SYBASE iAnywhere

MobiLink
服务器管理

2009 年 2 月

11.0.1 版

版权和商标

版权所有 © 2009 iAnywhere Solutions, Inc. 部分版权所有 © 2009 Sybase, Inc. 保留所有权利。

本文档按原样提供，并不做任何形式的担保或承担任何责任（除非在您与 iAnywhere 达成的书面协议中另行规定）。

对本文档（全部或部分）的使用、打印、复制和分发须符合下列条件：1) 必须在整个或部分文档的所有副本中保留此声明和所有其它所有权声明，2) 不得修改本文档，3) 不得以任何形式表明您或 iAnywhere 之外的任何人是本文档的作者或提供者。

iAnywhere®、Sybase® 以及在 http://www.sybase.com/detail?id=1011207 上所列出的商标均为 Sybase, Inc. 或其子公司的商标。® 表示在美国注册。

文中提及的所有其它公司和产品名可能是与其相关的各个公司的商标。

目录

关于本手册 xiii

关于 SQL Anywhere 文档 xiv

使用 MobiLink 服务器技术 1

MobiLink 统一数据库 3
统一数据库简介 4
建立统一数据库 6
RDBMS 相关同步脚本 8
Adaptive Server Enterprise 统一数据库 10
IBM DB2 LUW 统一数据库 12
IBM DB2 主机统一数据库 15
Microsoft SQL Server 统一数据库 19
MySQL 统一数据库 21
Oracle 统一数据库 23
SQL Anywhere 统一数据库 26
MobiLink 服务器 27
运行 MobiLink 服务器 28
停止 MobiLink 服务器 30
记录 MobiLink 服务器操作 31
在当前会话外运行 MobiLink 服务器 33
在服务器群中运行 MobiLink 服务器 37
MobiLink 服务器启动故障排除 38
MobiLink 服务器选项 39
mlsrv11 语法 41
@data 选项 46
-a 选项 47
-b 选项 48
-bn 选项 49
-c 选项 50
-cm 选项 51
-cn 选项 52
-cr 选项 53

-cs 选项 54
-ct 选项 55
-dl 选项 56
-dr 选项 57
-ds 选项 58
-dsd 选项 59
-dt 选项 60
-e 选项 61
-esu 选项 62
-et 选项 63
-f 选项 64
-fips 选项 65
-fr 选项 66
-ftr 选项 67
-lsc 选项 68
-m 选项 69
-nba 选项 70
-nc 选项 71
-notifier 选项 72
-o 选项 73
-on 选项 74
-oq 选项 75
-os 选项 76
-ot 选项 77
-ppv 选项 78
-q 选项 82
-r 选项 83
-rd 选项 84
-s 选项 85
-sl dnet 选项 86
-sl java 选项 87
-sm 选项 89
-ss 选项 90
-tc 选项 91
-tf 选项 92

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

-tx 选项 93
-ud 选项 94
-ui 选项 95
-ux 选项 96
-v 选项 97
-w 选项 100
-wu 选项 101
-x 选项 102
-xo 选项 107
-zp 选项 111
-zs 选项 112
-zt 选项 113
-zu 选项 114
-zus 选项 115
-zw 选项 116
-zwd 选项 117
-zwe 选项 118
同步技术 119
MobiLink 开发提示 120
基于时间戳的下载 121
快照同步 125
在远程数据库之间对行进行分区 127
仅上载同步和仅下载同步 130
维护唯一主键 131
冲突处理 137
强制冲突 145
数据输入 146
处理删除 147
处理失败的下载 149
下载确认 152
从存储过程调用中下载结果集 153
从自引用表上载数据 155
MobiLink 隔离级别 156
MobiLink 性能 159
性能提示 160

