毫秒数。使用 DBSC_INFINITY 可一直等待。

返回值

返回下列值之一：

- **DBSC_GETEVENT_OK** 表示成功检索了事件。
- **DBSC_GETEVENT_TIMED_OUT** 表示在出现任何可返回的事件之前超时已过期。
- **DBSC_GETEVENT_FAILED** 表示由于出现错误状况而无法返回事件。返回 DBSC_GETEVENT_FAILED 时，您可调用 GetErrorInfo 方法获取有关此方法失败原因的详细信息。请参见“GetErrorInfo 方法”一节第 301 页。

FreeEventInfo 方法

释放与 DBSC_Event 结构（由 GetEvent 方法返回）关联的内存。

语法

bool **FreeEventInfo(** DBSC_Event *event **)**

注释

在每个由 GetEvent 方法返回的 DBSC_Event 结构上必须调用 FreeEventInfo。

参数

- **event** 指向要释放的 DBSC_Event 结构的指针。

返回值

如果内存成功释放，则返回 true。

如果无法释放内存，则返回 false。返回 false 时，您可调用 GetErrorInfo 方法获取有关失败的详细信息。请参见“GetErrorInfo 方法”一节第 301 页。

GetErrorInfo 方法

对于在此接口中紧接着另一方法失败后发生的故障，可使用此方法来检索有关的更多信息。

语法

const DBSC_ErrorInfo ***GetErrorInfo( )**

返回值

返回到包含有关故障信息的 DBSC_ErrorInfo 结构的指针。DBSC_ErrorInfo 结构如下定义：

```
typedef struct {
    DBSC_ErrorType type;
    const char  *str1;
    const char  *str2;
    long int  val1;
    long int  val2;
    DBSC_SyncHdl hdl1;
} DBSC_ErrorInfo;
```

此结构的内容可能会在下次调用类方法时被覆盖。

注释

类型字段包含表示此故障原因的值。目前，类型可以采用下列值：

- **DBSC_ERR_OK** 未发生错误。
- **DBSC_ERR_NOT_INITIALIZED** 类未通过调用 Init 方法初始化。
- **DBSC_ERR_ALREADY_INITIALIZED** 在已初始化的类上调用了 Init 方法。
- **DBSC_ERR_NOT_CONNECTED** 尚不存在到 dbmlsync 服务器的连接。
- **DBSC_ERR_CANT_RESOLVE_HOST** 无法解析主机信息。
- **DBSC_ERR_CONNECT_FAILED** 连接失败。
- **DBSC_ERR_INITIALIZING_TCP_LAYER** 初始化 TCP 层时出错。
- **DBSC_ERR_ALREADY_CONNECTED** 由于连接已存在所以 Connect 方法失败。
- **DBSC_ERR_PROTOCOL_ERROR** 这是一个内部错误。
- **DBSC_ERR_CONNECTION_REJECTED** 连接被 dbmlsync 服务器拒绝。

  str1 指向服务器返回的字符串，该字符串可能会提供有关连接尝试为何被拒绝的详细信息。
- **DBSC_ERR_TIMED _OUT** 等待服务器的响应时超时过期。
- **DBSC_ERR_STILL_CONNECTED** 无法完成 (Fini) 此类，因为其仍与服务器处于连接状态。
- **DBSC_ERR_SYNC_NOT_CANCELED** 服务器无法取消同步请求，可能是由于同步已在进行中。
- **DBSC_ERR_INVALID_VALUE** 已将一个无效的属性值传递给 SetProperty 方法。
- **DBSC_ERR_INVALID_PROP_NAME** 指定的属性名称无效。

- **DBCS_ERR_VALUE_TOO_LONG** 属性值过长。属性的长度必须小于 DBCS_MAX_PROPERTY_LEN 个字节。
- **DBSC_ERR_SERVER_SIDE_ERROR** 服务器上存在错误。

  str1 指向服务器返回的字符串，该字符串可能会提供有关错误的详细信息。
- **DBSC_ERR_CREATE_PROCESS_FAILED** 无法启动新 dbmlsync 服务器。
- **DBSC_ERR_READ_FAILED** 从 dbmlsync 服务器中读取数据时出错。
- **DBSC_ERR_WRITE_FAILED** 将数据发送到 dbmlsync 服务器时出错。
- **DBSC_ERR_NO_SERVER_RESPONSE** 未能从完成请求的操作所需的服务器接收响应。
- **DBSC_ERR_UID_OR_PWD_TOO_LONG** 指定的 UID 或 PWD 过长。
- **DBSC_ERR_UID_OR_PWD_NOT_VALID** 指定的 UID 或 PWD 无效。
- **DBSC_ERR_INVALID_PARAMETER** 传递给此函数的参数之一无效。
- **DBSC_ERR_WAIT_FAILED** 等待服务器关闭时发生错误。
- **DBSC_SHUTDOWN_NOT_CALLED** 在未首先调用 ShutdownServer 方法的情况下调用了 WaitForServerShutdown 方法。
- **DBSC_ERR_NO_SYNC_ACK** 已将同步请求发送到服务器但未收到确认信息。无法获知服务器是否已收到请求。

  hdl1 是发送的 sync 请求的句柄。如果服务器收到此请求，该句柄可用于为使用“GetEvent 方法”一节第 299 页检索的同步标识事件。

str1、str2、val1、val2 和 hdl1 包含有关失败的更多信息，其含义取决于类型值。对于大多数类型值，所有这些字段中的信息都是无用的。例外情况在上述的前一个列表中进行了说明。

SetProperty 方法

允许在类实例上设置修改其各方面行为的属性。

语法

bool **SetProperty(** const char *name, const char *value **)**

注释

对属性的更改仅影响在属性值更改后所做出的同步请求。

可设置服务器路径属性以在调用 StartServer 方法时指定客户端应从中启动 *dbmlsync.exe* 的目录。未设置此属性时，会使用路径环境变量来寻找 *dbmlsync.exe*。请参见“StartServer 方法”一节第 295 页。

以下属性控制由 GetEvent 方法返回的事件类型。通过禁用不需要的事件，可以提高性能。将对应的属性设置为 "1" 可启用事件类型，设置为 "0" 可禁用事件类型。请参见“GetEvent 方法”一节第 299 页。

下面是可用属性和每个可用属性控制的事件类型的列表：

属性名称	控制的事件类型	缺省值
enable errors	DBSC_EVENTTYPE_ERROR_MSG	1
enable warnings	DBSC_EVENTTYPE_WARNING_MSG	1
enable info msgs	DBSC_EVENTTYPE_INFO_MSG	1
enable progress	DBSC_EVENTTYPE_PROGRESS_INDEX	0
enable progress text	DBSC_EVENTTYPE_PROGRESS_TEXT	0
enable title	DBSC_EVENTTYPE_TITLE	0
enable sync start	DBSC_EVENTTYPE_SYNC_START	1
enable sync done	DBSC_EVENTTYPE_SYNC_DONE	1

参数

- **name** 要设置的属性的名称。此名称必须是上述定义的属性名称之一。
- **value** 属性要设置成的值。指定的字符串必须包含少于 DBCS_MAX_PROPERTY_LEN 个字节。

返回值

如果属性设置成功，则返回 true。

如果属性未设置成功，则返回 false。返回 false 时，您可调用 GetErrorInfo 方法获取有关失败的详细信息。请参见“GetErrorInfo 方法”一节第 301 页。

GetProperty 方法

检索属性的当前值。

语法

bool **GetProperty(** const char *name, char *value **)**

参数

- **name** 要检索值的属性的名称。有关有效属性名的列表，请参见“SetProperty 方法”一节第 302 页。
- **value** 一个至少 DBSC_MAX_PROPERTY_LEN 个字节的缓冲区，属性的值在此存储。

返回值

如果属性检索成功，则返回 true。

如果无法检索属性，则返回 false。返回 false 时，您可调用 GetErrorInfo 方法获取有关失败的详细信息。请参见“GetErrorInfo 方法”一节第 301 页。

DBSC_Event 结构

DBSC_Event 结构包含有关已请求同步的信息。该结构如下定义：

```
typedef struct {
    DBSC_SyncHdl hdl;
    DBSC_EventType type;
    const char   *str1;
    const char  *str2;
    long int  val1;
    long int  val2;
    void   *data;
} DBSC_Event;
```

hdl 字段标识结构包含相应信息的同步请求。该值与 Sync 方法返回的句柄相匹配。

类型字段标识所报告的事件的类型。

其余字段包含附加数据，数据的性质取决于类型字段的值。下表列出了可能的类型值以及与每个值相关的其余字段的意义：

- **DBSC_EVENTTYPE_ERROR_MSG** 同步生成了一个错误，并且 str1 指向该错误的文本。
- **DBSC_EVENTTYPE_WARNING_MSG** 同步生成了一个警告，并且 str1 指向该警告的文本。
- **DBSC_EVENTTYPE_INFO_MSG** 同步生成了一个信息性消息，并且 str1 指向该消息的文本。
- **DBSC_EVENTTYPE_PROGRESS_INDEX** 为更新进度条提供信息。val1 包含新的进度值。可通过将 val1 除以 1000 来计算已完成的百分比。
- **DBSC_EVENTTYPE_PROGRESS_TEXT** 已更新与进度条相关的文本，并且 str1 指向新值。
- **DBSC_EVENTTYPE_TITLE** 同步窗口/控件的标题已更改，并且 str1 指向新标题。
- **DBSC_EVENTTYPE_SYNC_START** 同步已开始。没有与此事件相关的更多信息。
- **DBSC_EVENTTYPE_SYNC_DONE** 同步已完成，并且 val1 包含此同步的退出代码。0 值表示操作成功。非零值都表示同步失败。

用于 .NET 的 Dbmlsync API

本节介绍在 DbmlsyncClient 类的 .NET 实现中的方法。

以下示例显示了一个使用 .NET 版本的 Dbmlsync API 执行同步以接收输出事件的典型应用程序。为清晰起见，示例省略了错误处理。检查每个 API 调用的返回值始终是良好的习惯。

Dbmlsync .Net 示例

```
using System;
using System.Collections.Generic;
using System.Text;
using iAnywhere.MobiLink.Client;

namespace ConsoleApplication6
{
    class Program
    {
        static void Main(string[] args)
        {
            DbmlsyncClient cli1;
            DBSC_StartType st1;
            DBSC_Event ev1;
            UInt32 syncHdl;

            cli1 = DbmlsyncClient.InstantiateClient();
            cli1.Init();

            // Setting the "server path" is usually required on Windows
            // Mobile/CE. In other environments the server path is usually
            // not required unless you SA install is not in your path or
            // you have multiple versions of the product installed
            cli1.SetProperty("server path", "d:\\sybase\\asa1100r\\bin32");

            cli1.StartServer(3426,
             "-c eng=cons;dbn=rem1;uid=dba;pwd=sql -ve+ -ot c:\
\dbsync1.txt",
             5000, out st1);
            cli1.Connect(null, 3426, "dba", "sql");
            syncHdl = cli1.Sync("sp1", "");
            while (cli1.GetEvent(out ev1, 5000)
                      == DBSC_GetEventRet.DBSC_GETEVENT_OK)
            {
                if (ev1.hdl == syncHdl)
                {
                    Console.WriteLine("Event Type : {0}", ev1.type);
                    if (ev1.type == DBSC_EventType.DBSC_EVENTTYPE_INFO_MSG)
                    {
                        Console.WriteLine("Info : {0}", ev1.str1);
                    }
                    if (ev1.type == DBSC_EventType.DBSC_EVENTTYPE_SYNC_DONE)
                    {
                        break;
                    }
                }
            }

cli1.ShutdownServer(DBSC_ShutdownType.DBSC_SHUTDOWN_ON_EMPTY_QUEUE);
            cli1.WaitForServerShutdown(10000);
            cli1.Disconnect();
            cli1.Fini();
```

```
                Console.ReadLine();
            }
        }
    }
```

下面介绍了在 API 的 .NET 版本中的 DbmlsyncClient 类公共方法。

InstantiateClient 方法

调用 InstantiateClient 方法以实例化 DbmlsyncClient 类。

语法

static DbmlsyncClient **InstantiateClient()**

注释

此方法返回的对象可用于调用此类中其余的方法。

返回值

返回已创建的新 DbmlsyncClient 实例。如果发生错误，则返回空值。

Init 方法

初始化类的一个实例。

语法

Boolean **Init()**

注释

此方法必须是在使用 InstantiateClient 方法实例化类之后所调用的第一个方法。

返回值

如果类实例已成功初始化，则返回 true。

如果类实例未成功初始化，则返回 false。返回 false 时，您可调用 GetErrorInfo 方法获取有关失败的详细信息。只有成功初始化实例后，才能调用其它方法。请参见“GetErrorInfo 方法”一节 第 312 页。

StartServer 方法

此方法首先检查是否有 dbmlsync 服务器在指定的端口上监听。如果存在服务器，该方法将 starttype 参数设置为 DBSC_SS_ALREADY_RUNNING 并且不执行进一步动作即返回。如果未找到服务器，则此方法使用由 cmdline 参数所指定的选项启动一个新的服务器并在返回前等待其开始接受请求。

注意
在 Windows Mobile 设备中，在成功调用 StartServer 之前，通常需要设置服务器路径属性。在以下实例中不需要设置服务器路径属性：

- 应用程序与 *dbmlsync.exe* 位于同一目录。
- *dbmlsync.exe* 位于 Windows 目录。

语法

Boolean **StartServer(**Int32 port, String cmdline, UInt32 timeout, out DBSC_Starttype starttype **)**

参数

- **port** 检查是否已有 dbmlsync 服务器的 TCP 端口。如果启动新服务器，则它会被设置为监听此端口。
- **cmdline** 用于启动 dbmlsync 服务器的有效命令行。此命令行仅能包含以下选项（这些选项对于 dbmlsync 实用程序具有相同的含义）：
  - -a、-c、-dl、-do、-ek、-ep、-k、-l、-o、-os、-ot、-p、-pc+、-pc-、-pd、-pp、-q、-qi、-qc、-sc、-sp、-uc、-ud、-ui、-um、-un、-ux、-v[cnoprsut]、-wc、-wh。请参见“dbmlsync 语法”一节第 123 页。

  必须指定 -c 选项。
- **timeout** dbmlsync 服务器启动后到其可接受请求所需的最长等待时间，以毫秒为单位。使用 DBSC_INFINITY 可一直等待。DBSC_INFINITY 常量在 DbmlSyncClient 类而不是在命名空间中定义，因此需要为该常量加上前缀。例如，`timeout = DbmlSyncClient.DBSC_INFINITY;`。
- **starttype** 这是一个 OUT 参数。如果在退出时 StartServer 返回 true，则 starttype 将设置为以下值之一：
  - **DBSC_SS_STARTED** 表示已启动了新的 dbmlsync 服务器。
  - **DBSC_SS_ALREADY_RUNNING** 表示已找到一个现有 dbmlsync 服务器，所以未启动新的服务器。

返回值

如果服务器已经运行或成功启动，则返回 true。

如果服务器未成功启动或者在超时到期之前未开始处理请求，则返回 false。返回 false 时，您可调用 GetErrorInfo 方法获取有关失败的详细信息。请参见“GetErrorInfo 方法”一节第 312 页。

Connect 方法

打开与已在此计算机上运行的 dbmlsync 服务器的连接。

语法

Boolean **Connect(**String host, Int32 port, String uid, String pwd**)**

注释

传递的数据库用户 id 和口令用来校验此客户端是否有足够的权限来同步数据库。执行同步时，会在 dbmlsync 服务器启动时使用 -c 选项指定的用户 id。

参数

- **host** 保留。使用空值。
- **port** 服务器所监听的 TCP 端口。使用通过 StartServer 方法启动服务器时所指定的相同端口值。
- **uid** 在将要被同步的远程数据库上的具有 DBA 或 REMOTE DBA 权限的有效数据库用户 id。
- **pwd** 由 uid 指定的用户的数据库口令。

返回值

如果已建立了到此服务器的连接，则返回 true。

如果无法建立到此服务器的连接，则返回 false。返回 false 时，您可调用 GetErrorInfo 方法获取有关失败的详细信息。请参见“GetErrorInfo 方法”一节第 312 页。

Disconnect 方法

断开使用 Connect 方法创建的、与 dbmlsync 服务器的连接。

语法

Boolean **Disconnect()**

注释

当您用完连接之后，一定要调用 Disconnect。

返回值

如果成功断开了到此服务器的连接，则返回 true。

如果无法断开到此服务器的连接，则返回 false。返回 false 时，您可调用 GetErrorInfo 方法获取有关失败的详细信息。请参见“GetErrorInfo 方法”一节第 312 页。

Ping 方法

向 dbmlsync 服务器发送一个 ping 请求来检查连接是否正常以及服务器是否活动并响应请求。

语法

Boolean **Ping(**UInt32 timeout**)**

注释

只有连接到某个服务器才能调用此方法。

参数

- **timeout** 等待服务器响应 ping 请求的最大毫秒数。使用 DBSC_INFINITY 可一直等待。DBSC_INFINITY 常量在 DbmlSyncClient 类而不是在命名空间中定义，因此需要为该常量加上前缀。例如，`timeout = DbmlSyncClient.DBSC_INFINITY;`。

返回值

如果从服务器收到了对 ping 请求的响应，则返回 true。

如果未收到对 ping 请求的响应，则返回 false。返回 false 时，您可调用 GetErrorInfo 方法获取有关失败的详细信息。请参见“GetErrorInfo 方法”一节第 312 页。

Sync 方法

请求 dbmlsync 服务器执行同步。

语法

UInt32 **Sync(** string profile_name, string extra_opts **)**

注释

只有连接到服务器才能调用此方法。

syncName 和 opts 中的至少一个必须为非空值。

参数

- **profile_name** 在远程数据库中定义的包含同步选项的同步配置文件的名称。如果 syncName 为空值，则不会使用配置文件并且 extra_opts 参数应包含所有用于同步的选项。
- **extra_opts** 根据为同步配置文件定义选项字符串所使用的相同规则形成的字符串。在字符串中指定的选项将添加到那些已在由 profile_name 指定的同步配置文件的字符串中。如果字符串中的选项已在配置文件中存在，那么字符串的值会替换已存储在配置文件中的值。如果 profile_name 为空值，则 extra_opts 应为同步指定所有选项。

返回值

返回唯一标识此同步请求的整数值。直到客户端与服务器断开连接，返回的值才会有效。

如果错误阻止了对同步请求的创建，则返回 NULL_SYNCHDL。发生这种情况时，您可调用 GetErrorInfo 方法获取有关失败的详细信息。请参见“GetErrorInfo 方法”一节第 312 页。

ShutdownServer 方法

关闭与客户端连接的 dbmlsync 服务器。

语法

Boolean **ShutdownServer(**DBSC_ShutdownType how**)**

注释

Shutdown 方法立即返回，但是在服务器真正关闭前可能会有一些延迟。

可使用 WaitForServerShutdown 方法等待，直到服务器真正关闭。请参见“WaitForServerShutdown 方法”一节第 310 页。

参数

- **how** 表示应关闭服务器的迫切程度。支持以下值：
  - **DBSC_SHUTDOWN_ON_EMPTY_QUEUE** 表示服务器应完成所有未完成的同步请求然后关闭。一旦服务器收到关闭请求，它就无法再接受同步请求了。
  - **DBSC_SHUTDOWN_CLEANLY** 表示服务器应尽快完全关闭。不会执行未完成的同步请求，并且可能会中断正在运行的同步。

返回值

如果关闭请求已成功发送到服务器，则返回 true。

如果无法发送关闭请求，则返回 false。返回 false 时，您可调用 GetErrorInfo 方法获取有关失败的详细信息。请参见“GetErrorInfo 方法”一节第 312 页。