影响 MobiLink 性能的关键因素 163
监控 MobiLink 性能 166
MobiLink 监控器 167
MobiLink 监控器简介 168
启动 MobiLink 监控器 169
使用 MobiLink 监控器 171
保存 MobiLink 监控器数据 179
自定义您的统计信息 180
MobiLink 统计信息属性 181
用于 MobiLink 的 SQL Anywhere 监控器 185
SQL Anywhere 监控器简介 186
监控器快速入门 189
教程：使用监控器 190
启动监控器 195
退出监控器 196
连接到监控器 197
断开与监控器的连接 198
监控资源 199
管理资源 206
使用监控器用户 212
警告 216
在一台单独的计算机上安装 SQL Anywhere 监控器 220
监控器疑难解答 221
中继服务器 223
中继服务器简介 224
中继服务器配置文件 227
出站启动器 231
中继服务器状态管理器 234
部署中继服务器 236
更新中继服务器群配置 241
Sybase 中继服务器托管服务 243
将 MobiLink 与中继服务器结合使用 245
重定向器（不建议使用） 249
重定向器（不建议使用）简介 250
设置重定向器 252
为重定向器配置 MobiLink 客户端和服务器 253

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

配置重定向器属性 255
Windows 上 Netscape/Sun Web 服务器的 NSAPI 重定向器（不建议使用） 261
Unix 上 Netscape/Sun Web 服务器的 NSAPI 重定向器（不建议使用） 263
用于 Microsoft Web 服务器的 ISAPI 重定向器（不建议使用） 265
Servlet 重定向器（不建议使用） 267
Apache 重定向器（不建议使用） 269
M-Business Anywhere 重定向器（不建议使用） 271
MobiLink 基于文件的下载 275
基于文件的下载简介 276
建立基于文件的下载 277
校验检查 280
基于文件的下载示例 283

MobiLink 事件 291

编写同步脚本 293
同步脚本介绍 294
脚本和同步过程 297
脚本类型 298
脚本参数 299
脚本版本 303
必需的脚本 305
添加和删除脚本 306
编写用于上载行的脚本 309
编写用于下载行的脚本 312
编写用于处理错误的脚本 317
同步事件 319
MobiLink 事件概述 321
authenticate_file_transfer 连接事件 331
authenticate_parameters 连接事件 332
authenticate_user 连接事件 335
authenticate_user_hashed 连接事件 339
begin_connection 连接事件 343
begin_connection_autocommit 连接事件 344
begin_download 连接事件 345
begin_download 表事件 347

begin_download_deletes 表事件 349
begin_download_rows 表事件 352
begin_publication 连接事件 355
begin_synchronization 连接事件 358
begin_synchronization 表事件 360
begin_upload 连接事件 362
begin_upload 表事件 364
begin_upload_deletes 表事件 366
begin_upload_rows 表事件 368
download_cursor 表事件 370
download_delete_cursor 表事件 373
download_statistics 连接事件 376
download_statistics 表事件 379
end_connection 连接事件 382
end_download 连接事件 384
end_download 表事件 386
end_download_deletes 表事件 388
end_download_rows 表事件 391
end_publication 连接事件 394
end_synchronization 连接事件 397
end_synchronization 表事件 399
end_upload 连接事件 401
end_upload 表事件 403
end_upload_deletes 表事件 406
end_upload_rows 表事件 408
handle_DownloadData 连接事件 410
handle_error 连接事件 414
handle_odbc_error 连接事件 418
handle_UploadData 连接事件 422
modify_error_message 连接事件 428
modify_last_download_timestamp 连接事件 430
modify_next_last_download_timestamp 连接事件 433
modify_user 连接事件 436
nonblocking_download_ack 连接事件 438
prepare_for_download 连接事件 440

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

publication_nonblocking_download_ack 连接事件 442
report_error 连接事件 444
report_odbc_error 连接事件 447
resolve_conflict 表事件 450
synchronization_statistics 连接事件 453
synchronization_statistics 表事件 456
time_statistics 连接事件 459
time_statistics 表事件 462
upload_delete 表事件 465
upload_fetch 表事件 467
upload_fetch_column_conflict 表事件 469
upload_insert 表事件 471
upload_new_row_insert 表事件 473
upload_old_row_insert 表事件 476
upload_statistics 连接事件 479
upload_statistics 表事件 483
upload_update 表事件 488