WaitForServerShutdown 方法

当服务器已关闭或者当超时到期时（无论何种情况首先出现），WaitForServerShutdown 方法将返回。

语法

Boolean **WaitForServerShutdown(** UInt32 timeout **)**

注释

只有在调用了 ShutdownServer 方法之后才能调用 WaitForServerShutdown。请参见“ShutdownServer 方法”一节第 310 页。

参数

- **timeout** 以毫秒为单位表示等待服务器关闭所需的最长时间。使用 DBSC_INFINITY 可一直等待。DBSC_INFINITY 常量在 DbmlSyncClient 类而不是在命名空间中定义，因此需要为该常量加上前缀。例如，`timeout = DbmlSyncClient.DBSC_INFINITY;`。

返回值

如果由于服务器关闭而返回该方法，则返回 true。

否则返回 false。返回 false 时，您可调用 GetErrorInfo 方法获取有关失败的详细信息。请参见“GetErrorInfo 方法”一节第 312 页。

CancelSync 方法

允许客户端取消之前使用 Sync 方法做出的同步请求。

语法

Boolean **CancelSync**(UInt32 hdl)

注释

仅能取消正在等待被执行的同步请求。一旦同步开始，服务器就无法将其取消。

必须建立与服务器的连接才能使用此方法。如果从调用 Sync 方法时开始客户端与服务器已断开连接，则不能使用此方法。

参数

- **hdl** 请求同步时，Sync 方法返回的同步句柄。

返回值

如果已成功取消同步请求，则返回 true。

如果未取消同步请求，则返回 false。返回 false 时，您可调用 GetErrorInfo 方法获取有关失败的详细信息。请参见“GetErrorInfo 方法”一节第 312 页。

GetEvent 方法

当 dbmlsync 服务器运行同步时，它会生成一系列包含同步进度信息的事件。这些事件会从服务器发送到 DbmlsyncClient 类，DbmlsyncClient 类会对其排序。调用 GetEvent 方法时，将返回队列中的下一个事件（如果有正在等待的事件）。

语法

DBSC_GetEventRet **GetEvent**(out DBSC_Event ev, UInt32 timeout)

注释

如果队列中没有等待的事件，则此方法将等待事件变为可用或者指定的超时到期后再返回。

可使用属性控制为同步所生成的事件的类型。请参见“SetProperty 方法”一节第 314 页。

参数

- **ev** 如果返回值为 DBSC_GETEVENT_OK，则此事件由已检索事件的相关信息填充。请参见“DBSC_Event 结构”一节第 304 页。

- **timeout** 如果没有可立即返回的事件，则指定等待的最大毫秒数。使用 DBSC_INFINITY 可一直等待。DBSC_INFINITY 常量在 DbmlSyncClient 类而不是在命名空间中定义，因此需要为该常量加上前缀。例如，`timeout = DbmlSyncClient.DBSC_INFINITY;`。

返回值

返回下列值之一：

返回值	说明
DBSC_GETEVENT_OK	表示成功检索了事件。
DBSC_GETEVENT_TIMED_OUT	表示在出现任何可返回的事件之前超时已过期。
DBSC_GETEVENT_FAILED	表示由于出现错误状况而无法返回事件。返回 DBSC_GETEVENT_FAILED 时，您可调用 GetErrorInfo 方法获取有关失败的详细信息。请参见“GetErrorInfo 方法”一节 第 312 页。

GetErrorInfo 方法

可在此接口中的另一方法失败后立即使用此方法来检索附加信息。

语法

DBSC_ErrorInfo **GetErrorInfo()**

返回值

返回到包含有关故障信息的 DBSC_ErrorInfo 结构的指针。此结构的内容可能会在下次调用类方法时被覆盖。DBSC_ErrorInfo 结构如下定义：

```
public struct DBSC_ErrorInfo
 {
   public DBSC_ErrorType type;
   public String str1;
   public String str2;
   public Int32 val1;
   public Int32 val2;
   public UInt32 hdl1;
 }
```

注释

类型字段包含表示失败原因的值。目前，类型可以采用下列值：

类型值	说明
DBSC_ERR_OK	未发生错误。

类型值	说明
DBSC_ERR_NOT_INITIALIZED	类未通过调用 init 方法初始化。
DBSC_ERR_ALREADY_INITIALIZED	在已初始化的类上调用了 init 方法。
DBSC_ERR_NOT_CONNECTED	尚不存在到 dbmlsync 服务器的连接。
DBSC_ERR_CANT_RESOLVE_HOST	无法解析主机信息。
DBSC_ERR_CONNECT_FAILED	连接失败。
DBSC_ERR_INITIALIZING_TCP_LAYER	初始化 TCP 层时出错。
DBSC_ERR_ALREADY_CONNECTED	由于连接已存在所以连接操作失败。
DBSC_ERR_PROTOCOL_ERROR	这是一个内部错误。
DBSC_ERR_CONNECTION_REJECTED	连接被 dbmlsync 服务器拒绝。
DBSC_ERR_TIMED _OUT	等待服务器的响应时超时过期。
DBSC_ERR_STILL_CONNECTED	无法完成 (Fini) 此类，因为其仍与服务器处于连接状态。
DBSC_ERR_SYNC_NOT_CANCELED	服务器无法取消同步请求，可能是由于同步已在进行中。
DBSC_ERR_INVALID_VALUE	已将一个无效的属性值传递给 SetProperty 方法。
DBSC_ERR_INVALID_PROP_NAME	指定的属性名称无效。
DBCS_ERR_VALUE_TOO_LONG	属性值过长。属性的长度必须小于 DBCS_MAX_PROPERTY_LEN 个字节。
DBSC_ERR_SERVER_SIDE_ERROR	服务器上存在错误。
DBSC_ERR_CREATE_PROCESS_FAILED	无法启动新 dbmlsync 服务器。
DBSC_ERR_READ_FAILED	从 dbmlsync 服务器中读取数据时出错。
DBSC_ERR_WRITE_FAILED	将数据发送到 dbmlsync 服务器时出错。
DBSC_ERR_NO_SERVER_RESPONSE	未能从完成请求的操作所需的服务器接收响应。

类型值	说明
DBSC_ERR_UID_OR_PWD_TOO_LONG	指定的 UID 或 PWD 过长。
DBSC_ERR_UID_OR_PWD_NOT_VALID	指定的 UID 或 PWD 无效。
DBSC_ERR_INVALID_PARAMETER	传递给此函数的参数之一无效。
DBSC_ERR_WAIT_FAILED	等待服务器关闭时发生错误。
DBSC_SHUTDOWN_NOT_CALLED	在未首先调用 ShutdownServer 方法的情况下调用了 WaitForServerShutdown 方法。
DBSC_ERR_NO_SYNC_ACK	已将同步请求发送到服务器但未收到确认信息。无法获知服务器是否已收到请求。
DBSC_ERR_CONNECTION_REJECTED	str1 指向服务器返回的字符串，该字符串可能会提供有关连接尝试为何被拒绝的详细信息。
DBSC_ERR_NO_SYNC_ACK	hdl1 是发送的 sync 请求的句柄。如果服务器收到此请求，该句柄可用于为使用 GetEvent 方法检索的同步标识事件。
DBSC_ERR_SERVER_SIDE_ERROR	str1 指向服务器返回的字符串，该字符串可能会提供有关错误的详细信息。

str1、str2、val1、val2 和 hdl1 包含有关故障的更多信息，其含义取决于类型值。对于大多数类型值，所有这些字段中的信息都是无用的。以下类型值除外：

- DBSC_ERR_CONNECTION_REJECTED
- DBSC_ERR_NO_SYNC_ACK
- DBSC_ERR_SERVER_SIDE_ERROR

SetProperty 方法

允许在类实例上设置修改其各方面行为的属性。

语法

Boolean **SetProperty(**String name, String Value**)**

注释

对属性的更改仅影响在属性值更改后所做出的同步请求。

可设置服务器路径属性以在调用 StartServer 方法时指定客户端应从中启动 *dbmlsync.exe* 的目录。未设置此属性时，会使用路径环境变量来寻找 *dbmlsync.exe*。请参见“StartServer 方法”一节第 306 页。

以下属性控制由“GetEvent 方法”一节第 311 页返回的事件类型。通过禁用不需要的事件，可以提高性能。将对应的属性设置为 "1" 可启用事件类型，设置为 "0" 可禁用事件类型。下面是可用属性和每个可用属性控制的事件类型的列表：

属性名称	控制的事件类型	缺省值
enable errors	DBSC_EVENTTYPE_ERROR_MSG	1
enable warnings	DBSC_EVENTTYPE_WARNING_MSG	1
enable info msgs	DBSC_EVENTTYPE_INFO_MSG	1
enable progress	DBSC_EVENTTYPE_PROGRESS_INDEX	0
enable progress text	DBSC_EVENTTYPE_PROGRESS_TEXT	0
enable title	DBSC_EVENTTYPE_TITLE	0
enable sync start	DBSC_EVENTTYPE_SYNC_START	1
enable sync done	DBSC_EVENTTYPE_SYNC_DONE	1

参数

- **name** 要设置的属性的名称。此名称必须是上述定义的属性名称之一。
- **value** 属性要设置成的值。

返回值

如果属性设置成功，则返回 true。

如果属性未设置成功，则返回 false。返回 false 时，您可调用 GetErrorInfo 方法获取有关失败的详细信息。请参见“GetErrorInfo 方法”一节第 312 页。

GetProperty 方法

检索属性的当前值。

语法

Boolean **GetProperty(**String name, out String Value**)**

参数

- **name** 要检索值的属性的名称。有关有效属性名的列表，请参见“SetProperty 方法”一节第 314 页。
- **value** 退出时，属性的值存储在此变量中。

返回值

如果属性检索成功，则返回 true。

如果无法检索属性，则返回 false。返回 false 时，您可调用 GetErrorInfo 方法获取有关失败的详细信息。请参见“GetErrorInfo 方法”一节第 312 页。

Fini 方法

释放类的实例所使用的所有资源。

语法

Boolean **Fini()**

注释

删除此类的实例前必须调用 Fini 方法。

调用 Fini 方法前，如果实例已连接到服务器，则您必须调用 Disconnect 方法。

返回值

如果方法成功，则返回 true。

如果方法不成功，则返回 false。返回 false 时，您可调用 GetErrorInfo 方法获取有关失败的详细信息。请参见“GetErrorInfo 方法”一节第 312 页。

DBSC_Event 结构

DBSC_Event 结构包含有关已请求同步的信息。该结构如下定义：

```
public struct DBSC_Event
 {
   public UInt32 hdl;
   public DBSC_EventType type;
   public String str1;
   public String str2;
   public Int32 val1;
   public Int32 val2;
 }
```

hdl 字段标识结构包含相应信息的同步请求。该值与 Sync 方法返回的值相匹配。

类型字段标识所报告的事件的类型。

其余字段包含附加数据，数据的性质取决于类型字段的值。下表列出了可能的类型值以及与每个值相关的其余字段的意义：

值	说明
DBSC_EVENTTYPE_ERROR_MSG	同步生成了一个错误，并且 str1 包含该错误的文本。
DBSC_EVENTTYPE_WARNING_MSG	同步生成了一个警告，并且 str1 包含该警告的文本。
DBSC_EVENTTYPE_INFO_MSG	同步生成了一个信息消息，并且 str1 包含该消息的文本。
DBSC_EVENTTYPE_PROGRESS_INDEX	为更新进度条提供信息。val1 包含新的进度值。可通过将 val1 除以 1000 来计算已完成的百分比。
DBSC_EVENTTYPE_PROGRESS_TEXT	已更新与进度条关联的文本，并且该文本包含新的值。
DBSC_EVENTTYPE_TITLE	同步窗口/控件的标题已更改，并且 str1 包含新标题。
DBSC_EVENTTYPE_SYNC_START	同步已开始。没有与此事件相关的更多信息。
DBSC_EVENTTYPE_SYNC_DONE	同步已完成，并且 val1 包含此同步的退出代码。0 值表示操作成功。非零值都表示同步失败。

Dbmlsync 集成组件（不建议使用）

目录

注意
已不建议使用 Dbmlsync 集成组件。可以使用 dbmlsync 编程接口代替该组件。请参见“Dbmlsync API”第 291 页。

Dbmlsync 集成组件简介

注意
已不建议使用 Dbmlsync 集成组件。可以使用 dbmlsync 编程接口代替该组件。请参见“Dbmlsync API”第 291 页。

Dbmlsync 集成组件是可用于向应用程序添加同步的 ActiveX。它提供一组属性、事件和方法用以控制 SQL Anywhere 客户端的行为。

Dbmlsync 集成组件有两种形式，这两种形式具有相同的属性、事件和方法：

- 可视化组件，它提供了一种轻松地将标准 dbmlsync 用户界面集成到应用程序中的方法。
- 非可视化组件，它允许您在不使用用户界面的情况下访问组件的功能，也允许您使用自己创建的自定义用户界面访问组件的功能。

使用 Dbmlsync 集成组件时，应用程序可以启动同步并接收有关同步进度的信息，还可以实现基于同步事件的特殊处理。

dbmlsync 的 DBTools 接口

如果不使用 Dbmlsync 集成组件，也可以使用 dbmlsync 的 DBTools 接口。

请参见“数据库工具接口”《SQL Anywhere 服务器 - 编程》。

支持的平台

可以在支持 MobiLink 的所有 Windows 操作系统（包括支持 ActiveX 的 Windows Mobile 版本）上使用 Dbmlsync 集成组件。

支持的开发环境包括 Microsoft Visual Basic 6.0、eMbedded Visual Basic 和 Visual Studio。

有关受支持的平台的列表，请参见 http://www.sybase.com/detail?id=1062617。

设置 Dbmlsync 集成组件

作为 ActiveX，Dbmlsync 集成组件可以在多种编程环境中使用。有关如何设置 Dbmlsync 集成组件的信息，请查阅针对您的编程环境的相关文档。

Dbmlsync 集成组件的方法

DbmlsyncCOM.Dbmlsync 类实现了以下方法：

Run 方法

使用 dbmlsync 命令行选项开始一个或多个同步。

语法

Run(ByVal *cmdLine* As String **)**
Member of **DbmlsyncCOM.Dbmlsync**

参数

cmdLine 指定 dbmlsync 选项的字符串。

注释

有关选项的列表，请参见“dbmlsync 语法”一节第 123 页。

Run 方法立即返回，而不等待同步完成。可使用 DoneExecution 事件来确定同步完成的时间。

cmdLine 参数包含的选项应与通过 dbmlsync 命令行实用程序进行同步时所使用的选项相同。例如，以下命令行等同于 Run 方法调用：

```
dbmlsync -c uid=DBA;pwd=sql
dbmlsync1.Run "-c uid=DBA;pwd=sql"
```

示例

下面的示例为名为 remote1 的远程数据库启动同步。

```
dbmlsync1.Run "-c eng=remote1;uid=DBA;pwd=sql"
```

Stop 方法

请求终止活动的同步。

语法

Stop( )
Member of **DbmlsyncCOM.Dbmlsync**

注释

Stop 方法发出一个终止任何活动同步的请求。该方法会立即返回。

内置于可视化 Dbmlsync 集成组件的停止按钮会自动调用此方法。

示例

下面的示例会停止 Dbmlsync 集成组件实例 dbmlsync1 正在运行的同步。

```
dbmlsync1.Stop
```

Dbmlsync 集成组件的属性

Dbmlsync 集成组件的属性允许您自定义组件的行为和检查正在运行的同步的状态。

Path 属性

指定 *dbmlsync.exe* 的位置。

语法

Public Property **Path( )** As String
Member of **DbmlsyncCOM.Dbmlsync**

注释

如果 *dbmlsync.exe* 位于 Windows PATH 环境变量所指定的目录中，则无需设置此属性。

示例

下面的示例设置 Dbmlsync 集成组件实例的路径。

```
dbmlsync1.Path = "c:\program files\SQL Anywhere 11\bin32"
```

UploadEventsEnabled 属性

启用 UploadRow 事件。

语法

Public Property **UploadEventsEnabled()** As Boolean
Member of **DbmlsyncCOM.Dbmlsync**

注释

处理 UploadRow 事件时，应将此属性设置为 true。缺省值为 false，它禁用 UploadRow 事件。将该属性设置为 true 会降低性能。

请参见“UploadRow 事件”一节第 340 页。

示例

下面的示例将 UploadEventsEnabled 设置为 true：

```
dbmlsync1.UploadEventsEnabled = True
```

DownloadEventsEnabled 属性

启用 DownloadRow 事件。

语法

Public Property **DownloadEventsEnabled( )** As Boolean
Member of **DbmlsyncCOM.Dbmlsync**

注释

处理 DownloadRow 事件时，应将此属性设置为 true。缺省值为 false，它禁用 DownloadRow 事件。将该属性设置为 true 会降低性能。

请参见“DownloadRow 事件”一节第 332 页。

示例

下面的示例将 DownloadEventsEnabled 设置为 true：

```
dbmlsync1.DownloadEventsEnabled = True
```

ErrorMessageEnabled 属性

阻止为 MsgError 类型的消息调用 Message 事件。

语法

Public Property **ErrorMessageEnabled( )** As Boolean
Member of **DbmlsyncCOM.Dbmlsync**

注释

不在 Message 事件中处理错误消息时，应将此属性设置为 false 以提高性能。缺省值为 true，它启用 MsgError 类型的消息来触发 Message 事件。

请参见“Message 事件”一节第 336 页。

示例

下面的示例将 ErrorMessageEnabled 设置为 false：

```
dbmlsync1.ErrorMessageEnabled = False
```

WarningMessageEnabled 属性

阻止为 MsgWarning 类型的消息调用 Message 事件。

语法

Public Property **WarningMessageEnabled( )** As Boolean
Member of **DbmlsyncCOM.Dbmlsync**

注释

不在 Message 事件中处理警告信息时，应将此属性设置为 false 以提高性能。缺省值为 true，它启用 MsgWarning 类型的消息来触发 Message 事件。

请参见“Message 事件”一节第 336 页。

示例

下面的示例将 WarningMessageEnabled 设置为 false：

```
dbmlsync1.WarningMessageEnabled = False
```

InfoMessageEnabled 属性

阻止为 MsgInfo 类型的消息调用 Message 事件。

语法

Public Property **InfoMessageEnabled( )** As Boolean
Member of **DbmlsyncCOM.Dbmlsync**

注释

不在 Message 事件中处理一般的进度信息时，应将此属性设置为 false 以提高性能。缺省值为 true，它启用 MsgInfo 类型的消息来触发 Message 事件。

请参见“Message 事件”一节第 336 页。

示例

下面的示例将 InfoMessageEnabled 设置为 false：

```
dbmlsync1.InfoMessageEnabled = False
```

DetailedInfoMessageEnabled 属性

阻止为 MsgDetailedInfo 类型的消息调用 Message 事件。

语法

Public Property **DetailedInfoMessageEnabled( )** As Boolean
Member of **DbmlsyncCOM.Dbmlsync**

注释

不在 Message 事件中处理详细的进度信息时，应将此属性设置为 false 以提高性能。缺省值为 true，它启用 MsgDetailedInfo 类型的消息来触发 Message 事件。

请参见“Message 事件”一节第 336 页。

示例

下面的示例将 DetailedInfoMessageEnabled 设置为 false：

```
dbmlsync1.DetailedInfoMessageEnabled = False
```


UseVB6Types 属性

使用 Visual Basic 6 时，请将该属性设置为 true 以简化对由 UploadRow 和 DownloadRow 事件返回的行数据的处理。

语法

Public Property **DetailedInfoMessageEnabled( )** As Boolean
Member of **DbmlsyncCOM.Dbmlsync**

注释

Visual Basic 6 不支持 32 位无符号值和所有的 64 位值。IRowTransferData 对象的 ColumnValue 属性可能返回这些类型的数据。将 UseVB6Types 设置为 true 时，这些类型的数据将转换为 Visual Basic 6 所支持的其它类型，从而更容易处理。Uint32 值将转换为双精度值；64 位值将转换为字符串。

另请参见

- “IRowTransferData 接口”一节第 342 页
- “UploadRow 事件”一节第 340 页
- “DownloadRow 事件”一节第 332 页

示例

下面的示例为用于 Visual Basic 6.0 的 Dbmlsync 集成组件实例启用数据类型强制转换：

```
dbmlsync1.UseVB6Types = True
```

ExitCode 属性

返回由最近的 Run 方法调用所启动的同步的退出代码。

语法

Public Property **ExitCode( )** As Integer
Member of **DbmlsyncCOM.Dbmlsync**

注释

ExitCode 属性返回由最近的 Run 方法调用所启动的同步的退出代码。0 表示同步成功。任何其它值都表示同步失败。

> **注意**
> 在触发 DoneExecution 事件前检索该属性值可能会得到无意义的退出代码值。

示例

下面的示例显示触发 DoneExecution 事件时最近的同步尝试返回的退出代码。

```
Private Sub dbmlsync1_DoneExecution() Handles dbmlsync1.DoneExecution
        MsgBox(dbmlsync1.ExitCode)
    End Sub
```

EventChannelSize 属性

指定用于处理方法调用的内部缓冲区的大小。

语法

Public Property **EventChannelSize( )** As Integer
Member of **DbmlsyncCOM.Dbmlsync**

注释

大多数用户永远不需要更改此属性。

DispatchChannelSize 属性

指定用于处理事件信息的内部缓冲区的大小。

语法

Public Property **DispatchChannelSize( )** As Integer
Member of **DbmlsyncCOM.Dbmlsync**

注释

大多数用户永远不需要更改此属性。

Dbmlsync 集成组件的事件

事件为客户端应用程序提供了一种接收和操作有关同步进度的信息的机制。

BeginDownload 事件

在同步的下载阶段开始时，会触发 BeginDownload 事件。

语法

Public Event **BeginDownload( )**
Member of **DbmlsyncCOM.Dbmlsync**

注释

可以使用此事件在同步的下载阶段开始时添加自定义操作。

示例

下面的 Visual Basic .NET 示例在触发 BeginDownload 事件时输出一条消息。

```
Private Sub dbmlsync1_BeginDownload()
Handles dbmlsync1.BeginDownload

        MsgBox("Beginning Download")

End Sub
```

BeginLogScan 事件

在 dbmlsync 即将扫描事务日志以汇编上载时，会触发 BeginLogScan 事件。对于脚本式上载，不会触发此事件。

语法

Public Event **BeginLogScan(** ByVal *rescanLog* As Boolean **)**
Member of **DbmlsyncCOM.Dbmlsync**

参数

rescanLog　如果这是为此同步进行的第一次事务日志扫描，则该值为 false，否则为 true。如果 MobiLink 服务器与 dbmlsync 中关于从何处开始扫描的信息不同，则进行两次日志扫描。

注释

您可以使用此事件在即将为上载而扫描事务日志时添加自定义操作。

示例

下面的 Visual Basic .NET 示例在触发 BeginLogScan 事件时输出一条消息。

```
Private Sub dbmlsync1_BeginLogScan(
 ByVal rescanLog As Boolean
```

```
)
Handles dbmlsync1.BeginLogScan

        MsgBox("Begin Log Scan")

End Sub
```

BeginSynchronization 事件

在每个同步开始时，会触发 BeginSynchronization 事件。

语法

Public Event **BeginSynchronization(_**
  ByVal *userName* As String, _
  ByVal *pubNames* As String _
)
Member of **DbmlsyncCOM.Dbmlsync**

参数

userName　您正在为其进行同步的 MobiLink 用户。

pubNames　正在进行同步的发布。如果有多个发布，则为逗号分隔的列表。

注释

您可以使用此事件在同步开始时添加自定义操作。

示例

以下 Visual Basic .NET 示例在触发 BeginSynchronization 事件时输出一条消息。该消息输出用户名和发布名。

```
Private Sub dbmlsync1_BeginSynchronization(
 ByVal userName As String,
 ByVal pubNames As String
)
Handles dbmlsync1.BeginSynchronization

       MsgBox("Beginning synchronization for: " + userName _
     + " publication: " + pubNames)

End Sub
```

BeginUpload 事件

在即将开始传输上载时，会触发 BeginUpload 事件。

语法

Public Event **BeginUpload( )**
Member of **DbmlsyncCOM.Dbmlsync**

注释

可以使用此事件在上载即将开始传输到 MobiLink 服务器时添加自定义操作。

示例

下面的 Visual Basic .NET 示例在触发 BeginUpload 事件时输出一条消息。

```
Private Sub dbmlsync1_BeginUpload()
Handles dbmlsync1.BeginUpload

        MsgBox("Begin Upload")

End Sub
```

ConnectMobilink 事件

在该组件即将连接到 MobiLink 服务器时，会触发 ConnectMobilink 事件。

语法

Public Event **ConnectMobilink( )**
Member of **DbmlsyncCOM.Dbmlsync**

注释

可以使用此事件在远程数据库即将连接到 MobiLink 服务器时添加自定义操作。在此阶段，dbmlsync 已经生成了上载。

ConnectMobiLink 事件发生在 BeginSynchronization 事件之后。

示例

下面的 Visual Basic .NET 示例在触发 ConnectMobilink 事件时输出一条消息。

```
Private Sub dbmlsync1_ConnectMobilink()
Handles dbmlsync1.ConnectMobilink

        MsgBox("Connecting to the MobiLink server")

End Sub
```

DisconnectMobilink 事件

在组件断开与 MobiLink 服务器的连接之后，会立即触发 DisconnectMobilink 事件。

语法

Public Event **DisconnectMobilink( )**
Member of **DbmlsyncCOM.Dbmlsync**

注释

可以使用此事件在远程数据库断开与 MobiLink 服务器的连接后立即添加自定义操作。

示例

下面的 Visual Basic .NET 示例在触发 DisconnectMobilink 事件时输出一条消息。

```
Private Sub dbmlsync1_DisconnectMobilink()
Handles dbmlsync1.DisconnectMobilink

        MsgBox("Disconnected from the MobiLink server")

End Sub
```

DoneExecution 事件

在 Run 方法调用所启动的所有同步完成时，会触发 DoneExecution 事件。

语法

Public Event **DoneExecution( )**
Member of **DbmlsyncCOM.Dbmlsync**

注释

您可以使用此事件在 Run 方法调用所启动的所有同步完成时添加自定义操作。

示例

下面的 Visual Basic .NET 示例使用 ExitCode 属性，输出由最近的 Run 方法调用启动的同步所返回的退出代码：

```
Private Sub dbmlsync1_DoneExecution()
Handles dbmlsync1.DoneExecution

        MsgBox(dbmlsync1.ExitCode)

End Sub
```

DownloadRow 事件

从 MobiLink 服务器下载了一行时，会触发 DownloadRow 事件。

语法

Public Event **DownloadRow(**
 ByVal *rowData* As DbmlsyncCOM.IRowTransferData
)
Member of **DbmlsyncCOM.Dbmlsync**

参数

rowData　一个包含有关下载的行的详细信息的 IRowTransferData 对象。

有关 IRowTransferData 接口的详细信息，请参见“IRowTransferData 接口”一节第 342 页。

注释

可以使用此事件检查要从 MobiLink 服务器下载的行。

要启用 DownloadRow 事件，请使用 DownloadEventsEnabled 属性。

请参见“DownloadEventsEnabled 属性”一节第 324 页。

在下载行事件中遇到删除操作时，仅主键列中的值可用。

示例

以下 Visual Basic .NET 示例在 DownloadRow 事件中迭代通过行的所有列。它确定某列的值是否为空并输出列名和值。

```
Private Sub dbmlsync1_DownloadRow(
 ByVal rowData As DbmlsyncCOM.IRowTransferData
)
Handles dbmlsync1.DownloadRow

Dim liX As Integer
For liX = 0 To rowData.ColumnCount - 1
    If VarType(rowData.ColumnValue(liX)) <> VariantType.Null Then
        ' output the non-null column value
        MsgBox("Column " + CStr(liX) + ": " + rowData.ColumnName(liX) + _
        ", " + CStr(rowData.ColumnValue(liX)))
    Else
        ' output 'NULL' for the column value
        MsgBox("Column " + CStr(liX) + ": " + rowData.ColumnName(liX) + _
        ", " + "NULL")
    End If
Next liX

End Sub
```

EndDownload 事件

在同步过程的下载阶段结束时，会触发 EndDownload 事件。

语法

Public Event **EndDownload(**
  long *upsertRows*,
  long *deleteRows*
)
Member of **DbmlsyncCOM.Dbmlsync**

参数

upsertRows　指示下载已更新或插入的行数。

deleteRows　指示下载已删除的行数。

注释

您可以使用此事件在同步的下载阶段结束时添加自定义操作。

示例

下面的 Visual Basic .NET 示例在触发 EndDownload 事件时，输出一条消息以及已插入、更新和删除的行数。

```
Private Sub dbmlsync1_EndDownload(
 ByVal upsertRows As Integer,
 ByVal deleteRows As Integer
)
Handles dbmlsync1.EndDownload

    MsgBox("Download complete." + _
     CStr(upsertRows) + "Rows updated or inserted" + _
     CStr(deleteRows) + "Rows deleted")

End Sub
```

EndLogScan 事件

在为上载而进行事务日志扫描后，会立即触发 EndLogScan 事件。对于脚本式上载，不会触发此事件。

语法

Public Event **EndLogScan( )**
Member of **DbmlsyncCOM.Dbmlsync**

注释

可以使用此事件在为上载而进行事务日志扫描后立即添加自定义操作。

示例

下面的 Visual Basic .NET 示例在触发 EndLogScan 事件时输出一条消息。

```
Private Sub dbmlsync1_EndLogScan()
Handles dbmlsync1.EndLogScan

        MsgBox("Scan of transaction log complete...")

End Sub
```

EndSynchronization 事件

在同步完成时，会触发 EndSynchronization 事件。

语法

Public Event **EndSynchronization(**
  ByVal *exitCode* As Integer,
  ByRef *restart* As Boolean
)
Member of **DbmlsyncCOM.Dbmlsync**

参数

exitCode 设置为 0 以外的任何值时，则表示发生了同步错误。

restart 调用该事件时，会将此值设置为 false。如果事件将其值更改为 true，dbmlsync 会重新启动同步。

注释

您可以使用此事件在同步完成时添加自定义操作。

示例

以下 Visual Basic .NET 示例使用 EndSynchronization 事件重新启动最多 5 次失败的同步尝试。如果所有重新启动尝试均告失败，则输出 "All restart attempts failed" 和退出代码。如果同步成功，则输出消息 "同步成功" 和退出代码。

```
' Global variable for the number of restarts
Dim numberOfRestarts As Integer

Private Sub dbmlsync1_EndSynchronization(
 ByVal ExitCode As Integer,
 ByRef restart As Boolean
)
Handles dbmlsync1.EndSynchronization

    If numberOfRestarts < 5 Then
        MsgBox("Restart Number: " + CStr(numberOfRestarts + 1))
        If ExitCode <> 0 Then
            ' restart the failed synchronization
            restart = True
            numberOfRestarts = numberOfRestarts + 1
        Else
            ' the last synchronization succeeded
            MsgBox("Synchronization succeeded. " + _
              "Exit code: " + CStr(ExitCode))
        End If
    Else
        MsgBox("All restart attempts failed. " + _
             "Exit code: " + CStr(ExitCode))
    End If

End Sub
```

EndUpload 事件

在上载传输到 MobiLink 服务器后，会立即触发 EndUpload 事件。

语法

Public Event **EndUpload( )**
Member of **DbmlsyncCOM.Dbmlsync**

注释

可以使用此事件在上载从 dbmlsync 传输到 MobiLink 服务器后立即添加自定义操作。

示例

下面的 Visual Basic .NET 示例在触发 EndUpload 事件时输出一条消息。

```
Private Sub dbmlsync1_EndUpload()
Handles dbmlsync1.EndUpload

    MsgBox("End Upload")

End Sub
```

Message 事件

当 dbmlsync 记录信息时，会触发 Message 事件。

语法

Public Event **Message(_**
 ByVal *msgClass* As DbmlsyncCOM.MessageClass, _
 ByVal *msgID* As Integer, ByVal *msg* As String_
)
Member of **DbmlsyncCOM.Dbmlsync**

参数

msgClass 指示消息的严重级。值可以是：

- **MsgInfo** 包含有关同步的进度信息的消息。
- **MsgDetailedInfo** 类似 MsgInfo，但包含更多的详细信息。
- **MsgWarning** 表示存在潜在问题（但该问题不会妨碍同步成功）的消息。
- **MsgError** 表示存在会妨碍同步成功的问题的消息。

msgID 消息的唯一标识符。如果 msgID 为 0，则消息没有唯一标识符。

msg 消息的文本。

注释

可以使用此事件接收 dbmlsync 记录的信息。如果在生成特定消息时要添加特殊处理，可通过 MsgID 进行检查。这样，您的代码在消息文本更改后将会继续运行。

示例

下面的 Visual Basic .NET 示例将 dbmlsync 记录的消息添加到列表框控件中。

```
Private Sub dbmlsync1_Message(
 ByVal msgClass As DbmlsyncCOM.MessageClass,
 ByVal msgId As Integer, ByVal msg As String
)
Handles dbmlsync1.Message

    Select Case msgClass
        Case DbmlsyncCOM.MessageClass.MsgError
            lstMessages.Items.Add("Error: " + msg)
        Case DbmlsyncCOM.MessageClass.MsgWarning
```



```
            lstMessages.Items.Add("Warning: " + msg)
        Case DbmlsyncCOM.MessageClass.MsgInfo
            lstMessages.Items.Add("Info: " + msg)
        Case DbmlsyncCOM.MessageClass.MsgDetailedInfo
            lstMessages.Items.Add("DetInfo: " + msg)
    End Select

End Sub
```

示例

以下 Visual Basic .NET 示例设置 Message 事件来处理错误。错误消息被添加到称为 lstMessages 的列表框控件中。

```
Private Sub dbmlsync1_Message(ByVal msgClass As DbmlsyncCOM.MessageClass,
ByVal msgId As Integer, ByVal msg As String) Handles dbmlsync1.Message
  If msgClass = DbmlsyncCOM.MessageClass.MsgError Then
   lstMessages.Items.Add("Error: " + msgId.ToString() + " " + msg)
  End If
End Sub
```

要查看可能存在的错误 ID 值，可测试运行 Dbmlsync 集成组件。例如，如果 dbmlsync 返回错误消息 "无法连接到 MobiLink 服务器"，Message 事件将在 lstMessages 中插入以下条目：

```
Error: 14173 Unable to connect to MobiLink server.
```

现在，可以将错误 "无法连接到 MobiLink 服务器" 与错误 ID 14173 相关联。以下示例设置 Dbmlsync 集成组件以在每次发生错误 14173 时重试同步。发生错误 14173 后，Message 事件设置称为 restartSynchronization 的变量并且重置称为 numberOfRestarts 的变量。EndSynchronization 事件会重试同步，最多重试五次。

```
' variables for restarting synchronization
Dim numberOfRestarts As Integer = 0
Dim restartSynchronization As Integer = 0



Private Sub dbmlsync1_Message
(
 ByVal msgClass As DbmlsyncCOM.MessageClass,
 ByVal msgId As Integer, ByVal msg As String ) Handles dbmlsync1.Message

 If msgClass = DbmlsyncCOM.MessageClass.MsgError Then
     lstMessages.Items.Add("Error: " + msgId.ToString() + " " + msg)

     If msgId = 14173 Then
       restartSynchronization = 1
       numberOfRestarts = 0
     End If
 End If
End Sub

Private Sub dbmlsync1_EndSynchronization(ByVal ExitCode As Integer, _
    ByRef restart As Boolean _
    ) Handles dbmlsync1.EndSynchronization


    If restartSynchronization = 1 Then
      If numberOfRestarts < 5 Then
         restart = True
         numberOfRestarts = numberOfRestarts + 1
```

```
        End If
    End If
End Sub
```

ProgressIndex 事件

当 dbmlsync 更新其进度条时，会触发 ProgressIndex 事件。

语法

Public Event **ProgressIndex(_**
 ByVal *index* As Integer, _
 ByVal *max* As Integer _
)
Member of **DbmlsyncCOM.Dbmlsync**

参数

index　一个表示同步进度的整数。

max　最大进度值。已完成的百分比 = `index/max x 100`。如果该值为 0，则自上次触发该事件后，最大值未曾更改。

注释

您可以使用此事件更新进度指示器，例如进度条。

示例

以下 Visual Basic .NET 示例根据 Index 值来更新进度条控件。index 的最大值在同步开始时设置。

```
Private Sub dbmlsync1_ProgressIndex(
 ByVal index As Integer,
 ByVal max As Integer
)
Handles dbmlsync1.ProgressIndex

    If max <> 0 Then
        ProgressBar1.Maximum = max
    End If
    ProgressBar1.Value = index

End Sub
```

ProgressMessage 事件

当同步进度信息更改时，会触发 ProgressMessage 事件。

语法

Public Event **ProgressMessage(** ByVal *msg* As String **)**
Member of **DbmlsyncCOM.Dbmlsync**

参数

msg 新的进度字符串。

注释

可以使用此事件接收通常和 dbmlsync 进度条一起显示的字符串。

示例

下面的 Visual Basic .NET 示例在触发 ProgressMessage 事件时设置进度标签的值。

```
Private Sub dbmlsync1_ProgressMessage(
 ByVal msg As String
)
Handles dbmlsync1.ProgressMessage

    lblProgressMessage.Text = msg

End Sub
```

SetTitle 事件

当状态信息更改时，会触发 SetTitle 事件。在 dbmlsync 实用程序中，此信息显示在标题栏中。

语法

Public Event **SetTitle(** ByVal *title* **)** As String
)
Member of **DbmlsyncCOM.Dbmlsync**

参数

title dbmlsync 窗口标题栏中的标题。

注释

您可以使用此事件接收通常显示在 dbmlsync 窗口中的标题（在其值更改时）。

示例

下面的 Visual Basic .NET 示例在触发 SetTitle 事件时设置 Windows 窗体的标题。

```
Private Sub dbmlsync1_SetTitle(
 ByVal title As String
)
Handles dbmlsync1.SetTitle

        Me.Text = title

End Sub
```

UploadAck 事件

组件接收到来自 MobiLink 服务器的上载确认后，会触发 UploadAck 事件。

语法

Public Event **UploadAck(_**
  ByVal *status* As DbmlsyncCOM.UploadAckStatus _
)
Member of **DbmlsyncCOM.Dbmlsync**

参数

status　指示处理完上载后 MobiLink 返回远程数据库的状态。其值为以下各值之一：

- **StatCommitted**　表示 MobiLink 服务器已收到并已提交上载。
- **StatRetry**　表示 MobiLink 服务器与远程数据库中的上载开始位置的日志偏移值不同。MobiLink 服务器未提交上载。组件会尝试发送从 MobiLink 服务器的日志偏移处开始的另一个上载。
- **StatFailed**　表示 MobiLink 服务器未提交上载。

注释

可以使用此事件在 dbmlsync 接收到来自 MobiLink 服务器的上载确认后添加自定义操作。

示例

在下面的 Visual Basic .NET 示例中，如果在触发 UploadAck 事件时上载已失败，会输出一条消息。

```
Private Sub dbmlsync1_UploadAck(ByVal status As DbmlsyncCOM.UploadAckStatus)
Handles dbmlsync1.UploadAck

    If status = DbmlsyncCOM.UploadAckStatus.StatFailed Then
         MsgBox("Upload Failed")
    End If

End Sub
```