MobiLink 服务器 API 491

使用 Java 语言编写同步脚本 493
Java 同步逻辑简介 494
设置 Java 同步逻辑 495
编写 Java 同步逻辑 497
Java 同步示例 504
用于 Java 的 MobiLink 服务器 API 参考 508
使用 .NET 编写同步脚本 551
.NET 同步逻辑简介 552
设置 .NET 同步逻辑 553
编写 .NET 同步逻辑 555
.NET 同步技术 562
装载共享程序集 563
.NET 同步示例 565
用于 .NET 参考的 MobiLink 服务器 API 567
直接行处理 609
直接行处理简介 610

处理直接上载 614
处理直接下载 620

MobiLink 参考 621

MobiLink 服务器系统过程 623
MobiLink 系统过程 624
MobiLink 实用程序 647
MobiLink 实用程序简介 648
MobiLink 停止实用程序 (mlstop) 649
MobiLink 用户验证实用程序 (mluser) 650
MobiLink 服务器系统表 653
MobiLink 系统表简介 655
IBM DB2 主机系统表名称转换 656
ml_active_remote_id 657
ml_column 658
ml_connection_script 659
ml_database 660
ml_device 661
ml_device_address 662
ml_listening 663
ml_passthrough 665
ml_passthrough_repair 666
ml_passthrough_script 667
ml_passthrough_status 669
ml_property 670
ml_qa_clients 671
ml_qa_delivery 672
ml_qa_delivery_archive 673
ml_qa_global_props 674
ml_qa_notifications 675
ml_qa_repository 676
ml_qa_repository_archive 677
ml_qa_repository_props 678
ml_qa_repository_props_archive 679
ml_qa_repository_staging 680

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

ml_qa_status_history 681
ml_qa_status_history_archive 682
ml_qa_status_staging 683
ml_script 684
ml_script_version 685
ml_scripts_modified 686
ml_server 687
ml_sis_sync_state 688
ml_subscription 689
ml_table 691
ml_table_script 692
ml_user 693
远程数据库和统一数据库之间的 MobiLink 数据映射 695
Adaptive Server Enterprise 数据映射 696
IBM DB2 LUW 数据映射 704
IBM DB2 主机数据映射 711
Microsoft SQL Server 数据映射 720
MySQL 数据映射 726
Oracle 数据映射 731
字符集注意事项 739
字符集注意事项 740
用于 MobiLink 的 iAnywhere Solutions ODBC 驱动程序 743
MobiLink 支持的 ODBC 驱动程序 744
iAnywhere Solutions Oracle 驱动程序 745
部署 MobiLink 应用程序 749
MobiLink 部署简介 750
部署 MobiLink 服务器 751
部署 SQL Anywhere MobiLink 客户端 763
部署 UltraLite MobiLink 客户端 765
部署 QAnywhere 应用程序 766

术语表 771

术语表 773

索引 801

关于本手册

主题

本手册将介绍如何设置和管理 MobiLink 服务器、统一数据库和 MobiLink 应用程序。它还将介绍用于 MobiLink 的 SQL Anywhere 监控器。这是一个基于 Web 浏览器的管理工具，它提供关于 MobiLink 服务器和中继服务器健康状况和可用性方面的信息，从而使移动设备与 MobiLink 之间、Afaria 与 iAnywhere Mobile Office 服务器之间能够通过 Web 服务器安全地进行通信。

目标读者

本手册适用于任何想要创建分布式信息系统的用户。中央数据源和远程数据存储区可以是关系数据库系统，但并不以此为限。

开始之前

有关 MobiLink 与其它 SQL Anywhere 同步和复制技术的比较，请参见“比较同步技术”一节《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。

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

- **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*。

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

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`。

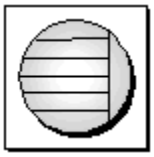
- **环境变量** 本文档介绍设置环境变量。在 Windows shell 中，环境变量使用语法 *%ENVVAR%* 来指定。在 Unix shell 中，环境变量使用语法 *$ENVVAR* 或 *${ENVVAR}* 来指定。

图标

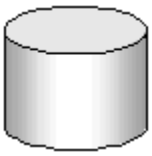
本文档中使用了下列图标。

- 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。

Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

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。

Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

使用 MobiLink 服务器技术

本节介绍 MobiLink 技术，同时介绍如何利用该技术在两个或两个以上数据源之间同步数据。

MobiLink 统一数据库 3
MobiLink 服务器 27
MobiLink 服务器选项 39
同步技术 119
MobiLink 性能 159
MobiLink 监控器 167
用于 MobiLink 的 SQL Anywhere 监控器 185
中继服务器 223
重定向器（不建议使用） 249
MobiLink 基于文件的下载 275

MobiLink 统一数据库

目录

统一数据库简介 4
建立统一数据库 6
RDBMS 相关同步脚本 8
Adaptive Server Enterprise 统一数据库 10
IBM DB2 LUW 统一数据库 12
IBM DB2 主机统一数据库 15
Microsoft SQL Server 统一数据库 19
MySQL 统一数据库 21
Oracle 统一数据库 23
SQL Anywhere 统一数据库 26

Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

统一数据库简介

统一数据库保存 MobiLink 所需的系统对象。多数情况下，它还保存应用程序数据，但您也可以其它形式保存全部或部分应用程序数据。

MobiLink 对 32 位和 64 位环境下的 Windows 和 Linux 支持统一数据库。统一数据库可以是符合 ODBC 标准的以下 RDBMS 之一：

- Adaptive Server Enterprise（不提供 64 位 Linux 支持）
- IBM DB2 LUW
- IBM DB2 主机
- Microsoft SQL Server
- MySQL
- Oracle
- SQL Anywhere

有关版本支持信息，请参见 http://www.sybase.com/detail?id=1062617。

SQL Anywhere 安装程序包括各种类型 RDBMS 的安装脚本。需要运行适当的安装脚本以便将该 RDBMS 用于 MobiLink。安装脚本将添加 MobiLink 所需的表和存储过程。

有关设置各种类型统一数据库的信息，请参见“建立统一数据库”一节第 6 页。

有关为特定的统一数据库编写同步脚本的信息，请参见“RDBMS 相关同步脚本”一节第 8 页。

同步到其它数据源

您的 MobiLink 环境必须具有一个已设置为统一数据库的数据库。但您可以同步统一数据库以外的其它数据源。其它数据源几乎可以为任何类型：文本文件、web 服务、非关系数据库、电子表格，等等。您可以：

- 创建一个混合应用程序，从中可以同步到统一数据库和某些其它数据源。
- 只同步到统一数据库。
- 只同步到另一数据源。

请参见“直接行处理”第 609 页。

修改统一数据库的限制

一些用户发现很难更改他们的统一数据库的模式。对于这类情况，MobiLink 提供了一些尽可能减少更改统一数据库的解决方案。例如，MobiLink 提供多种维护唯一主键的解决方案，其中一些对统一数据库模式的影响降至最低。

此外，将 MobiLink 系统对象放置在一个单独的数据库中几乎可以避免对统一数据库的所有影响。请参见“MobiLink 系统数据库”一节第 7 页。

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

远程表与统一表的关联方式

同步设计可以指定远程数据库中的表和列与统一数据库中的表和列之间的映射关系。通常，远程数据库中的表和列与统一数据库或其子集中的表和列完全匹配。

允许任何关系

远程数据库中的表不必与统一数据库中的表相同。一个远程应用程序表中的同步数据可以分发到不同表的列中，甚至可以分发到不同的统一数据库中。您可以使用同步脚本指定这些关系。

直接的关系比较简单

在最简单和最常见的设计中，使用的远程数据库中的表结构是统一数据库中表结构的子集。通过这种设计，远程数据库中的每个表都存在于统一数据库中。对应的表跟统一数据库中的表具有同样的结构和外键关系。

统一数据库中经常会包含未经同步的列和表。其中某些列或表可以用于同步。例如，在统一数据库中，时间戳列可以标识新行或更新行，也可以使用影子表跟踪删除操作。统一数据库中未同步的列或表也可能含有远程站点不需要的信息。