UploadRow 事件

在将一行上载到 MobiLink 服务器时，会触发 UploadRow 事件。

语法

Public Event **UploadRow(**
  ByVal *rowData* As DbmlsyncCOM.IRowTransferData
)
Member of **DbmlsyncCOM.Dbmlsync**

参数

rowData　一个包含有关上载的行的详细信息的 IRowTransferData 对象。

请参见“IRowTransferData 接口”一节第 342 页。

注释

可以使用此事件检查要上载到 MobiLink 服务器的行。

要启用 UploadRow 事件，请使用 UploadEventsEnabled 属性。请参见“UploadEventsEnabled 属性”一节第 324 页。

示例

以下 Visual Basic .NET 示例在 UploadRow 事件中迭代通过行的所有列。它确定某列的值是否为空并输出列名和值。

```
Private Sub dbmlsync1_UploadRow(
 ByVal rowData As DbmlsyncCOM.IRowTransferData
)
Handles dbmlsync1.UploadRow

Dim liX As Integer
For liX = 0 To rowData.ColumnCount - 1
    If VarType(rowData.ColumnValue(liX)) <> VariantType.Null Then
        ' output the non-null column value
        MsgBox("Column " + CStr(liX) + ": " + rowData.ColumnName(liX) + _
        ", " + CStr(rowData.ColumnValue(liX)))
    Else
        ' output 'NULL' for the column value
        MsgBox("Column " + CStr(liX) + ": " + rowData.ColumnName(liX) + _
        ", " + "NULL")
    End If
Next liX

End Sub
```

WaitingForUploadAck 事件

当组件开始等待来自 MobiLink 服务器的上载确认时，会触发 WaitingForUploadAck 事件。

语法

Public Event **WaitingForUploadAck( )**
Member of **DbmlsyncCOM.Dbmlsync**

注释

可以使用此事件在组件等待来自 MobiLink 服务器的上载确认时添加自定义操作。

示例

下面的 Visual Basic .NET 示例在触发 WaitingForUploadAck 事件时输出一条消息。

```
Private Sub dbmlsync1_WaitingForUploadAck()
Handles dbmlsync1.WaitingForUploadAck

    MsgBox("Waiting for Upload Acknowledgement")

End Sub
```

IRowTransferData 接口

Public Interface **IRowTransferData**
Member of **DbmlsyncCOM**

UploadRow 和 DownloadRow 事件接受 DbmlsyncCOM.IRowTransferData 对象，将其用作检查上载和下载的行的参数。此接口定义了详细的行信息，包括表名、行操作和列名。

RowOperation 属性

指定在行上执行的操作。

语法

Public Property **RowOperation( )** As DbmlsyncCOM.RowEventOp
Member of **DbmlsyncCOM.IRowTransferData**

注释

该属性的值可为以下值之一：

OpInsert 该行是插入的。

OpUpdate 该行已更新。

OpDelete 该行已删除。

OpTruncate 表已截断（表中的所有行均已删除）。当 RowOperation 属性取此值时，ColumnName 和 ColumnValue 属性返回的信息无效。

注意：对于 DownloadRow 事件，upsert（更新或插入）操作的值为 OpInsert。

TableName 属性

在其上进行上载或下载操作的表的名称。

语法

Public Property **TableName( )** As String
Member of **DbmlsyncCOM.IRowTransferData**

注释

TableName 属性指定在其上进行上载或下载操作的表的名称。以下示例说明在 UploadRow 事件中如何使用 TableName 属性。

请参见“UploadRow 事件”一节第 340 页。

示例

下面是一个 Visual Basic .NET 示例。


```
Private Sub dbmlsync1_UploadRow(
 ByVal rowData As DbmlsyncCOM.IRowTransferData
)
Handles dbmlsync1.UploadRow

    MsgBox ("Table name:" + rowData.TableName)

End Sub
```

ColumnName 属性

检索在其上进行上载或下载操作的行的列名。

语法

Public Property **ColumnName(**ByVal *index* As Integer**)** As Object
Member of **DbmlsyncCOM.IRowTransferData**

参数

索引 不小于 0 的整数，指定要检索的列名。索引值的范围是从零到 ColumnCount 属性值减 1。

请参见“ColumnCount 属性”一节第 344 页。

注释

可以使用具有相同索引的 ColumnValue 属性检索相关列的值。

示例

以下 Visual Basic .NET 示例在 UploadRow 事件中迭代通过行的所有列。它确定某列的值是否为空并输出列名和值。

请参见“UploadRow 事件”一节第 340 页。

```
Private Sub dbmlsync1_UploadRow(
 ByVal rowData As DbmlsyncCOM.IRowTransferData
)
Handles dbmlsync1.UploadRow

Dim liX As Integer
For liX = 0 To rowData.ColumnCount - 1
    If VarType(rowData.ColumnValue(liX)) <> VariantType.Null Then
        ' output the non-null column value
        MsgBox("Column " + CStr(liX) + ": " + rowData.ColumnName(liX) + _
        ", " + CStr(rowData.ColumnValue(liX)))
    Else
        ' output 'NULL' for the column value
        MsgBox("Column " + CStr(liX) + ": " + rowData.ColumnName(liX) + _
        ", " + "NULL")
    End If
Next liX

End Sub
```

ColumnValue 属性

检索在其上进行上载或下载操作的列的值。

语法

Public Property **ColumnValue(** ByVal *index* As Integer **)** As Object
Member of **DbmlsyncCOM.IRowTransferData**

参数

index　不小于 0 的整数，指定要检索的列值。索引值的范围是从零到 ColumnCount 属性值减 1。

请参见“ColumnCount 属性”一节第 344 页。

注释

当遇到更新操作时，该属性给出的列值是应用更新后的值。

可以使用具有相同索引的 ColumnName 属性检索相关的列名。

BLOB 列的值不能通过此属性获取。遇到 BLOB 列时，ColumnValue 为字符串 "(blob)"。

示例

以下 Visual Basic .NET 示例在 UploadRow 事件中迭代通过行的所有列。它确定某列的值是否为空并输出列名和值。

请参见“UploadRow 事件”一节第 340 页。

```
Private Sub dbmlsync1_UploadRow(
 ByVal rowData As DbmlsyncCOM.IRowTransferData
)
Handles dbmlsync1.UploadRow

Dim liX As Integer
For liX = 0 To rowData.ColumnCount - 1
    If VarType(rowData.ColumnValue(liX)) <> VariantType.Null Then
        ' output the non-null column value
        MsgBox("Column " + CStr(liX) + ": " + rowData.ColumnName(liX) + _
        ", " + CStr(rowData.ColumnValue(liX)))
    Else
        ' output 'NULL' for the column value
        MsgBox("Column " + CStr(liX) + ": " + rowData.ColumnName(liX) + _
        ", " + "NULL")
    End If
Next liX

End Sub
```

ColumnCount 属性

在其上进行上载或下载操作的行所包含的列数。

语法

Public Property **ColumnCount( )** As Integer
Member of **DbmlsyncCOM.IRowTransferData**

注释

ColumnCount 属性指定在其上进行上载或下载操作的行的列数。下面的示例说明在 UploadRow 事件中如何使用 ColumnCount 属性。

请参见“UploadRow 事件”一节第 340 页。

示例

下面是一个 Visual Basic .NET 示例。

```
Private Sub dbmlsync1_UploadRow(
 ByVal rowData As DbmlsyncCOM.IRowTransferData
)
Handles dbmlsync1.UploadRow

    MsgBox "Number of Columns:" + CStr(rowData.ColumnCount)

End Sub
```

dbmlsync 的 DBTools 接口

目录

dbmlsync 的 DBTools 接口简介

数据库工具 (DBTools) 是一个库，可用于将数据库管理（包括同步）集成到应用程序中。所有数据库管理实用程序都是在 DBTools 上构建的。

请参见“数据库工具接口”《SQL Anywhere 服务器 - 编程》。

可以使用 dbmlsync 的 DBTools 接口将同步功能集成到 MobiLink 同步客户端应用程序中。例如，可以使用此接口在自定义用户界面中显示 dbmlsync 输出消息。

dbmlsync 的 DBTools 接口包含以下元素，这些元素允许您配置并运行 MobiLink 同步客户端：

- **a_sync_db 结构** 此结构包含与 dbmlsync 命令行选项相对应的设置，这些设置允许您自定义同步。此结构还包含指向接收同步和进度信息的回调函数的指针。

  请参见“a_sync_db 结构”一节《SQL Anywhere 服务器 - 编程》。

- **a_syncpub 结构** 此结构包含发布信息。可以为同步指定一个发布的链接列表。

  请参见“a_syncpub 结构”一节《SQL Anywhere 服务器 - 编程》。

- **DBSynchronizeLog 函数** 此函数启动同步过程。它的唯一参数是一个指向 a_sync_db 实例的指针。

  请参见“DBSynchronizeLog 函数”一节《SQL Anywhere 服务器 - 编程》。

Dbmlsync 集成组件

您可以使用 dbmlsync API 作为 dbmlsync 的 DBTools 接口的替代方法。

请参见“Dbmlsync API”第 291 页。

设置 dbmlsync 的 DBTools 接口

本节指导您使用 dbmlsync 的 DBTools 接口的基本步骤。

有关 DBTools 库的详细信息，请参见“数据库工具接口简介”一节 《SQL Anywhere 服务器 - 编程》。

有关使用开发环境的导入库的详细信息，请参见“使用数据库工具接口”一节 《SQL Anywhere 服务器 - 编程》。

♦ 使用 DBTools 接口以 C 或 C++ 配置和启动 dbmlsync

1. 包括 DBTools 头文件。

   DBTools 头文件 *dbtools.h* 列出了 DBTools 库的入口点并定义了需要的数据类型。

   ```
   #include "dbtools.h"
   ```

2. 启动 DBTools 接口。

   - 声明并初始化 a_dbtools_info 结构。

     ```
     a_dbtools_info   info;
     short ret;
     ...
     // clear a_dbtools_info fields
     memset( &info, 0, sizeof( info ) );
     info.errorrtn = dbsyncErrorCallBack;
     ```

     dbsyncErrorCallBack 函数用于处理错误消息，在此过程的第 4 步中进行定义。

   - 使用 DBToolsInit 函数初始化 DBTools。

     ```
     ret = DBToolsInit( &info );
     if( ret != 0 ) {
      printf("dbtools initialization failure \n");
     }
     ```

     有关 DBTools 初始化的详细信息，请参见：

     - “使用数据库工具接口”一节 《SQL Anywhere 服务器 - 编程》
     - “a_dbtools_info 结构”一节 《SQL Anywhere 服务器 - 编程》
     - “DBToolsInit 函数”一节 《SQL Anywhere 服务器 - 编程》

3. 初始化 a_sync_db 结构。

   - 声明一个 a_sync_db 实例。例如，声明一个名为 dbsync_info 的实例：

     ```
     a_sync_db dbsync_info;
     ```

   - 清除 a_sync_db 结构字段。

     ```
     memset( &dbsync_info, 0, sizeof( dbsync_info ) );
     ```

   - 设置需要的 a_sync_db 字段。

     ```
     dbsync_info.version = DB_TOOLS_VERSION_NUMBER;
     dbsync_info.output_to_mobile_link = 1;
     ```

```
dbsync_info.default_window_title
  = "dbmlsync dbtools sample";
```

- 设置数据库连接字符串。

  ```
  dbsync_info.connectparms = "uid=DBA;pwd=sql";
  ```

  有关数据库连接参数的详细信息，请参见“-c 选项”一节第 134 页。

- 设置其它 a_sync_db 字段以自定义同步。

  多数字段都对应于 dbmlsync 命令行选项。有关此对应关系的详细关系，请参见 *dbtools.h*。

  在下面的示例中，启用了详细操作。

  ```
  dbsync_info.verbose_upload = 1;
  dbsync_info.verbose_option_info = 1;
  dbsync_info.verbose_row_data = 1;
  dbsync_info.verbose_row_cnts = 1;
  ```

有关 a_sync_db 字段的详细信息，请参见“a_sync_db 结构”一节 《SQL Anywhere 服务器 - 编程》。

4. 创建在同步过程中接收反馈的回调函数，并将这些函数指派给相应的 a_sync_db 字段。

   以下函数使用标准输出流显示 dbmlsync 错误、日志和进度信息。

   有关 DBTools 回调函数的详细信息，请参见“使用回调函数”一节 《SQL Anywhere 服务器 - 编程》。

   - 例如，创建一个名为 dbsyncErrorCallBack 的函数来处理生成的错误消息：

     ```
     extern short _callback dbsyncErrorCallBack( char *str )
     {
         if( str != NULL ) {
             printf( "Error Msg    %s\n", str );
         }
         return 0;
     }
     ```

   - 例如，创建一个名为 dbsyncWarningCallBack 的函数来处理生成的警告消息：

     ```
     extern short _callback dbsyncWarningCallBack( char *str )
     {
         if( str != NULL ) {
             printf( "Warning Msg  %s\n", str );
         }
         return 0;
     }
     ```

   - 例如，创建一个名为 dbsyncLogCallBack 的函数来接收您可能希望记录到文件而不是显示在窗口中的详细的信息性消息：

     ```
     extern short _callback dbsyncLogCallBack( char *str )
     {
         if( str != NULL ) {
             printf( "Log Msg      %s\n", str );
         }
         return 0;
     }
     ```

   - 例如，创建一个名为 dbsyncMsgCallBack 的函数来接收同步过程中生成的信息性消息。


```
extern short _callback dbsyncMsgCallBack( char *str )
{
    if( str != NULL ) {
        printf( "Display Msg  %s\n", str );
    }
    return 0;
}
```

- 例如，创建一个名为 dbsyncProgressMessageCallBack 的函数来接收进度文本。在 dbmlsync 实用程序中，此文本直接显示在进度条上。

```
extern short _callback dbsyncProgressMessageCallBack(
 char *str )
{
    if( str != NULL ) {
        printf( "ProgressText %s\n", str );
    }
    return 0;
}
```

- 例如，创建一个名为 dbsyncProgressIndexCallBack 的函数来接收更新进度指示器或进度条的信息。此函数接收两个参数：
  - **index**　一个表示同步的当前进度的整数。
  - **max**　最大进度值。如果该值为 0，则自上次触发该事件后，最大值未曾更改。

```
extern short _callback dbsyncProgressIndexCallBack
(a_sql_uint32 index, a_sql_uint32 max )
{
    printf( "ProgressIndex    Index %d Max: %d\n",
         index, max );
    return 0;
}
```

对此回调函数的典型调用顺序显示如下

```
// example calling sequence
dbsyncProgressIndexCallBack( 0, 100 );
dbsyncProgressIndexCallBack( 25, 0 );
dbsyncProgressIndexCallBack( 50, 0 );
dbsyncProgressIndexCallBack( 75, 0 );
dbsyncProgressIndexCallBack( 100, 0 );
```

此顺序可使进度条被设置为已完成 0%、已完成 25%、已完成 50%、已完成 75% 和已完成 100%。

- 例如，创建一个名为 dbsyncWindowTitleCallBack 的函数来接收状态信息。在 dbmlsync 实用程序中，此信息显示在标题栏中。

```
extern short _callback dbsyncWindowTitleCallBack(
 char *title )
{
    printf( "Window Title     %s\n", title );
    return 0;
}
```

- 在需要延迟或休眠时调用 dbsyncMsgQueueCallBack 函数。它必须返回以下值之一，这些值在 *dllapi.h* 中定义。

- **MSGQ_SLEEP_THROUGH** 表示例程休眠请求的毫秒数。多数情况下，这是应返回的值。
- **MSGQ_SHUTDOWN_REQUESTED** 表示希望尽快终止同步。
- **MSGQ_SYNC_REQUESTED** 表示例程休眠的时间少于请求的毫秒数，并且如果当前没有同步操作，应立即开始下一个同步。

```
extern short _callback dbsyncMsgQueueCallBack(
   a_sql_uint32 sleep_period_in_milliseconds )
{

 printf( "Sleep %d ms\n", sleep_period_in_milliseconds );
 Sleep( sleep_period_in_milliseconds );
 return MSGQ_SLEEP_THROUGH;
}
```

- 将回调函数指针指派给相应的 a_sync_db 同步结构字段。

```
// set call back functions
dbsync_info.errorrtn    = dbsyncErrorCallBack;
dbsync_info.warningrtn  = dbsyncWarningCallBack;
dbsync_info.logrtn      = dbsyncLogCallBack;
dbsync_info.msgrtn      = dbsyncMsgCallBack;
dbsync_info.msgqueuertn = dbsyncMsgQueueCallBack;
dbsync_info.progress_index_rtn
     = dbsyncProgressIndexCallBack;
dbsync_info.progress_msg_rtn
     = dbsyncProgressMessageCallBack;
dbsync_info.set_window_title_rtn
     = dbsyncWindowTitleCallBack;
```

5. 创建一个 a_syncpub 结构的链接列表以指定应同步哪些发布。

   链接列表中的每个节点对应于 dbmlsync 命令行上的一个 -n 选项实例。

   - 声明一个 a_syncpub 实例。例如，称其为 publication_info：

     ```
     a_syncpub publication_info;
     ```

   - 初始化 a_syncpub 字段，指定要同步的发布。

     例如，在一个同步会话中同时标识 template_p1 和 template_p2 发布：

     ```
     publication_info.next = NULL; // linked list terminates
     publication_info.pub_name = "template_p1,template_p2";
     publication_info.ext_opt  = "sv=template_ver1";
     publication_info.alloced_by_dbsync = 0;
     ```

     这等同于在 dbmlsync 命令行上指定 `-n template_p1,template_p2`。

     使用 ext_opt 字段指定的关联脚本版本提供与 dbmlsync -eu 选项相同的功能。

     请参见“-eu 选项”一节第 145 页。

   - 将发布结构指派给 a_sync_db 实例的 upload_defs 字段。

     ```
     dbsync_info.upload_defs   = &publication_info;
     ```

   可以创建一个 a_syncpub 结构的链接列表。链接列表中的每个 a_syncpub 实例都等同于 dbmlsync 命令行上 -n 选项的一个说明。

请参见“-n 选项”一节第 151 页和“a_syncpub 结构”一节 《SQL Anywhere 服务器 - 编程》。

6. 使用 DBSynchronizeLog 函数运行 dbmlsync。

   在以下代码列表中，sync_ret_val 包含代表成功的返回值 0 或代表失败的返回值非 0。

```
short sync_ret_val;
printf("Running dbmlsync using dbtools interface...\n");
sync_ret_val = DBSynchronizeLog(&dbsync_info);
printf("\n Done... synchronization return value is: %I \n", sync_ret_val);
```

   可以使用相同或不同的参数值多次重复第 6 步。

7. 关闭 DBTools 接口。

   DBToolsFini 函数释放 DBTools 资源。

```
DBToolsFini( &info );
```

   请参见“DBToolsFini 函数”一节 《SQL Anywhere 服务器 - 编程》。

脚本式上载

目录

脚本式上载简介

脚本式上载仅适用于使用 SQL Anywhere 远程数据库的 MobiLink 应用程序。

警告
实现脚本式上载时，dbmlsync 不使用事务日志确定上载内容。因此，如果您的脚本未捕获所有更改，远程数据库上的数据将会丢失。基于上述原因，对于大多数应用程序，建议使用基于日志的同步方法。

在大多数 MobiLink 应用程序中，上载由数据库事务日志确定，以同步自上次上载以来对远程数据库所做的更改。对于大多数应用程序来说这种设计很合理，可确保不丢失远程数据库上的数据。

但是，在某些特殊情况下，您可能要忽略事务日志而自己定义上载。使用脚本式上载可以精确定义要上载的数据。进行脚本式上载时，不必为远程数据库维护事务日志。对于小型设备，事务日志所占据的空间可能非常珍贵。但是，事务日志对于数据库备份和恢复非常重要，并能提高数据库性能。