远程数据库也经常包含未同步的表或列。

另请参见

- “远程数据库和统一数据库之间的 MobiLink 数据映射”第 695 页

建立统一数据库

安装脚本

要设置某个数据库以便其可用作 MobiLink 统一数据库，必须运行安装脚本。SQL Anywhere 安装包括每个受支持 RDBMS 的安装脚本。这些脚本都位于 *install-dir\MobiLink\setup* 中。您也可以使用以下方法更新 MobiLink 系统设置：

- 在 Sybase Central 的 MobiLink 插件中，选择 [**模式**] » [**管理**] 并连接到服务器数据库。右击数据库名，选择 [**检查 MobiLink 系统设置**]。如果需要对您的数据库进行设置，系统将提示您继续。
- 使用 [**创建同步模型向导**] 或 [**部署同步模型向导**] 时，将在连接到服务器数据库时检查系统设置。如果需要对您的数据库进行设置，系统将提示您继续。请参见“MobiLink 模型简介”一节 《MobiLink - 入门》。

MobiLink 安装脚本将向数据库添加 MobiLink 系统表、存储过程、触发器和视图。这些表和过程是实现 MobiLink 同步所必需的。

有关安装的 MobiLink 系统表的信息，请参见“MobiLink 服务器系统表”第 653 页。

有关安装的存储过程的信息，请参见“MobiLink 系统过程”一节第 624 页。

如果要检查安装脚本所进行的操作，可以在文本编辑器中查看各安装脚本。

小心
运行安装脚本的数据库用户被授予了更新 MobiLink 系统表的权限，这是启动 MobiLink 服务器和配置 MobiLink 所必需的。请参见“所需权限”一节第 28 页。

有关如何运行安装脚本的说明，请参见有关 RDBMS 的部分：

- “Adaptive Server Enterprise 统一数据库”一节第 10 页
- “IBM DB2 LUW 统一数据库”一节第 12 页
- “IBM DB2 主机统一数据库”一节第 15 页
- “Microsoft SQL Server 统一数据库”一节第 19 页
- “MySQL 统一数据库”一节第 21 页
- “Oracle 统一数据库”一节第 23 页
- “SQL Anywhere 统一数据库”一节第 26 页

注意
如果所设置的统一数据库将用作 [QAnywhere 服务器存储库]，则应该将数据库配置成对于比较和字符串操作不区分大小写。

ODBC 连接

MobiLink 服务器需要与统一数据库建立 ODBC 连接。您的服务器必须配置适当的 ODBC 驱动程序，并在运行 MobiLink 服务器的计算机上为统一数据库创建 ODBC 数据源。

有关 MobiLink ODBC 驱动程序的详细信息，请参见“用于 MobiLink 的 iAnywhere Solutions ODBC 驱动程序”第 743 页。

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

有关可与 MobiLink 一起使用的 ODBC 驱动程序的更新信息及完整的功能说明，请参见推荐用于 MobiLink 的 ODBC 驱动程序。

MobiLink 系统数据库

在极少数情况下，您可能想将统一数据库拆分为两个：一个用于存储数据而另一个用于存储 MobiLink 系统信息。这样做时，不必为统一数据库添加 MobiLink 系统对象。所有 MobiLink 系统对象都可以存储在一个叫做 MobiLink 系统数据库的单独数据库中。

您的 MobiLink 系统数据库可以是所支持的作为统一数据库的任何数据库。它不必是与统一数据库相同的 RDBMS。

MobiLink 系统数据库易于设置。只要将 MobiLink 安装脚本应用到统一数据库之外的数据库即可。启动 MobiLink 服务器时，连接到两个数据库：

注意

- 只能在 Windows 上运行 MobiLink 服务器。
- 不能与 MobiLink [**创建同步模型向导**] 或 [模型] 模式一起使用 MobiLink 系统数据库。
- 将 MobiLink 系统对象存储在一个单独的数据库中会导致性能下降。

Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

RDBMS 相关同步脚本

MobiLink 使用同步脚本来定义用于同步数据的规则。同步脚本定义：

- 如何将从远程数据库上载的数据应用到统一数据库。
- 应从统一数据库将哪些数据下载到远程数据库。

请参见“编写同步脚本”第 293 页。

有关可为其编写脚本的事件的完整列表，请参见“同步事件”第 319 页。

有关各种类型统一数据库的特定信息，请参见：

- “SQL Anywhere 统一数据库”一节第 26 页
- “Adaptive Server Enterprise 统一数据库”一节第 10 页
- “IBM DB2 LUW 统一数据库”一节第 12 页
- “IBM DB2 主机统一数据库”一节第 15 页
- “Microsoft SQL Server 统一数据库”一节第 19 页
- “MySQL 统一数据库”一节第 21 页
- “Oracle 统一数据库”一节第 23 页

.NET 和 Java 同步脚本

您可以用您的数据库所使用的 SQL 语言版本编写同步逻辑。还可以使用 Java 或 .NET 编写可移植性更强并且更强大的脚本。Java 和 .NET 提供的灵活性超出每个 RDBMS 通过 SQL 提供的灵活程度，它们同时还提供与 SQL 的完全兼容。使用 Java 或 .NET 同步逻辑时，可以保存会话范围的变量、创建用户定义的过程、集成到外部服务器的验证等等。

有关 Java 同步逻辑的信息，请参见“编写 Java 同步逻辑”一节第 497 页。

有关 .NET 同步逻辑的信息，请参见“使用 .NET 编写同步脚本”第 551 页。

在脚本中调用过程

有些数据库（如 Microsoft SQL Server）要求使用 ODBC 语法编写带有参数的过程调用。

```
{ CALL procedure_name( {ml param1}, {ml param2}, ... ) }
```

您可以通过在过程定义中将参数定义为 OUT 或 INOUT 来返回值。

CHAR 列

在许多其它 RDBMS 中，CHAR 数据类型长度固定并填充空白以达到字符串的全长。在 SQL Anywhere 或 UltraLite 远程 MobiLink 数据库中，CHAR 与 VARCHAR 相同：值不填充空白以达到固定宽度。如果未将 SQL Anywhere 用作统一数据库，强烈建议在统一数据库中使用 VARCHAR 而不是 CHAR。如果必须使用 CHAR，在同步过程中可以使用 mlsrv11 -b 命令行选项删除字符串的尾随空白。此选项对于用于检测冲突的字符串比较十分重要。

请参见“-b 选项”一节第 48 页。

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

数据转换

有关 MobiLink 服务器与非 SQL Anywhere 的统一数据库通信时必须发生的数据转换的信息，请参见“远程数据库和统一数据库之间的 MobiLink 数据映射”第 695 页。

Adaptive Server Enterprise 统一数据库

将 Adaptive Server Enterprise 设置为统一数据库

要设置 Adaptive Server Enterprise 用作 MobiLink 统一数据库，必须运行设置过程，此过程将添加 MobiLink 同步所需的 MobiLink 系统表、存储过程、触发器和视图。可通过多种方法实现这一点：

- 运行 *syncase.sql* 安装脚本，它位于 *install-dir\MobiLink\setup* 中。
- 在 Sybase Central 的 MobiLink 插件中，选择 [**模式**] » [**管理**] 并连接到服务器数据库。右击数据库名，选择 [**检查 MobiLink 系统设置**]。如果需要对您的数据库进行设置，系统将提示您继续。
- 使用 [**创建同步模型向导**] 或 [**部署同步模型向导**] 时，将在连接到服务器数据库时检查系统设置。如果需要对您的数据库进行设置，系统将提示您继续。

注意

运行安装脚本的数据库用户是具有更改 MobiLink 系统表权限的唯一用户，这是配置 MobiLink 应用程序所必需的。请参见“所需权限”一节第 28 页。

MobiLink 服务器用于连接统一数据库的 RDBMS 用户必须能够使用 MobiLink 系统表、过程等，但不能使用任何限定符（例如，SELECT * from ml_user）。请参见“MobiLink 服务器系统表”第 653 页。

ODBC 驱动程序

必须使用随附于 Adaptive Server Enterprise 数据库的 ODBC 驱动程序为 Adaptive Server Enterprise 统一数据库设置 ODBC DSN。请参见：