为实现脚本式上载，需要创建特殊类型的发布，此发布指定您所创建的存储过程的名称。这些存储过程通过返回包含要在统一数据库上插入、更新或删除的行的结果集定义上载。

注意：不要将脚本式上载与上载脚本混淆。上载脚本是统一数据库上的 MobiLink 事件脚本，这些脚本由您编写，用来告诉 MobiLink 服务器对于上载要执行什么操作。使用脚本式上载时，您仍需要编写上载脚本（将上载应用于统一数据库）和下载脚本（确定下载什么）。

适用情形

以下是脚本式上载可能有用的一些情形：

- 远程数据库在存储空间有限的设备上运行，没有足够的存储空间保存事务日志。
- 要从所有远程数据库上载所有数据，以创建新的统一数据库。
- 要编写用于确定将哪些更改上载到统一数据库的自定义逻辑。

警告

在实现脚本式上载之前，确保阅读本章全部内容。具体地讲，注意以下要点：

- 如果未正确设置脚本式上载，会丢失数据。
- 实现脚本式上载时，必须维护或引用通常可由 dbmlsync 替您处理的内容。这些信息包括数据的前映像和后映像以及同步的进度。
- 通过脚本式上载进行同步时需要锁定远程数据库上的表。使用基于日志的同步时，则不需要锁定。
- 使用脚本式上载实现事务性上载极为困难。

设置脚本式上载

以下步骤概述设置脚本式上载所需完成的任务，并假设已设置 MobiLink 同步。

♦ **设置脚本式上载概述**

1. 创建标识要上载的行的存储过程。每个表可定义三个存储过程：上载、插入和删除各一个。

   请参见“定义脚本式上载的存储过程”一节第 363 页。

2. 创建包含关键字 WITH SCRIPTED UPLOAD 并指定存储过程名称的发布。

   请参见“创建脚本式上载发布”一节第 366 页。

> 使用脚本式上载时，强烈建议为 dbmlsync 扩展选项 LockTables 使用缺省设置。
>
> 使用 LockTables 的缺省设置，可使 dbmlsync 在构建上载之前获得所有同步表的锁，从而避免许多与脚本式上载相关的问题。这可防止其它连接在您的脚本构建上载时更改同步表。还可确保所有未提交事务均不影响在您的脚本构建上载期间打开的同步表。

其它入门资源

- “脚本式上载示例”一节第 368 页

脚本式上载的设计注意事项

每行一个操作

上载时，对于一行不能包含多个操作（插入、更新或删除）。但可将多个操作组合为一个上载操作；例如，如果要插入一行，然后进行更新，可用一个插入最终值的操作替代这两个操作。

操作的顺序

上载应用于统一数据库时，将首先应用插入和更新操作，然后应用删除操作。对于给定表中的操作顺序，不能进行任何其它假设。

处理冲突

在多个同步之间在多个数据库上更新行时会发生冲突。MobiLink 服务器可识别冲突，因为上载中的每个更新操作均包含要更新行的前映像。前映像是上次成功上载或下载时该行中所有列的值。如果应用上载时前映像与统一数据库中的值不匹配，MobiLink 服务器便会认为发现冲突。

如果您的应用程序需要冲突检测，并且您使用脚本式上载，则需要在远程数据库上跟踪每行在上次成功上载或下载时的值。这样您就可以上载正确的前映像。

维护前映像数据的一个方法是创建与同步表完全相同的前映像表。然后可在同步表上创建一个触发器，每次执行更新时均填充前映像表。成功上载后，可删除前映像表中的行。

有关实现冲突解决的示例，请参见“脚本式上载示例”一节第 368 页。

不处理冲突

如果不需要处理冲突检测，则可以不跟踪前映像，这样可大大简化应用程序。您可以改用插入操作上载更新。即在统一数据库上编写 upload_insert 脚本，使该脚本在行不存在时插入行，在行已存在时更新行。如果使用 SQL Anywhere 统一数据库，可通过在 upload_insert 脚本的 INSERT 语句中使用 ON EXISTING 子句来实现此功能。

请参见“INSERT 语句”一节 《SQL Anywhere 服务器 - SQL 参考》。

不处理冲突时，如果两个或多个远程数据库更改同一行，则最后同步的数据库将覆盖之前的更改。

处理强制冲突

对于删除操作，上载行的主键必须正确。然而，在大多数情况下，非主键列的值是否与统一数据库中的值匹配并不重要。只有在一种情况下非主键列的值很重要，就是在 MobiLink 服务器上使用了强制冲突模式。在这种情况下，所有列值均传送到统一数据库上的 upload_old_row_insert 脚本。根据实现此脚本的方式不同，非主键列值可能必须要正确。

请参见“强制冲突”一节 《MobiLink - 服务器管理》。

锁定

使用 dbmlsync 扩展选项 LockTables 的缺省设置，可使 dbmlsync 在构建上载之前获得所有同步表的独占锁，从而避免许多与脚本式上载相关的问题。这可防止其它连接在您的脚本构建上载时更改同步表。还可确保所有未提交事务均不影响在您的脚本构建上载期间打开的同步表。

如果必须关闭表锁定，请参见“没有表锁定的脚本式上载”一节第 360 页。

冗余上载

在大多数情况下，每个操作要在远程数据库上正好上载一次。为帮助实现这个目的，MobiLink 为每个预订维护一个进度值。缺省情况下，进度值是 dbmlsync 开始构建最后一个成功上载的时间。使用 sp_hook_dbmlsync_set_upload_end_progress 挂接，可用其它值覆盖此进度值。

请参见“sp_hook_dbmlsync_set_upload_end_progress”一节第 280 页。

每次调用您的一个上载过程时，就通过 #hook_dict 表将值传递给它。其中包括 '开始进度' 值和 '结束进度' 值。这些值定义所构建的上载应包括对远程数据库所做更改的时间段。在 '开始进度' 之前发生的操作已经上载。在 '结束进度' 之后发生的操作应在下一同步期间上载。

未知上载状态

在实现脚本式上载时有一种常见错误，就是所创建的存储过程只能通过使用 sp_hook_dbmlsync_upload_end 或 sp_hook_dbmlsync_end 挂接判断上载是否成功应用于统一数据库。这种方法并不可靠。

例如，下面的示例尝试通过在每一行上使用一位来跟踪该行是否需要上载的方法处理插入。插入行时设置该位，成功提交上载后即在 sp_hook_dbmlsync_upload_end 挂接中清除该位。

```
//
// DO NOT DO THIS!
//
CREATE TABLE t1 (
   pk    integer primary key,
   val      varchar( 256 ),
   to_upload   bit DEFAULT 1
);

CREATE PROCEDURE t1_ins()
RESULT( pk integer, val varchar(256) )
BEGIN
    SELECT pk, val
    FROM t1
    WHERE to_upload = 1;
END;

CREATE PROCEDURE sp_hook_dbmlsync_upload_end()
BEGIN
    DECLARE     upload_status   varchar(256);

    SELECT value
    INTO upload_status
    FROM #hook_dict
    WHERE name = 'upload status';

    if upload_status = 'committed' THEN
        UPDATE t1 SET to_upload = 0;
    END IF
END;

      CREATE PUBLICATION p1 WITH SCRIPTED UPLOAD (
           TABLE t1 USING ( PROCEDURE t1_ins FOR UPLOAD INSERT )
      );
```

此方法在多数情况下有效。如果在发送上载后服务器确认上载前的这段时间，因为发生硬件或软件故障停止了 dbmlsync，此方法就会失败。在这种情况下，上载可应用于统一数据库，但不会调用 sp_hook_dbmlsync_upload_end 挂接，也不会清除 to_upload 位。因此，在下一同步中，将为已经上载的行上载插入。这通常会导致同步失败，因为这会在统一数据库上生成重复主键错误。

还有一种情况会出现问题，就是在发送上载后确认上载前的这段时间丢失与 MobiLink 服务器的通信。在这种情况下，dbmlsync 不知道上载是否已成功应用。Dbmlsync 调用 sp_hook_dbmlsync_upload_end 挂接并将上载状态设置为未知。写入挂接时，这会阻止清除 to_upload 位。如果服务器尚未应用上载，该操作就是正确的。但如果已应用上载，就会同样发生上一段所述的问题。在上述两种情况下，在手工干预解决问题之前，受影响的远程数据库将不能再次同步。

防止下载时数据丢失

使用脚本式上载时，远程数据库中需要上载的数据可能会被从统一数据库中下载的数据覆盖。这会导致对远程数据库所做的更改丢失。如果您的上载过程构建的每个上载都包括在调用 sp_hook_dbmlsync_set_upload_end_progress 挂接之前在远程数据库中提交的所有更改，Dbmlsync 可防止这种数据丢失。

下面的示例显示违反此规则时将如何丢失数据：

时间	
1:05:00	在统一数据库和远程数据库中均存在行 R。在远程数据库中，使用一些新值 R1 更新了行 R 并提交了更改。
1:06:00	在统一数据库中，将行 R 更新为一些新值 R2 并提交了更改。
1:07:00	发生同步。编写上载脚本，以便上载仅包含 1:00:00 之前提交的操作。这违反了我们的规则，因为这会阻止上载构建上载之前发生的所有操作。上载中不包括对行 R 的更改，因为此更改发生在 1:00:00 以后。从服务器收到的下载中包含行 R2。在远程数据库中应用下载时，行 R2 将替换行 R1。在远程数据库上的更新将丢失。

Dbmlsync 使用多种机制来确保下载不会覆盖在调用 sp_hook_dbmlsync_set_upload_end_progress 挂接时尚未提交，或在调用 sp_hook_dbmlsync_set_upload_end_progress 挂接以后提交的任何更改。

在调用此挂接之前提交的任何更改不受保护，并可能在应用下载时被覆盖。但只要更改包括在（在构建下载之前发送的）上载中，更改就会发送到 MobiLink 服务器，然后服务器端脚本就能够在构建下载之前使用统一数据库中的数据解决这个问题。

没有表锁定的脚本式上载

缺省情况下，dbmlsync 在调用任何上载脚本之前锁定正在同步的表并一直维护这些锁到提交下载。通过将扩展选项 LockTables 设置为 off，可防止表锁定。

建议尽可能使用缺省表锁定行为。进行没有表锁定的脚本式上载会显著增加必须考虑的问题，以及创建正确可行解决方案的难度。这种操作应仅由非常了解数据库并发和同步概念的高级用户来尝试。

使用没有表锁定的隔离级别

禁用表锁定时，运行上载存储过程的隔离级别非常重要，因为它将决定处理未提交事务的方式。启用表锁定时，这不是问题，因为表锁定可确保构建上载时已同步表上没有未提交更改。

上载存储过程将在数据库用户（在 dbmlsync 命令行上指定）的缺省隔离级别运行，除非在上载存储过程中明确更改隔离级别。

隔离级别 0 是数据库的缺省隔离级别，但在使用没有表锁定的脚本式上载时，建议您不要在隔离级别 0 运行上载过程。如果实现没有表锁定的脚本式上载并且使用隔离级别 0，则可能上载未提交的更改，这可能导致以下问题：

- 未提交更改可能被回退，从而导致将不正确的数据发送到统一数据库。
- 未提交事务可能不完整，在这种情况下，可能仅上载了部分事务，从而导致统一数据库处于不一致状态。

可以改用隔离级别 1、2、3 或快照。所有这些隔离级别均可确保不上载未提交的事务。

使用隔离级别 1、2 或 3 时，如果表中有未提交更改，则会导致上载存储过程阻塞。因为是在 dbmlsync 连接到 MobiLink 服务器时调用上载存储过程，所以这会阻碍服务器连接。如果使用隔离级别 1，通过在 select 语句中使用 READPAST table-hint 子句，能够避免阻塞。

快照隔离是个好选择，因为它可同时防止阻塞和读取未提交更改。

丢失未提交更改

如果选择放弃表锁定，则必须有在发生同步时处理未提交操作的机制。若要理解其中的原因，请看下面的示例。

假设表正由脚本式上载进行同步。为简单起见，假设仅在上载插入。表包含 insert_time 列，该列是指示每行的插入时间的时间戳。

通过选择表中 insert_time 在上次成功上载之后和开始构建当前上载（即调用 sp_hook_dbmlsync_set_upload_end_progress 挂接的时刻）之前的所有已提交行，构建每个上载。假设发生了以下操作。

时间	
1:00:00	成功进行同步。
1:04:00	行 R 插入到表中但未提交。R 的 insert_time 列设置为 1:04:00。
1:05:00	发生同步。上载插入时间介于 1:00:00 和 1:05:00 之间的行。未上载行 R，因为它未提交。同步进度设置为 1:05:00。
1:07:00	提交在 1:04:00 插入的行。R 的 insert_time 列继续包含 1:04:00。
1:10:00	发生同步。上载插入时间介于 1:05:00 和 1:10:00 之间的行。不上载行 R，因为它的 insert_time 不在范围内。事实上，从未上载行 R。

总而言之，在同步之前发生却在同步之后提交的所有操作都容易这样丢失。

处理未提交的事务

处理未提交事务有一种最简单的方法，就是使用 sp_hook_dbmlsync_set_upload_end_progress 挂接将每个同步的结束进度设置为调用此挂接时的最早未提交事务的开始时间。可使用 sa_transactions 系统过程确定此时间，如下所示：

```
SELECT min( start_time )
FROM sa_transactions()
```

在这种情况下，上载存储过程必须忽略在 sp_hook_dbmlsync_set_upload_end_progress 挂接中使用 sa_transactions 计算并使用 #hook_dict 表传递的结束进度。存储过程应只上载开始进度之后发生的所有已提交操作。这可确保下载不覆盖含有仍需上载的更改的行。还可确保即使有未提交事务也会及时上载操作。

此解决方案可确保不丢失任何操作，但某些操作可能会上载多次。必须将服务器端脚本编写为能够处理多次上载的操作。下面的示例显示了此设置中如何多次上载一行。

时间	
1:00:00	成功进行同步。
2:00:00	插入行 R1 但未提交。
2:10:00	插入行 R2 并提交。
3:00:00	发生同步。上载在 1:00 和 3:00 之间发生的操作。上载行 R2 并将进度设置为 2:00，因为它是最早未提交事务的开始时间。
4:00:00	提交行 R1。
5:00:00	发生同步。上载在 2:00 和 5:00 之间发生的操作并将进度设置为 5:00。上载包含行 R1 和 R2，因为这两行的时间戳都在上载范围内。这样，R2 就上载了两次。

如果统一数据库是 SQL Anywhere，则可以通过在统一数据库的 upload_insert 脚本中使用 INSERT ...ON EXISTING UPDATE 语句处理冗余上载的插入操作。

对于其它统一数据库，可在 upload_insert 脚本调用的存储过程中采用类似的逻辑。只需编写一个检查，检查统一数据库中是否已经存在主键与插入行主键相同的行。如果该行存在，则将其更新；否则插入新行。

服务器端有冲突检测或解决逻辑时，冗余上载的删除和更新操作就成了问题。如果在服务器端编写冲突检测和解决脚本，则这些脚本必须能够处理冗余上载。

如果统一数据库会重复使用主键值，冗余上载的删除就是个大问题。请看下面的事件序列：

1. 将主键为 100 的行 R 插入远程数据库并上载到统一数据库。
2. 在远程数据库上删除行 R 并上载删除操作。
3. 将主键为 100 的新行 R' 插入统一数据库。
4. 再次从远程数据库上载第 2 步中删除行 R 的删除操作。这样很容易导致从统一数据库不恰当地删除 R'。

另请参见

- “sa_transactions 系统过程”一节 《SQL Anywhere 服务器 - SQL 参考》
- “设置隔离级别”一节 《SQL Anywhere 服务器 - SQL 的用法》

定义脚本式上载的存储过程

要实现脚本式上载，需要创建通过返回特定（即包含要更新、插入或删除的行）的结果集来定义上载的存储过程。

调用这些存储过程时，会创建名为 #hook_dict 的临时表，该表有两列：名称和值。该表用于将名称值配对传送到存储过程。存储过程从此表中检索有用信息。

下面的名称值配对已定义：

名称	值	说明
start progress	字符串形式的时间戳	远程数据库上上载完所有更改的截止时间。上载应只反映此时间后发生的操作。
raw start progress	64 位无符号整数	以无符号的整数形式表示的开始进度。
end progress	字符串形式的时间戳	上载时间段的结束时间。上载应只反映此时间前发生的操作。
raw end progress	64 位无符号整数	以无符号的整数形式表示的结束进度。
generating download exclusion list	true|false	如果同步是仅下载或基于文件的同步，则为 True。在此类情况下，不发送任何上载，并且如果下载会影响到脚本式上载存储过程选择的任何行，则不应用下载。（这样可确保需要上载的、在远程数据库上所做的更改不会被下载覆盖。）
publication_*n*	发布名称	正被同步的发布，其中 *n* 是一个整数。*n* 的编号从零开始。
script version	版本名称	将用于同步的 MobiLink 脚本版本。
MobiLink user	MobiLink 用户名	您正在为其进行同步的 MobiLink 用户。

请参见“#hook_dict 表”一节第 229 页。

脚本式上载中的自定义进度值

缺省情况下，传递到脚本式上载过程的开始进度值和结束进度值表示时间戳。缺省情况下，结束进度是 dbmlsync 开始构建上载的时间。同步的开始进度始终是该预订的最近成功上载使用的结束进度。此缺省行为适用于大多数实现。

对于需要不同行为的特殊情况，可使用 sp_hook_dbmlsync_set_upload_end_progress 挂接。使用此挂接，可设置上载使用的结束进度。所选择的结束进度必须大于开始进度。不能变更开始进度。

可在 sp_hook_dbmlsync_set_upload_end_progress 挂接中以时间戳或无符号整数形式指定结束进度。这两种形式的值均可用于上载存储过程。为方便起见，可使用 sa_convert_ml_progress_to_timestamp 和 sa_convert_timestamp_to_ml_progress 函数在这两种形式的进度值之间转换。

请参见：

- “sp_hook_dbmlsync_set_upload_end_progress”一节第 280 页
- “sa_convert_ml_progress_to_timestamp 系统过程”一节 《SQL Anywhere 服务器 - SQL 参考》
- “sa_convert_timestamp_to_ml_progress 系统过程”一节 《SQL Anywhere 服务器 - SQL 参考》

定义插入存储过程

插入存储过程必须返回包含所有要上载列的结果集，这些列在 CREATE PUBLICATION 语句中定义，按 CREATE TABLE 语句中声明的顺序列出。

列顺序

使用下面的查询，可获得名为 T1 的表中的列创建顺序：

```
SELECT column.name
FROM SYSTAB JOIN SYSTABCOL
   WHERE table_name = 't1'
ORDER BY column_id
```

示例

有关如何定义插入存储过程的详细说明，请参见“脚本式上载示例”一节第 368 页。

下面的示例创建一个名为 t1 的表和一个名为 p1 的发布。发布指定 WITH SCRIPTED UPLOAD 并将存储过程 t1_insert 注册为插入过程。在 t1_insert 存储过程的定义中，结果集包含所有在 CREATE PUBLICATION 语句中列出的列，并按在 CREATE TABLE 语句中声明的顺序列出。

```
CREATE TABLE t1(
   //The column ordering is taken from here
   pk integer primary key,
   c1 char( 30),
   c2,   float,
   c3 double );

CREATE PROCEDURE t1_insert ()
RESULT( pk integer, c1 char(30), c3 double )
begin
   ...
end

CREATE PUBLICATION WITH SCRIPTED UPLOAD p1(
   // Order of columns here is ignored
   TABLE t1( c3, pk, c1 ) USING (
     PROCEDURE t1_insert FOR UPLOAD INSERT
   )
)
```