- 推荐用于 MobiLink 的 ODBC 驱动程序
- Adaptive Server Enterprise 文档

Adaptive Server Enterprise 问题

- **BLOB 大小** 要下载数据大小大于 32 KB（缺省值）的 BLOB 数据，执行以下操作：
  - 在 Windows 上，将 [**Adaptive Server Enterprise ODBC 驱动程序配置**] 窗口的 [**高级**] 页上的 [**文本大小**] 设置为大于最大预期 BLOB。
  - 在 Linux 上，将 *obdc.ini* 文件中的 TextSize 条目设置为大于最大预期 BLOB。
- **CHAR 列** 在 Adaptive Server Enterprise 中，CHAR 数据类型长度固定并填充空白以达到字符串的全长。在 MobiLink 远程数据库（SQL Anywhere 或 UltraLite）中，CHAR 与 VARCHAR 相同：值不填充空白以达到固定宽度。强烈建议在统一数据库中使用 VARCHAR 而不是 CHAR。如果必须使用 CHAR，在同步过程中可以使用 mlsrv11 -b 命令行选项删除字符串的尾随空白。此选项对于用于检测冲突的字符串比较十分重要。

  有关详细信息，请参见“-b 选项”一节第 48 页。
- **数据类型映射** 列的数据类型必须在统一数据库和远程数据库之间正确映射。请参见“Adaptive Server Enterprise 数据映射”一节第 696 页。

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

- **版本 11.5 和更早版本的特殊注意事项** 无法使用 MobiLink 系统过程（例如 ml_add_connection_script）将长度超过 255 个字节的脚本添加到 Adaptive Server Enterprise 11.5 或其更早的版本中。定义较长的脚本，需要使用 Sybase Central 或直接插入。
- **VARBIT 限制** MobiLink 不支持将 0 长度（为空）VARBIT 或 LONG VARBIT 值同步到 Adaptive Server Enterprise 统一数据库。Adaptive Server Enterprise 不支持 VARBIT 类型，因此通常将这些类型同步到 Adaptive Server Enterprise 数据库上的 VARCHAR 或 TEXT 列。在 Adaptive Server Enterprise 上，空字符串值被转换为一个空格。SQL Anywhere 上的 VARBIT 列中不允许存在空格，因此尝试下载这些值将在远程数据库上导致错误。

隔离级别

请参见“MobiLink 隔离级别”一节第 156 页。

IBM DB2 LUW 统一数据库

MobiLink 支持用于 Linux、Unix 和 Windows 的 IBM DB2 LUW。MobiLink 不支持用于 AS/400 的 IBM DB2。

将 DB2 LUW 设置为统一数据库

要设置 DB2 用作 MobiLink 统一数据库，必须运行设置过程，此过程将添加 MobiLink 同步所需的 MobiLink 系统表、存储过程、触发器和视图。可通过多种方法实现这一点：

- 运行 *syncdb2.sql* 安装脚本，它位于 *install-dir\MobiLink\setup* 中。运行此文件前，必须将其复制到其它位置并且进行修改。说明如下。
- 在 Sybase Central 的 MobiLink 插件中，选择 [**模式**] » [**管理**] 并连接到服务器数据库。右击数据库名，选择 [**检查 MobiLink 系统设置**]。如果需要对您的数据库进行设置，系统将提示您继续。请注意，您必须执行以下过程的第 1 步。
- 使用 [**创建同步模型向导**] 或 [**部署同步模型向导**] 时，将在连接到服务器数据库时检查系统设置。如果需要对您的数据库进行设置，系统将提示您继续。请注意，您必须执行以下过程的第 1 步。

注意

运行安装脚本的数据库用户是具有更改 MobiLink 系统表权限的唯一用户，这是配置 MobiLink 应用程序所必需的。请参见“所需权限”一节第 28 页。

MobiLink 服务器用于连接统一数据库的 RDBMS 用户必须能够使用 MobiLink 系统表、过程等，但不能使用任何限定符（例如，SELECT * from ml_user）。请参见“MobiLink 服务器系统表”第 653 页。

♦ 运行 DB2 安装脚本

1. 要用安装脚本安装 MobiLink 系统表，IBM DB2 LUW 表空间必须使用至少 8 KB 页。如果表空间不使用 8 KB 页，请完成以下步骤：
   - 验证是否至少存在一个具有 8 KB 页的缓冲池。否则，创建一个具有 8 KB 页的缓冲池。
   - 创建一个使用 8 KB 页缓冲池的新的表空间和临时表空间。