定义删除存储过程

删除存储过程必须返回包含所有要上载列的结果集，这些列在 CREATE PUBLICATION 语句中定义，按 CREATE TABLE 语句中声明的顺序列出。

列顺序

使用下面的查询，可获得名为 T1 的表中的列创建顺序：

```
SELECT column.name
FROM SYSTAB JOIN SYSTABCOL
   WHERE table_name = 't1'
ORDER BY column_id
```

示例

有关如何定义删除存储过程的详细说明，请参见“脚本式上载示例”一节第 368 页。

下面的示例创建一个名为 t1 的表和一个名为 p1 的发布。发布指定 WITH SCRIPTED UPLOAD 并将存储过程 t1_delete 注册为删除过程。在 t1_delete 存储过程的定义中，结果集包含所有在 CREATE PUBLICATION 语句中列出的列，并按在 CREATE TABLE 语句中声明的顺序列出。

```
CREATE TABLE t1(
   //The column ordering is taken from here
   pk integer primary key,
   c1 char( 30),
   c2,   float,
   c3 double );

CREATE PROCEDURE t1_delete ()
RESULT( pk integer, c1 char(30), c3 double )
begin
   ...
end

CREATE PUBLICATION WITH SCRIPTED UPLOAD p1(
   // Order of columns here is ignored
   TABLE t1( c3, pk, c1 ) USING (
     PROCEDURE t1_delete FOR UPLOAD DELETE
   )
)
```

定义更新存储过程

更新存储过程必须返回包含以下两组值的结果集：

- 第一组值指定更新的前映像（上次从 MobiLink 服务器接收或成功上载到 MobiLink 服务器时行中的值）。
- 第二组值指定更新的后映像（统一数据库中行应更新为的目标值）。

这意味着，更新存储过程返回的结果集所包含的列数必定是插入或删除存储过程所包含列数的两倍。

示例

有关如何定义更新存储过程的详细说明，请参见“脚本式上载示例”一节第 368 页。

下面的示例创建一个名为 t1 的表和一个名为 p1 的发布。发布指定 WITH SCRIPTED UPLOAD 并将存储过程 t1_update 注册为更新过程。发布指定要同步三列：pk、c1 和 c3。更新过程返回包含六列的结果集。前三列包含 pk、c1 和 c3 列的前映像；后三列包含上述三列的后映像。请注意，在上述两种情况下，列顺序均为创建表时的列顺序，不是 CREATE PUBLICATION 语句中的列顺序。

```
CREATE TABLE t1(
   //Column ordering is taken from here
   pk integer primary key,
   c1 char( 30),
   c2 float,
   c3 double );

CREATE PROCEDURE t1_update ()
RESULT( preimage_pk integer, preimage_c1 char(30), preimage_c3 double,
postimage_pk integer, postimage_c1 char(30), postimage_c3 double  )
BEGIN
   ...
END

CREATE PUBLICATION WITH SCRIPTED UPLOAD p1 (
         // Order of columns here is ignored
         TABLE t1( c3, pk, c1 ) USING (
   PROCEDURE t1_update FOR UPLOAD UPDATE
   )
)
```

创建脚本式上载发布

若要创建脚本式上载发布，请使用关键字 WITH SCRIPTED UPLOAD 并在 USING 子句中指定存储过程。

如果在脚本式上载发布中未定义表存储过程，则不上载对表的操作。不能使用 ALTER PUBLICATION 将常规发布更改为脚本式上载发布。

示例

下面的发布使用存储过程为名为 t1 和 t2 的两个表上载数据。对于表 t1，上载插入、删除和更新。对于表 t2，仅上载插入。

```
CREATE PUBLICATION pub WITH SCRIPTED UPLOAD (
      TABLE t1 (col1, col2, col3) USING (
         PROCEDURE my.t1_ui FOR UPLOAD INSERT,
         PROCEDURE my.t1_ud FOR UPLOAD DELETE,
         PROCEDURE my.t1_uu FOR UPLOAD UPDATE
      ),
      TABLE t2 USING (
         PROCEDURE my.t2_ui FOR UPLOAD INSERT
      )
   )
```


另请参见

- “CREATE PUBLICATION 语句 [MobiLink] [SQL Remote]”一节 《SQL Anywhere 服务器 - SQL 参考》
- “ALTER PUBLICATION 语句 [MobiLink] [SQL Remote]”一节 《SQL Anywhere 服务器 - SQL 参考》

脚本式上载示例

此示例说明如何设置提供冲突检测的脚本式上载。此示例创建脚本式上载所需的统一数据库和远程数据库、存储过程、发布和预订。所提供的示例既可以只用于阅读，也通过剪切和粘贴文本来运行该示例。

创建统一数据库

创建保存示例文件的目录。例如，将该目录命名为 *scriptedupload*。打开命令提示符，转到该目录。

（在此示例中，我们指定文件名，并假设这些文件在当前目录中。在实际的应用程序中，应指定文件的完整路径。）

运行以下命令创建统一数据库：

```
dbinit consol.db
```

下一步，运行以下命令定义统一数据库的 ODBC 数据源：

```
dbdsn -w dsn_consol -y -c "uid=DBA;pwd=sql;dbf=consol.db;eng=consol"
```

要将数据库用作统一数据库，必须运行一个安装脚本，此脚本添加 MobiLink 使用的系统表、视图和存储过程。以下命令将 consol.db 设置为统一数据库：

```
dbisql -c "dsn=dsn_consol" %sqlany11%\MobiLink\setup\syncsa.sql
```

打开 Interactive SQL 并使用 dsn_consol DSN 连接到 consol.db。运行下面的 SQL 语句。这些语句在统一数据库上创建 employee 表，向该表中插入值，然后创建所需的同步脚本。

```
CREATE TABLE employee (
   id      unsigned integer primary key,
   name    varchar( 256),
   salary  numeric( 9, 2 )
);

INSERT INTO employee VALUES( 100, 'smith', 225000 );
COMMIT;

CALL ml_add_table_script( 'default', 'employee', 'upload_insert',
       'INSERT INTO employee ( id, name, salary ) VALUES ( ?, ?, ? )' );

CALL ml_add_table_script( 'default', 'employee', 'upload_update',
       'UPDATE employee SET name = ?, salary = ? WHERE id = ?' );

CALL ml_add_table_script( 'default', 'employee', 'upload_delete',
       'DELETE FROM employee WHERE id = ?' );

CALL ml_add_table_script( 'default', 'employee', 'download_cursor',
       'SELECT * from employee' );
```

创建远程数据库

在命令提示符处，从示例目录运行以下命令创建远程数据库：

```
dbinit remote.db
```

下一步，运行以下命令定义 ODBC 数据源：

```
dbdsn -w dsn_remote -y -c "uid=dba;pwd=sql;dbf=remote.db;eng=remote"
```


在 Interactive SQL 中，使用 dsn_remote DSN 连接到 remote.db。运行下面一组语句，在远程数据库中创建对象。

首先，创建要同步的表。insert_time 列和 delete_time 列不进行同步，但这两列中包含上载存储过程确定要上载哪些行所使用的信息。

```
CREATE TABLE employee (
     id             unsigned integer primary key,
     name           varchar( 256),
     salary         numeric( 9, 2 ),
     insert_time    timestamp default '1900-01-01'
);
```

接下来，需要定义用于处理上载的存储过程和其它项目。需要分别为插入、删除和更新定义。

处理插入

首先，创建一个触发器，此触发器在插入每一行时设置该行上的 insert_time。此时间戳用于确定自上次同步以来是否插入了行。在 dbmlsync 应用从统一数据库下载的插入时将不触发此触发器，因为在此示例中稍后会将 FireTriggers 扩展选项设置为 off。通过下载插入的行所获得的 insert_time 值为 1900-01-01，这是创建 employee 表时定义的缺省值。此值应始终早于开始进度，以使这些行不会被视为新插入行，也不会在下一同步期间上载。

```
CREATE TRIGGER emp_ins AFTER INSERT ON employee
REFERENCING NEW AS newrow
FOR EACH ROW
BEGIN
    UPDATE employee SET insert_time = CURRENT TIMESTAMP
    WHERE id = newrow.id
END;
```

接下来，创建一个以结果集形式返回所有要上载的插入行的过程。此过程返回所有自上次成功上载以来（根据 insert_time）插入但随后未删除的行。上次成功上载的时间由 #hook_dict 表中的 start progress 值确定。此示例使用 dbmlsync 扩展选项 LockTables 的缺省设置，该设置可使 dbmlsync 锁定正在同步的表。因此，不必排除结束进度后插入的行：在构建上载时，表锁定可防止在结束进度后发生任何操作。

```
CREATE PROCEDURE employee_insert()
RESULT( id  unsigned integer,
          name varchar( 256 ),
          salary numeric( 9,2 )
      )
BEGIN
    DECLARE start_time timestamp;
    SELECT value
    INTO start_time
    FROM #hook_dict
    WHERE name = 'start progress as timestamp';

    // Upload as inserts all rows inserted after the start_time
    // that were not subsequently deleted
    SELECT id, name, salary
    FROM employee e
    WHERE insert_time > start_time AND
       NOT EXISTS( SELECT id FROM employee_delete ed  WHERE ed.id = e.id );

END;
```

处理更新

若要处理上载，必须确保根据构建上载时的开始进度使用正确的前映像。

首先，创建用于维护更新行的前映像的表。生成脚本式上载时将使用前映像。

```
CREATE TABLE employee_preimages (
   id           unsigned integer NOT NULL,
   name         varchar( 256),
   salary       numeric( 9, 2 ),
   img_time     timestamp default CURRENT TIMESTAMP,
   primary key( id, img_time )
);
```

接下来，创建一个触发器，此触发器在更新每一行时存储该行的前映像。与插入触发器一样，下载时不触发此触发器。

请注意，每次更新行时，此触发器均存储前映像行（除非两次更新时间太近，致使它们获得相同的时间戳）。乍看起来这种方法浪费了资源。只有在表中没有行的前映像时才存储前映像，在上载前映像后依靠 sp_hook_dbmlsync_upload_end 挂接将其删除，这种方法更具吸引力。

但是，sp_hook_dbmlsync_upload_end 挂接在这种方法中并不可靠。如果在发送上载后确认上载前的这段时间内由于发生硬件或软件故障停止了 dbmlsync，则不会调用此挂接，从而导致不会从前映像表中删除行，即使这些行已成功上载。另外，如果发生通信故障，dbmlsync 也不会从服务器收到上载确认。在这种情况下，传送到挂接的上载状态为 'unknown'。如果发生这种情况，则挂接将无法判断前映像表是应该清除还是应该保留原样。通过存储多个前映像，将始终能够根据构建上载时的开始进度选择正确的前映像。

```
CREATE TRIGGER emp_upd AFTER UPDATE OF name,salary ON employee
   REFERENCING OLD AS oldrow
   FOR EACH ROW
BEGIN
   INSERT INTO employee_preimages ON EXISTING SKIP VALUES(
      oldrow.id, oldrow.name, oldrow.salary, CURRENT TIMESTAMP );
END;
```

接下来，创建一个用于处理更新的上载过程。此存储过程返回一个所包含列数为其它脚本包含列数两倍的结果集：该结果集包含前映像（上次从 MobiLink 服务器接收或成功上载到 MobiLink 服务器时行中的值）和后映像（要输入到统一数据库中的值）。

前映像是 employee_preimages 中在 start_progress 后记录的最早的一组值。请注意，此示例不能正确处理删除后再次插入的现有行。在更完整的解决方案中，这些将作为更新上载。

```
CREATE PROCEDURE employee_update()
RESULT(
       preimage_id  unsigned integer,
       preimage_name varchar( 256),
       preimage_salary numeric( 9,2 ),
       postimage_id  unsigned integer,
       postimage_name varchar( 256),
       postimage_salary numeric( 9,2 )
      )
BEGIN
    DECLARE start_time timestamp;

    SELECT value
    INTO start_time
    FROM #hook_dict
    WHERE name = 'start progress as timestamp';
```



```
    // Upload as an update all rows that have been updated since
    // start_time that were not newly inserted or deleted.
    SELECT ep.id, ep.name, ep.salary, e.id, e.name, e.salary
    FROM employee e JOIN employee_preimages ep
        ON ( e.id = ep.id )
    // Do not select rows inserted since the start time. These should be
    // uploaded as inserts.
    WHERE insert_time <= start_time
      // Do not upload deleted rows.
      AND NOT EXISTS( SELECT id FROM employee_delete ed  WHERE ed.id = e.id )
      // Select the earliest pre-image after the start time.
      AND ep.img_time = ( SELECT MIN( img_time )
            FROM employee_preimages
            WHERE id = ep.id
            AND img_time > start_time );
END;
```

处理删除

首先，创建一个用于维护删除行列表的表：

```
CREATE TABLE employee_delete (
    id           unsigned integer  primary key NOT NULL,
    name         varchar( 256 ),
    salary       numeric( 9, 2 ),
    delete_time  timestamp
);
```

接下来，创建一个触发器，此触发器在从 employee 表中删除行时填充 employee_delete 表。下载时不调用此触发器，因为稍后会将 dbmlsync 扩展选项 FireTriggers 设置为 false。请注意，此触发器假设不会再插入已删除行；所以它不处理多次删除的行。

```
CREATE TRIGGER emp_del AFTER DELETE ON employee
REFERENCING OLD AS delrow
FOR EACH ROW
BEGIN
     INSERT INTO employee_delete
VALUES( delrow.id, delrow.name, delrow.salary, CURRENT TIMESTAMP );
END;
```

下一个 SQL 语句创建一个用于处理删除的上载过程。此存储过程返回一个包含统一数据库上要删除行的结果集。此存储过程使用 employee_preimages 表，这样如果对行进行更新然后再删除，上载的删除映像是成功下载或上载的最后一个映像。

```
CREATE PROCEDURE employee_delete()
RESULT( id  unsigned integer,
          name varchar( 256),
          salary numeric( 9,2 )
      )
BEGIN
    DECLARE start_time timestamp;

    SELECT value
    INTO start_time
    FROM #hook_dict
    WHERE name = 'start progress as timestamp';

   // Upload as a delete all rows that were deleted after the
   // start_time that were not inserted after the start_time.
   // If a row was updated before it was deleted, then the row
   // to be deleted is the pre-image of the update.
```

```
    SELECT IF ep.id IS NULL THEN ed.id ELSE ep.id ENDIF,
           IF ep.id IS NULL THEN ed.name ELSE ep.name ENDIF,
           IF ep.id IS NULL THEN ed.salary ELSE ep.salary ENDIF
    FROM employee_delete ed LEFT OUTER JOIN employee_preimages ep
          ON( ed.id = ep.id AND ep.img_time > start_time )
    WHERE
      // Only upload deletes that occurred since the last sync.
      ed.delete_time > start_time
      // Don't upload a delete for rows that were inserted since
      // the last upload and then deleted.
    AND NOT EXISTS (
      SELECT id
         FROM employee e
         WHERE e.id = ep.id AND e.insert_time > start_time )
    // Select the earliest preimage after the start time.
    AND ( ep.id IS NULL OR ep.img_time = (SELECT MIN( img_time )
                                          FROM employee_preimages
                                          WHERE id = ep.id
                                           AND img_time > start_time ) );
END;
```

清除前映像表

接下来，创建 upload_end 挂接，以在上载成功后清理 employee_preimage 和 employee_delete 表。此示例使用 dbmlsync 扩展选项 LockTables 的缺省设置，以便同步时锁定这些表。因此，您不必担心在表中留下在 end_progress 后发生的操作的行。锁定会防止发生这些操作。

```
CREATE PROCEDURE sp_hook_dbmlsync_upload_end()
BEGIN
    DECLARE val   varchar(256);

    SELECT value
    INTO val
    FROM #hook_dict
    WHERE name = 'upload status';

    IF val = 'committed' THEN
      DELETE FROM employee_delete;
      DELETE FROM employee_preimages;
    END IF;
END;
```

创建发布、MobiLink 用户和预订

名为 pub1 的发布使用脚本式上载语法 (WITH SCRIPTED UPLOAD)。它为 employee 表创建一个项目，然后注册刚创建的三个存储过程用于脚本式上载。它创建一个名为 u1 的 MobiLink 用户和一个在 v1 和 pub1 之间的预订。将扩展选项 FireTriggers 设置为 off 以防止在远程数据库上应用下载时触发触发器，从而防止下次同步时上载下载的更改。

```
CREATE PUBLICATION pub1 WITH SCRIPTED UPLOAD (
TABLE employee( id, name, salary ) USING (
   PROCEDURE employee_insert FOR UPLOAD INSERT,
   PROCEDURE employee_update FOR UPLOAD UPDATE,
   PROCEDURE employee_delete FOR UPLOAD DELETE,
      )
)

CREATE SYNCHRONIZATION USER u1;

CREATE SYNCHRONIZATION SUBSCRIPTION TO pub1 FOR u1
TYPE 'tcpip'
```



```
ADDRESS 'host=localhost'
OPTION FireTriggers='off';
```

演示脚本式上载

使用脚本式上载连接到远程数据库并插入要同步的数据。例如，在 Interactive SQL 中针对远程数据库运行以下 SQL 语句：

```
INSERT INTO employee(id, name, salary) VALUES( 7, 'black', 700 );
INSERT INTO employee(id, name, salary) VALUES( 8, 'anderson', 800 );
INSERT INTO employee(id, name, salary) VALUES( 9, 'dilon', 900 );
INSERT INTO employee(id, name, salary) VALUES( 10, 'dwit', 1000 );
INSERT INTO employee(id, name, salary) VALUES( 11, 'dwit', 1100 );
COMMIT;
```

在命令提示符处，启动 MobiLink 服务器：

```
mlsrv11 -c "dsn=dsn_consol" -o mlserver.mls -v+ -dl -zu+
```

使用 dbmlsync 启动同步：

```
dbmlsync -c "dsn=dsn_remote" -k -uo -o remote.mlc -v+
```

现在可验证是否已上载插入。

示例清除

若要在完成示例后清理计算机，请执行以下步骤：

```
mlstop -h -w
dbstop -y -c eng=consol
dbstop -y -c eng=remote

dberase -y consol.db
dberase -y remote.db

del remote.mlc mlserver.mls
```

术语表

术语表

Adaptive Server Anywhere (ASA)

SQL Anywhere Studio 的关系数据库服务器组件，专供在移动和嵌入式环境中使用，或作为中小型企业的服务器使用。在版本 10.0.0 中，Adaptive Server Anywhere 更名为 SQL Anywhere 服务器，SQL Anywhere Studio 更名为 SQL Anywhere。

另请参见： “SQL Anywhere”一节第 394 页

包

Java 中相关类的集合。

被引用对象

一种对象（如表），该对象在另一个对象（如视图）的定义中被直接引用。

另请参见： “主键”一节第 404 页

编码

也称作字符编码，编码是一种方法，通过该方法可以将字符集中的每个字符映射到一个或多个字节的信息，这些信息通常以十六进制数字表示。编码的一个例子是 UTF-8。

另请参见：

- “字符集”一节第 404 页
- “代码页”一节第 379 页
- “归类”一节第 383 页

标识符

用于引用数据库对象（如表或列）的字符串。标识符可以包含 A 到 Z、a 到 z、0 到 9、下划线 (_)、at 符号 (@)、数字符号 (#) 或美元符号 ($) 中的任何字符。

并发

同时执行两个或更多个独立并且可能存在竞争关系的进程。SQL Anywhere 会自动使用锁定来隔离事务，并确保每个并发应用程序看到的数据集均一致。

另请参见：

- “事务”一节第 391 页
- “隔离级别”一节第 382 页

参考数据库

MobiLink 中一种用于 UltraLite 客户端开发的 SQL Anywhere 数据库。在开发过程中，可以将一个 SQL Anywhere 数据库同时作为参考数据库和统一数据库使用。通过其它产品建立的数据库无法用作参考数据库。

参照完整性

遵守数据一致性控制规则（具体而言，不同表中主键值与外键值之间的关系）。若要实现参照完整性，每个外键中的值必须与被引用表中行的主键值相符。

另请参见：

- “主键”一节第 404 页
- “外键”一节第 396 页

策略

QAnywhere 中指定应在何时进行消息传输的方式。

插件模块

Sybase Central 中一种用于访问和管理产品的方法。当您安装相应的产品时，插件通常会自动安装并注册 Sybase Central。通常，插件在 Sybase Central 主窗口中作为顶级容器出现，并且使用产品本身的名称，如 SQL Anywhere。

另请参见： “Sybase Central”一节第 395 页

查询

一条或一组 SQL 语句，用于访问和/或操作数据库中的数据。

另请参见： “SQL”一节第 394 页

冲突解决

在 MobiLink 中，冲突解决是指一种逻辑，它指定当两个用户修改不同远程数据库上同一行时的处理方法。

重定向器

一种 Web 服务器插件，用于为客户端与 MobiLink 服务器之间的请求和响应选择发送路径。此插件还实现了负荷平衡和故障转移机制。

抽取

SQL Remote 复制中从统一数据库卸载相应结构和数据的行为。此信息用于初始化远程数据库。

另请参见： “复制”一节第 381 页

触发器

一种特殊形式的存储过程，用户运行修改数据的查询时会自动执行该存储过程。

另请参见：

- “行级触发器”一节第 383 页
- “语句级触发器”一节第 401 页
- “完整性”一节第 397 页

传输规则

QAnywhere 中用于确定何时进行消息传输、传输哪些消息以及应在何时删除消息的逻辑。

窗口

作为分析功能执行对象的行组。一个窗口可以包含一行、多行或所有行的数据，这些数据已根据窗口定义中提供的分组规格进行了分区。窗口会进行移动，以包括为输入中的当前行执行计算所需的行数或行范围。窗口结构的主要优点是，不需要执行附加查询就可以有机会对结果进行分组和分析。

创建者 ID

UltraLite Palm OS 应用程序中一种在创建应用程序时指派的 ID。

存储过程

存储过程是数据库中存储的一组 SQL 指令，用于在数据库服务器上执行一组操作或查询。

代理表

一种本地表，它所包含的元数据可以像访问本地表一样访问远程数据库服务器上的表。

另请参见： “元数据”一节第 402 页

代理 ID

另请参见： “客户端消息存储库 ID”一节第 387 页

代码页

代码页是一种将字符集的字符映射到数字表示的编码，数字表示通常是 0 到 255 之间的一个整数。例如，Windows 代码页 1252 就是一个代码页。就本文档而言，代码页和编码这两个术语可以互换。

另请参见：

- “字符集”一节第 404 页
- “编码”一节第 377 页
- “归类”一节第 383 页

DBA 权限

使用户能够在数据库中执行管理活动的权限级别。DBA 用户在缺省情况下具有 DBA 权限。

另请参见： “数据库管理员 (DBA)”一节第 393 页

dbspace

用于创建更多数据存储空间的附加数据库文件。一个数据库可以包含在最多 13 个独立的文件（一个初始文件和 12 个 dbspace）中。每个表及其索引必须包含在单个数据库文件中。SQL 命令 CREATE DBSPACE 可将新文件添加到数据库中。

另请参见： “数据库文件”一节第 394 页

动态 SQL

执行前由程序以编程方式生成的 SQL。UltraLite 动态 SQL 是一种专用于小型设备的 SQL 变体。

对象树

Sybase Central 中数据库对象的层次。对象树的顶层显示您的 Sybase Central 版本所支持的全部产品。每种产品展开后会显示其自己的对象子树。

另请参见： “Sybase Central”一节第 395 页

EBF

快速错误修正软件。快速错误修正软件是含有一个或多个错误修正软件的软件子集。错误修正软件列在更新程序的发行说明中。错误修正软件更新可能只适用于具有相同版本号的已安装软件。已对该软件执行了一些测试，但该软件尚未进行完全测试。除非您自己已验证了软件的适用性，否则不要随应用程序分发这些文件。

发布

MobiLink 或 SQL Remote 中一种用于标识将要同步的数据的数据库对象。在 MobiLink 中，发布仅存在于客户端。一个发布包括多个项目。SQL Remote 用户可以通过预订发布来接收发布。MobiLink 用户可以通过创建发布的同步预订来同步发布。

另请参见：

- “复制”一节第 381 页
- “项目”一节第 399 页
- “发布更新”一节第 380 页

发布更新

SQL Remote 复制中对一个数据库中的一个或多个发布所做更改的列表。发布更新将作为复制消息的一部分定期发送到远程数据库。

另请参见：

- “复制”一节第 381 页
- “发布”一节第 380 页

发布者

SQL Remote 复制中数据库内可以与其它复制数据库交换复制消息的单个用户。

另请参见： “复制”一节第 381 页。

FILE

SQL Remote 复制中一种使用共享文件来交换复制消息的消息系统。它对测试以及在没有显式消息传送系统的情况下进行的安装很有用。

另请参见“复制”一节第 381 页。

分析树

查询的代数表示。

服务

在 Windows 操作系统上，服务是在运行应用程序的用户 ID 未登录时的应用程序运行方式。

服务器管理请求

一种 QAnywhere 消息，其格式设置为 XML 并发送到 QAnywhere 系统队列，作为一种管理服务器消息存储库或监控 QAnywhere 应用程序的方法。

服务器启动的同步

一种从 MobiLink 服务器启动 MobiLink 同步的方式。

服务器消息存储库

QAnywhere 中在消息传输到客户端消息存储库或 JMS 系统之前服务器上用于临时存储消息的关系数据库。消息通过服务器消息存储库在各客户端之间进行交换。

复制

在物理上不相同的数据库之间共享数据。Sybase 有三种复制技术：MobiLink、SQL Remote 和复制服务器。

复制代理

请参见： “LTM”一节第 388 页

复制服务器

Sybase 的一种基于连接的复制技术，用于与 SQL Anywhere 和 Adaptive Server Enterprise 一起使用。它专用于在一些数据库之间进行接近实时的复制。

另请参见： “LTM”一节第 388 页

复制频率

SQL Remote 复制中一项针对每个远程用户的设置，它决定发布者的消息代理向该远程用户发送复制消息的频率应为多少。

另请参见： “复制”一节第 381 页。

复制消息

SQL Remote 或复制服务器中一种在发布数据库与预订数据库之间发送的通信。消息包含复制系统所需的数据、直通语句及信息。

另请参见：

- “复制”一节第 381 页
- “发布更新”一节第 380 页

隔离级别

一个事务中的操作对其它并发事务中的操作的可见程度。隔离级别有四级，编号依次为 0 至 3。第 3 级提供最高级别的隔离。级别 0 为缺省设置。SQL Anywhere 还支持以下三个快照隔离级别：快照、语句快照和只读语句快照。

另请参见： “快照隔离”一节第 387 页

个人服务器

与客户端应用程序在同一台计算机上运行的数据库服务器。个人数据库服务器通常由单个用户在一台计算机上使用，但它可以支持来自该用户的几个并发连接。

工作表

一种内部存储区域，用于在查询优化过程中存储中间结果。

故障切换

在活动服务器、系统或网络出现故障或意外终止时切换到冗余或备用的服务器、系统或网络。故障转移会自动进行。

关系数据库管理系统 (RDBMS)

一种以相关表的形式存储数据的数据库管理系统。

另请参见： “数据库管理系统 (DBMS)”一节第 393 页

规范化

对数据库模式的改进，目的在于按照基于关系数据库理论的规则消除冗余并改善组织。

归类

定义数据库中文本属性的字符集与排序顺序的组合。对于 SQL Anywhere 数据库，缺省归类取决于运行服务器时所使用的操作系统和语言；例如，英语 Windows 系统上的缺省归类为 1252LATIN1。归类（也称作归类序列）用于对字符串进行比较和排序。

另请参见：

- “字符集”一节第 404 页
- “代码页”一节第 379 页
- “编码”一节第 377 页

行级触发器

每更改一行即执行一次的触发器。

另请参见：

- “触发器”一节第 379 页
- “语句级触发器”一节第 401 页

回退日志

对在每个未提交的事务执行过程中所做更改的记录。当收到 ROLLBACK 请求或者系统出现故障时，未提交的事务会从数据库中回退，将数据库返回其原先的状态。每个事务都有一个单独的回退日志，事务完成时日志会被删除。

另请参见： “事务”一节第 391 页

iAnywhere JDBC 驱动程序

iAnywhere JDBC 驱动程序提供了一个 JDBC 驱动程序，与纯 Java jConnect JDBC 驱动程序相比，该驱动程序拥有一些性能优势和功能优点，但它不是纯 Java 解决方案。建议在大多数情况下使用 iAnywhere JDBC 驱动程序。

另请参见：

- “JDBC”一节第 384 页
- “jConnect”一节第 384 页

InfoMaker

一种报告和数据维护工具，它用于创建复杂的表格、报告、图形、交叉表和表，并创建将这些报告用作构件块的应用程序。

Interactive SQL

一种 SQL Anywhere 应用程序，用于查询和更改数据库中的数据以及修改数据库的结构。Interactive SQL 不但提供了一个用于输入 SQL 语句的窗格，还提供了一些用于返回有关查询处理过程的信息和结果集的窗格。

JAR 文件

Java 档案文件。一种压缩的文件格式，由一个或多个用于 Java 应用程序的包的集合组成。它将安装和运行 Java 程序所需的全部资源都放在一个压缩文件中。

Java 类

Java 中的主要代码结构单元。它是组合在一起的过程和变量的集合，将过程和变量组合在一起的原因是它们都与某个特定的可识别类别有关。

jConnect

JavaSoft JDBC 标准的 Java 实现。它为 Java 开发人员提供多层和异类环境中的本地数据库访问。但在大多数情况下，iAnywhere JDBC 驱动程序是首选的 JDBC 驱动程序。

另请参见：

- “JDBC”一节第 384 页
- “iAnywhere JDBC 驱动程序”一节第 383 页

JDBC

Java 数据库连接。一种 SQL 语言编程接口，它允许 Java 应用程序访问关系数据。首选的 JDBC 驱动程序是 iAnywhere JDBC 驱动程序。

另请参见：

- “jConnect”一节第 384 页
- “iAnywhere JDBC 驱动程序”一节第 383 页

基表

永久性的数据表。有时为区别于临时表和视图，会将这种表称作**基表**。

另请参见：

- “临时表”一节第 387 页
- “视图”一节第 391 页

基于会话的同步

一种同步类型，这种同步会使数据表示在统一数据库和远程数据库都一致。MobiLink 基于会话。

基于脚本的上载

MobiLink 中一种将上载过程自定义为使用日志文件的替代方法的方式。

基于 SQL 的同步

MobiLink 中一种使用 MobiLink 事件将表数据与支持 MobiLink 的统一数据库进行同步的方式。对于基于 SQL 的同步，可以直接使用 SQL，也可以使用面向 Java 和 .NET 平台的 MobiLink 服务器 API 返回 SQL。

基于文件的下载

在 MobiLink 中同步数据的一种方式，其中下载以文件的方式进行分发，从而支持脱机分发同步更改。

集成登录

一种登录功能，它允许将同一个用户 ID 和口令用于操作系统登录、网络登录和数据库连接。

监听器

一个程序 (dblsn)，用于 MobiLink 服务器启动的同步。监听器安装在远程设备上，它们被配置为在接收到来自通告程序的信息时启动针对设备的操作。

另请参见： “服务器启动的同步”一节第 381 页

检查点

将对数据库的所有更改都保存到数据库文件中的时间点。在其它时间，所提交的更改仅保存到事务日志中。

检查约束

对列或列集强制实施指定条件的一种限制。

另请参见：

- “约束”一节第 403 页
- “外键约束”一节第 397 页
- “主键约束”一节第 404 页
- “唯一约束”一节第 398 页

脚本

MobiLink 中为处理 MobiLink 事件而编写的代码。脚本通过编程方式控制数据交换，以满足业务需要。

另请参见： “事件模型”一节第 391 页

脚本版本

MobiLink 中为创建同步而一起应用的一组同步脚本。

校验

测试数据库、表或索引是否受到特定类型的文件损坏。

校验和

随数据库页本身一起记录的计算出的数据库页位数。校验和能够确保数据库页写入磁盘时位数相符，因此数据库管理系统可以通过它来验证数据库页的完整性。如果计数相符，即认为数据库页已成功写入。

镜像日志

另请参见： “事务日志镜像”一节第 392 页

角色

概念性数据库建模中从一个角度描述某种关系的动词或短语。您可以用两个角色来描述每种关系。例如，"包含" 和 "隶属于" 便是角色。

角色名

外键的名称。由于它命名外表和主表之间的关系，因此称作角色名。缺省情况下，角色名就是表名，除非其它外键已经使用该名称（在这种情况下，缺省的角色名是表名后接一个三位的唯一数字）。也可以自己创建角色名。

另请参见： “外键”一节第 396 页

局部临时表

一种临时表，仅在复合语句执行期间或连接结束之前存在。当您只需要将数据集装载一次时，局部临时表非常有用。缺省情况下，行会在提交时被删除。

另请参见：

- “临时表”一节第 387 页
- “全局临时表”一节第 390 页

客户端/服务器

一种软件体系结构，在这种体系结构中，一个应用程序（客户端）从另一个应用程序（服务器）获取信息并向该应用程序发送信息。这两个应用程序常位于通过网络连接的不同计算机上。

客户端消息存储库

QAnywhere 中一种用于在远程设备上存储消息的 SQL Anywhere 数据库。

客户端消息存储库 ID

QAnywhere 中一种对客户端消息存储库进行唯一标识的 MobiLink 远程 ID。

快照隔离

一种为发出读请求的事务返回数据的已提交版本的隔离级别。SQL Anywhere 提供了以下三种快照隔离级别：快照、语句快照和只读语句快照。使用快照隔离时，读操作不会阻塞写操作。

另请参见： “隔离级别”一节第 382 页

连接

关系系统中的一种基本操作，它通过比较指定列中的值将两个或更多个表中的行链接在一起。

连接 ID

用于标识客户端应用程序与数据库之间给定连接的唯一编号。可以使用以下 SQL 语句来确定当前连接 ID：

```
SELECT CONNECTION_PROPERTY( 'Number' );
```

连接类型

SQL Anywhere 提供了四种类型的连接：交叉连接、键连接、自然连接和使用 ON 子句的连接。

另请参见： “连接”一节第 387 页

连接配置

连接到数据库所需的一组参数，如用户名、口令和服务器名称，它们在存储后即可方便地使用。

连接启动的同步

一种 MobiLink 服务器启动的同步，在这种同步下，连接发生变化时会启动同步。

另请参见： “服务器启动的同步”一节第 381 页

连接条件

一种影响连接结果的限制。您可以通过紧跟在连接语句的后面插入 ON 子句或 WHERE 子句来指定连接条件。对于自然连接和关键连接，SQL Anywhere 会生成连接条件。

另请参见：

- “连接”一节第 387 页
- “生成的连接条件”一节第 392 页

临时表

为临时存储数据而创建的表。有两种类型：全局临时表和局部临时表。

另请参见：

- “局部临时表”一节第 386 页
- “全局临时表”一节第 390 页

LTM

日志传送管理器（Log Transfer Manager，简称 LTM）也称作复制代理。LTM 是一个与 Replication Server 一起使用的程序，它读取数据库事务日志并将提交的更改发送到 Sybase 复制服务器。

请参见： “复制服务器”一节第 382 页

轮询

在 MobiLink 服务器启动的同步中，轻量级轮询器（例如 MobiLink 监听器）从通告程序请求推式通知的方式。

另请参见： “服务器启动的同步”一节第 381 页

逻辑索引

指向物理索引的引用（指针）。磁盘上不存储逻辑索引的索引结构。

命令文件

包含 SQL 语句的文本文件。命令文件可以手工建立，也可以通过数据库实用程序自动建立。例如，dbunload 实用程序会创建一个命令文件，其中包含重新创建给定数据库所需的 SQL 语句。

MobiLink

一种基于会话的同步技术，其设计用途是将 UltraLite 和 SQL Anywhere 远程数据库与统一数据库同步。

另请参见：

- “统一数据库”一节第 395 页
- “同步”一节第 395 页
- “UltraLite”一节第 396 页

MobiLink 服务器

运行 MobiLink 同步的计算机程序，即 mlsrv11。

MobiLink 监控器

一种用于监控 MobiLink 同步的图形化工具。

MobiLink 客户端

有两种 MobiLink 客户端。对于 SQL Anywhere 远程数据库，MobiLink 客户端是 dbmlsync 命令行实用程序。对于 UltraLite 远程数据库，MobiLink 客户端内置于 UltraLite 运行时库中。

MobiLink 系统表

MobiLink 同步所需的系统表。它们由 MobiLink 安装程序脚本安装到 MobiLink 统一数据库中。

MobiLink 用户

MobiLink 用户用于与 MobiLink 服务器进行连接。在远程数据库上创建 MobiLink 用户，然后在统一数据库中注册该用户。MobiLink 用户名完全独立于数据库用户名。

模式

数据库的结构，其中包括表、列和索引以及它们之间的关系。

内连接

一种连接，在这种连接中，仅当两个表都满足连接条件时行才会出现在结果集中。内连接是缺省设置。

另请参见：

- “连接”一节第 387 页
- “外连接”一节第 397 页

ODBC

开放式数据库连接。一种用于与数据库管理系统连接的标准 Windows 接口。ODBC 是 SQL Anywhere 所支持的几种接口之一。

ODBC 管理器

一种随 Windows 操作系统提供的 Microsoft 程序，用于设置 ODBC 数据源。

ODBC 数据源

用户要通过 ODBC 访问的数据的规范以及获取该数据时所需的信息。

PDB

Palm 数据库文件。

PowerDesigner

一种数据库建模应用程序。PowerDesigner 为设计数据库或数据仓库提供了结构化的方法。SQL Anywhere 包括 PowerDesigner 的 Physical Data Model 组件。

PowerJ

一种 Sybase 产品，用于开发 Java 应用程序。

QAnywhere

应用程序到应用程序的消息传递（包括移动设备到移动设备和移动设备与企业之间的消息传递），它使在移动或无线设备上运行的自定义程序能够与处在中央位置的服务器应用程序进行通信。

QAnywhere 代理

QAnywhere 中一种运行在客户端设备上的进程，用于监控客户端消息存储库和确定应在何时传输消息。

嵌入式 SQL

一种 C 语言程序编程接口。SQL Anywhere 嵌入式 SQL 是 ANSI 和 IBM 标准的实现。

轻量级轮询器

在 MobiLink 服务器启动的同步中，轮询来自 MobiLink 服务器的推式通知的设备应用程序。

另请参见： “服务器启动的同步”一节第 381 页

全局临时表

一种临时表，在被显式地删除之前，其数据定义对所有用户都可见。全局临时表允许用户各自打开一个表的相同实例。缺省情况下，行在提交时被删除，并且始终是在连接结束时被删除。

另请参见：

- “临时表”一节第 387 页
- “局部临时表”一节第 386 页

日志文件

SQL Anywhere 所维护的事务日志。该日志文件用于确保在出现系统或介质故障时可以恢复数据库、提高数据库性能以及使用 SQL Remote 实现数据复制。

另请参见：

- “事务日志”一节第 391 页
- “事务日志镜像”一节第 392 页
- “完全备份”一节第 397 页

散列

散列是一种将索引条目转化为键的索引优化。索引散列旨在通过将足够的行实际数据与其行 ID 包括在一起，以避免进行先查找行、后装载行然后再将行解出才能得出索引值的高开销操作。

上载

同步过程的一个阶段，在此阶段数据从远程数据库传送到统一数据库。

设备跟踪

在 MobiLink 服务器启动的同步中，允许使用标识设备的 MobiLink 用户名来对消息进行寻址的功能。

另请参见： “服务器启动的同步”一节第 381 页

实例化视图

实例化视图是指已计算并已存储在磁盘上的视图。实例化视图同时具有视图的特征（使用查询说明进行定义）和表的特征（可以对其执行大多数表操作）。

另请参见：

- “基表”一节第 384 页
- “视图”一节第 391 页

世代号

MobiLink 中的一种机制，用于强制远程数据库先上载数据，然后再应用任何其它下载文件。

另请参见： “基于文件的下载”一节第 385 页

事件模型

MobiLink 中组成同步的事件（如 begin_synchronization 和 download_cursor）序列。如果为事件创建了脚本，则会调用事件。

视图

一种作为对象存储在数据库中的 SELECT 语句。它使用户能够看到一个或多个表中的行子集或列子集。每当用户使用特定表或表组合的视图时，都将利用存储在这些表中的信息重新计算视图。视图对确保安全以及定制数据库信息的外观来使数据访问简单明了有帮助。

事务

组成一个逻辑工作单元的 SQL 语句序列。事务要么全部得到处理，要么根本不做处理。SQL Anywhere 支持事务处理，并内置了锁定功能，使并发事务能够访问数据库而又不损坏数据。事务要么以 COMMIT 语句结束，该语句使对数据的更改成为永久性更改；要么以 ROLLBACK 语句结束，该语句撤消在事务执行过程中所做的全部更改。

事务日志

一种按进行更改的顺序存储对数据库所做全部更改的文件。它会提高性能并支持在数据库文件损坏时恢复数据。

事务日志镜像

同时维护的事务日志文件的完全相同副本（可选）。每当数据库更改写入事务日志文件时，也会同时写入事务日志镜像文件。

镜像文件应该与事务日志保留在不同的设备上，这样在任意设备出现故障时，日志的其它副本会确保数据可以安全地恢复。

另请参见： “事务日志”一节第 391 页

事务完整性

MobiLink 中对整个同步系统事务的有保证维护。要么同步整个事务，要么不对事务的任何部分进行同步。

生成的连接条件

一种自动生成的对连接结果的限制。有两种类型：关键和自然。指定 KEY JOIN 或指定关键字 JOIN 但不使用关键字 CROSS、NATURAL 或 ON 时，会生成关键连接。对于关键连接，所生成的连接条件取决于表之间的外键关系。指定 NATURAL JOIN 时会生成自然连接；所生成的连接条件基于两个表中的公用列名。

另请参见：

- “连接”一节第 387 页
- “连接条件”一节第 387 页

受保护的功能

数据库服务器启动时由 -sf 选项指定的功能，该数据库服务器上运行的任何数据库都无法使用该功能。

授权选项

一种权限级别，它允许用户向其他用户授予权限。

数据操作语言 (DML)

用于操作数据库中数据的 SQL 语句子集。DML 语句可以检索、插入、更新和删除数据库中的数据。

数据定义语言 (DDL)

用于定义数据库中数据结构的 SQL 语句子集。DDL 语句可以创建、修改和删除数据库对象（如表和用户）。

数据类型

数据的格式，如 CHAR 或 NUMERIC。在 ANSI SQL 标准中，数据类型也可以包括对大小、字符集和归类的限制。

另请参见： “域”一节第 401 页

数据立方体

一种多维结果集，每一维都以不同的方式对相同的结果进行分组和排序。数据立方体提供了有关数据的综合性信息，如果不使用数据立方体，要获得同样的信息就必须进行自连接查询和相关子查询。数据立方体是 OLAP 功能的一部分。

数据库

通过主键和外键关联的表的集合。表包含数据库中的信息。表和键一起定义数据库的结构。数据库管理系统会访问此信息。

另请参见：

- “外键”一节第 396 页
- “主键”一节第 404 页
- “数据库管理系统 (DBMS)”一节第 393 页
- “关系数据库管理系统 (RDBMS)”一节第 382 页

数据库对象

包含或接收信息的数据库组件。表、索引、视图、过程和触发器便是数据库对象。

数据库服务器

对所有针对数据库信息的访问进行管理的计算机程序。SQL Anywhere 提供了两种类型的服务器：网络服务器和个人服务器。

数据库管理系统 (DBMS)

用于创建和使用数据库的程序的集合。

另请参见： “关系数据库管理系统 (RDBMS)”一节第 382 页

数据库管理员 (DBA)

具有维护数据库所需权限的用户。DBA 通常负责对数据库模式的所有更改以及管理用户和组。数据库管理员角色自动内置于数据库中，其用户 ID 为 DBA，口令是 sql。

数据库连接

客户端应用程序与数据库之间的通信渠道。必须具有有效的用户 ID 和口令才能建立连接。为用户 ID 授予的特权决定了在连接过程中可以执行的操作。

数据库名称

服务器装载数据库时为数据库指定的名称。缺省数据库名是初始数据库文件的文件名（不含扩展名）。

另请参见： “数据库文件”一节第 394 页

数据库所有者 (dbo)

一种特殊的用户，他拥有不归 SYS 所有的系统对象。

另请参见：

- “数据库管理员 (DBA)”一节第 393 页
- “SYS”一节第 395 页

数据库文件

数据库保存在一个或多个数据库文件中。其中一个是初始文件，后面的文件称作 dbspace。每个表（包括其索引）都必须包含在单个数据库文件中。

另请参见： “dbspace”一节第 380 页

死锁

一组事务会进入的一种特殊状态，在该状态下这些事务都不能继续执行。

SQL

用于与关系数据库进行通信的语言。ANSI 定义了 SQL 的标准，其最新标准是 SQL-2003。SQL 的非官方全称是结构化查询语言。

SQL Anywhere

SQL Anywhere 的关系数据库服务器组件，专供在移动和嵌入式环境中使用，或作为中小型企业的服务器使用。SQL Anywhere 也是包含 SQL Anywhere RDBMS、UltraLite RDBMS、MobiLink 同步软件和其它组件的软件包的名称。

SQL Remote

一种基于消息的数据复制技术，用于在统一数据库与远程数据库之间进行双向复制。统一数据库和远程数据库必须是 SQL Anywhere。

SQL 语句

包含用于将指令传递给 DBMS 的 SQL 关键字的字符串。

另请参见：

- “模式”一节第 389 页
- “SQL”一节第 394 页
- “数据库管理系统 (DBMS)”一节第 393 页

锁定

一种在同时执行多个事务的过程中保护数据完整性的并发控制机制。SQL Anywhere 会自动应用锁以防止两个连接同时更改同一数据，并防止其它连接读取正接受更改的数据。

您可以通过设置隔离级别来控制锁定。

另请参见：

- “隔离级别”一节第 382 页
- “并发”一节第 377 页
- “完整性”一节第 397 页

索引

一组已排序的、与基表中的一个或多个列关联的键和指针。在表中一个或多个列上设置索引可以提高性能。

Sybase Central

一种数据库管理工具，通过图形用户界面提供 SQL Anywhere 数据库设置、属性和实用程序。Sybase Central 也可用于管理其它 Sybase 产品，其中包括 MobiLink。

SYS

一种拥有大多数系统对象的特殊用户。无法以 SYS 身份登录。

统一数据库

在分布式数据库环境中，是指用于存储数据主副本的数据库。出现冲突或差异时，将把统一数据库视为具有数据的主副本。

另请参见：

- “同步”一节第 395 页
- “复制”一节第 381 页

通信流

MobiLink 中 MobiLink 客户端与 MobiLink 服务器之间进行通信时所使用的网络协议。

通告程序

一种由 MobiLink 服务器启动的同步使用的程序。通告程序集成在 MobiLink 服务器中。它们会检查统一数据库是否有推式请求，并发送推式通知。

另请参见：

- “服务器启动的同步”一节第 381 页
- “监听器”一节第 385 页

同步

利用 MobiLink 技术在数据库之间复制数据的过程。

在 SQL Remote 中，同步专指以初始数据集初始化远程数据库的过程。

另请参见：

- “MobiLink”一节第 388 页
- “SQL Remote”一节第 394 页

推式请求

在 MobiLink 服务器启动的同步中，通告程序通过检查它来确定推式通知是否需要发送到设备的结果集中的一行值。

另请参见： “服务器启动的同步”一节第 381 页

推式通知

QAnywhere 中一种从服务器传送到 QAnywhere 客户端的特殊消息，用于提示客户端启动消息传输。在 MobiLink 服务器启动的同步中，从通告程序传送到包含推式请求数据和内部信息的设备的特殊消息。

另请参见：

- “QAnywhere”一节第 390 页
- “服务器启动的同步”一节第 381 页

UltraLite

一种针对小型设备、移动设备和嵌入式设备进行了优化的数据库。所面向的平台包括手机、传呼机和个人记事本。

UltraLite 运行时

一种过程中关系数据库管理系统，其中包括一个内置 MobiLink 同步客户端。每个 UltraLite 编程接口使用的库以及 UltraLite 引擎中都包括 UltraLite 运行时。

外表

包含外键的表。

另请参见： “外键”一节第 396 页

外部登录

与远程服务器通信时使用的替代登录名和口令。缺省情况下，SQL Anywhere 每次代表其客户端连接到远程服务器时都会使用这些客户端的名称和口令。但是，您可以通过创建外部登录来替换这一缺省设置。外部登录是指与远程服务器通信时使用的替代登录名和口令。

外键

一个表中复制另一个表中主键值的一个或多个列。外键建立表间的关系。

另请参见：

- “主键”一节第 404 页
- “外表”一节第 396 页

外键约束

对单个列或一组列的限制，指定表中的数据与某个其它表中数据的关系。对列集施加外键约束可使这些列成为外键。

另请参见：

- “约束”一节第 403 页
- “检查约束”一节第 385 页
- “主键约束”一节第 404 页
- “唯一约束”一节第 398 页

外连接

一种保留表中所有行的连接。SQL Anywhere 支持左、右和完全外连接。左外连接保留表中位于连接运算符左侧的行，当右表中的行不满足连接条件时，它将返回空值。完全外连接保留两个表中的所有行。

另请参见：

- “连接”一节第 387 页
- “内连接”一节第 389 页

完全备份

对整个数据库和事务日志（可选）的备份。完全备份包含数据库中的所有信息，因此可以在系统或介质出现故障时提供保护。

另请参见： “增量备份”一节第 403 页

完整性

遵守完整性规则的情况，完整性规则确保数据正确并准确，而且数据库的关系结构保持不变。

另请参见： “参照完整性”一节第 378 页

网关

一种 MobiLink 对象，存储在 MobiLink 系统表或通告程序属性文件中，包含有关如何发送用于服务器启动同步的消息的信息。

另请参见： “服务器启动的同步”一节第 381 页

网络服务器

从共享公共网络的计算机接受连接的数据库服务器。

另请参见： “个人服务器”一节第 382 页

网络协议

通信类型，如 TCP/IP 或 HTTP。

维护版本

维护版本是一套完整的软件，它升级已安装的具有相同主版本号的较早版本的软件（版本号格式是 *major.minor.patch.build*）。升级程序的发行说明中列出了错误修正软件和其它更改。

唯一约束

对某个列或一组列的限制，它要求所有非空值都各不相同。一个表可以有多个唯一约束。

另请参见：

- “外键约束”一节第 397 页
- “主键约束”一节第 404 页
- “约束”一节第 403 页

谓语

一种条件表达式，可以选择性地将其与逻辑运算符 AND 和 OR 组合在一起，以组成 WHERE 或 HAVING 子句中的条件集。在 SQL 中，求值结果为 UNKNOWN 的谓语将解释为 FALSE。

位数组

位数组是一种用于有效率地存储位序列的数组数据结构。位数组与字符串类似，不同的是其各个部分由 0（零）和 1（一）而不是字符组成。位数组通常用于保存一串布尔值。

Windows

Microsoft Windows 操作系统系列，如 Windows Vista、Windows XP 和 Windows 200x。

Windows CE

请参见“Windows Mobile”一节第 398 页。

Windows Mobile

Microsoft 为移动设备制造的操作系统系列。

文件定义数据库

MobiLink 中一种用于创建下载文件的 SQL Anywhere 数据库。

另请参见： “基于文件的下载”一节第 385 页

物理索引

索引存储在磁盘上的实际索引结构。

系统表

一种表，由 SYS 或 dbo 拥有，用于保存元数据。系统表也称作数据字典表，由数据库服务器创建并维护。

系统对象

由 SYS 或 dbo 拥有的数据库对象。

系统视图

存在于每一个数据库中的一种视图，它以易于理解的格式表示系统表中包含的信息。

下载

同步过程的一个阶段，在此阶段数据从统一数据库传送到远程数据库。

相关名

查询的 FROM 子句中使用的表或视图的名称—要么是表或视图的原始名称，要么是在 FROM 子句中定义的替代名称。

项目

在 MobiLink 或 SQL Remote 中，项目是表示整个表或表中行和列子集的数据库对象。项目在发布中组合在一起。

另请参见：

- “复制”一节第 381 页
- “发布”一节第 380 页

消息存储库

QAnywhere 中客户端和服务器设备上存储消息的数据库。

另请参见：

- “客户端消息存储库”一节第 386 页
- “服务器消息存储库”一节第 381 页

消息类型

SQL Remote 复制中指定远程用户与统一数据库发布者通信方式的数据库对象。一个统一数据库可能定义了几种消息类型，这样一来，不同的远程用户就可以使用不同的消息系统与统一数据库进行通信。

另请参见：

- “复制”一节第 381 页
- “统一数据库”一节第 395 页

消息日志

可存储来自数据库服务器或 MobiLink 服务器等应用程序的消息的日志。此类信息还可以出现在消息窗口中或记录到文件中。消息日志包括信息性消息、错误、警告以及来自 MESSAGE 语句的消息。

消息系统

SQL Remote 复制中用于在统一数据库与远程数据库之间交换消息的协议。SQL Anywhere 包括对以下消息系统的支持：FILE、FTP 和 SMTP。

另请参见：

- “复制”一节第 381 页
- “FILE”一节第 381 页

卸载

卸载数据库时会将数据库的结构和/或数据导出到文本文件（如果是结构，则导出到 SQL 命令文件中；如果是数据，则导出到 ASCII 逗号分隔文件中）。使用卸载实用程序来卸载数据库。

此外，您也可以使用 UNLOAD 语句卸载数据的选定部分。

性能统计

反映数据库系统性能的值。例如，CURRREAD 统计表示数据库服务器已发出但尚未完成的文件读取次数。

业务规则

基于实际要求的准则。通常，业务规则通过检查约束、用户定义数据类型以及事务的正确使用来实现。

另请参见：

- “约束”一节第 403 页
- “用户定义数据类型”一节第 401 页

引用对象

一种对象（如视图），其定义直接引用数据库中的另一个对象（如表）。

另请参见： “外键”一节第 396 页

用户定义数据类型

请参见“域”一节第 401 页。

游标

指向结果集的已命名链接，用于通过编程接口访问和更新行。在 SQL Anywhere 中，游标支持在查询结果中进行向前和向后移动。游标由两部分组成：游标结果集（通常由 SELECT 语句定义）和游标位置。

另请参见：

- “游标结果集”一节第 401 页
- “游标位置”一节第 401 页

游标结果集

与游标关联的查询所得到的行集。

另请参见：

- “游标”一节第 401 页
- “游标位置”一节第 401 页

游标位置

指向游标结果集中一个行的指针。

另请参见：

- “游标”一节第 401 页
- “游标结果集”一节第 401 页

语句级触发器

在整个触发语句完成后执行的触发器。

另请参见：

- “触发器”一节第 379 页
- “行级触发器”一节第 383 页

域

内置数据类型的别名，其中包括适用的精度值和小数位值，还可以选择是否包括 DEFAULT 值和 CHECK 条件。SQL Anywhere 中预定义了一些域，如货币数据类型。也称作用户定义数据类型。

另请参见：“数据类型”一节第 392 页

预订

MobiLink 同步中发布与 MobiLink 用户之间的客户端数据库中的一个链接，它使发布所描述的数据能够得到同步。

SQL Remote 复制中发布与远程用户之间的一种链接，它使用户能够与统一数据库交换该发布上的更新。

另请参见：

- “发布”一节第 380 页
- “MobiLink 用户”一节第 389 页

元数据

数据的数据。元数据描述其它数据的性质和内容。

另请参见： “模式”一节第 389 页

原子事务

保证成功完成或保证根本不予完成的事务。如果错误使原子事务的一部分无法完成，则将回退事务以防止数据库处于不一致的状态。

REMOTE DBA 特权

在 SQL Remote 中，消息代理 (dbremote) 所需的权限级别。MobiLink 中 SQL Anywhere 同步客户端 (dbmlsync) 所需的权限级别。当消息代理或同步客户端作为具有该权限的用户建立连接时，它将具有完全的 DBA 访问权。如果不是通过消息代理或同步客户端进行连接，则该用户 ID 将不具有附加权限。

另请参见： “DBA 权限”一节第 380 页

远程 ID

SQL Anywhere 和 UltraLite 数据库中一种由 MobiLink 使用的唯一标识符。远程 ID 初始情况下设置为 NULL，在数据库第一次同步期间将设置为 GUID。

远程数据库

MobiLink 或 SQL Remote 中一种与统一数据库交换数据的数据库。远程数据库可以共享统一数据库中的全部或部分数据。

另请参见：

- “同步”一节第 395 页
- “统一数据库”一节第 395 页

约束

对特定数据库对象（如表或列）中所包含值的限制。例如，列可以具有唯一性约束，该约束要求该列中的所有值互不相同。表可以具有外键约束，该约束指定该表中的信息与某个其它表中数据的关系。

另请参见：

- “检查约束”一节第 385 页
- “外键约束”一节第 397 页
- “主键约束”一节第 404 页
- “唯一约束”一节第 398 页

运营公司

一种 MobiLink 对象，存储在 MobiLink 系统表或通告程序属性文件中，包含有关供服务器启动的同步使用的公共运营公司的信息。

另请参见： “服务器启动的同步”一节第 381 页

增量备份

仅包含事务日志的备份，通常在两次完全备份之间使用。

另请参见： “事务日志”一节第 391 页

争用

为获取资源而竞争的行为。例如，就数据库而言，如果有两个或更多个用户试图编辑数据库的同一行，就会为获得编辑该行的权利而发生争用。

正则表达式

正则表达式是字符、通配符和运算符的序列，用于定义某种模式以在字符串内进行搜索。

直方图

直方图是列统计信息最重要的组成部分，是一种表示数据分布的方式。SQL Anywhere 维护直方图以为优化程序提供有关列值分布情况的统计信息。

直接行处理

MobiLink 中一种用于将表数据同步到 MobiLink 支持的统一数据库以外的数据源的方法。使用直接行处理时，上载和下载都可以实现。

另请参见：

- “统一数据库”一节第 395 页
- “基于 SQL 的同步”一节第 385 页

主表

包含外键关系中的主键的表。

主键

其值唯一标识表中各行的一个列或多个列。

另请参见： “外键”一节第 396 页

主键约束

一种对主键列的唯一性约束。一个表只能有一个主键约束。

另请参见：

- “约束”一节第 403 页
- “检查约束”一节第 385 页
- “外键约束”一节第 397 页
- “唯一约束”一节第 398 页
- “完整性”一节第 397 页

子查询

嵌套在 SELECT、INSERT、UPDATE 或 DELETE 语句或者其它子查询中的 SELECT 语句。

有两种类型的子查询：相关子查询和嵌套子查询。

字符串

字符串是以单引号围起的字符序列。

字符集

字符集是一组符号，包括字母、数字、空格和其它符号。字符集的一个例子是 ISO-8859-1，又称作 Latin1。

另请参见：

- “代码页”一节第 379 页
- “编码”一节第 377 页
- “归类”一节第 383 页

索引

其它

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Y

Z