   有关详细信息，请参见 DB2 LUW 文档。
2. 使用连接信息自定义 *syncdb2.sql*：
   a. 将 *syncdb2.sql* 复制到可以修改并存储它的新位置。
   b. *syncdb2.sql* 脚本包含一个缺省连接语句 [`connect to DB2Database`]。将此行变更为连接到您的 DB2 数据库。使用以下语法：

      ```
      connect to DB2Database user userid using password ~
      ```

      其中，*DB2Database*、*userid* 和 *password* 是您提供的名称。（*syncdb2.sql* 脚本使用代字号字符 (~) 作为命令分隔符。）
3. 运行 *syncdb2.sql*：

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

```
db2 -c -ec -td~ +s -v -f syncdb2.sql
```

ODBC 驱动程序

必须使用随附于 DB2 数据库的 ODBC 驱动程序为 DB2 统一数据库设置 ODBC DSN。请参见：

- Recommended ODBC Drivers for MobiLink
- IBM DB2 LUW 文档

DB2 LUW 问题

- **表空间容量** 您希望用作统一数据库的任何 DB2 LUW 数据库中的表空间和临时表空间都必须使用 8 KB 页。

  此外，有些列要求 LONG 型表空间。如果没有缺省的 LONG 型表空间，创建包含这些列的表的语句必须恰当限定，如下例所示：

  ```
  CREATE TABLE ... ( ... )
  IN tablespace
  LONG IN long-tablespace
  ```

  有关使用示例应用程序的示例，请参见“研究 MobiLink 的 CustDB 示例” 《MobiLink - 入门》。

- **会话范围的变量** 版本 8 以前的 DB2 LUW 不支持会话范围的变量。对此，方便的解决方法是使用一个基表，在其中加入列来包含 MobiLink 用户名和其它会话数据。此基表包含表示并发同步的行。

- **用户定义的过程** 版本 8.2 以前的 DB2 LUW 要求您将 SQL 过程编译为可执行库（如 DLL）。结果 DLL/共享库必须复制到服务器的专门目录中。请注意，您可以使用几种不同的语言（其中包括 C/C++ 和 Java）编写过程。

  有关 Java 和 .NET 同步脚本的详细信息，请参见：

  - “使用 Java 语言编写同步脚本”第 493 页
  - “使用 .NET 编写同步脚本”第 551 页

- **CHAR 列** 在 IBM DB2 LUW 中，CHAR 数据类型长度固定并以空白填充以达到字符串的全长。在 MobiLink 远程数据库（SQL Anywhere 或 UltraLite）中，CHAR 与 VARCHAR 相同：值不填充空白以达到固定宽度。强烈建议在统一数据库中使用 VARCHAR 而不是 CHAR。如果必须使用 CHAR，在同步过程中可以使用 mlsrv11 -b 命令行选项删除字符串的尾随空白。此选项对于用于检测冲突的字符串比较十分重要。

  请参见“-b 选项”一节第 48 页。

- **数据类型映射** 列的数据类型必须在统一数据库和远程数据库之间正确映射。有关详细信息，请参见“IBM DB2 LUW 数据映射”一节第 704 页。

- **在系统过程调用中使用双重引号** 使用 MobiLink 系统过程将脚本添加到 DB2 统一数据库时，需要使用双重引号。例如，如果通过 ml_add_table_script 添加的脚本包括用于任何其它统一数据库的行 [`SET "DELETED"=''Y''`]，则对于 DB2，必须将其写成 [`SET "DELETED" = ''''Y''''`]。

- **版本 5 和更早版本的特殊注意事项** 如果使用版本 6 之前的 IBM DB2 LUW，仅支持不超过 18 个字符的列名称和其它标识符。这意味着必须截断 MobiLink 系统过程的名称。例如，要调用 ml_add_connection_script，则使用名称 ml_add_connection_。

隔离级别

请参见“MobiLink 隔离级别”一节第 156 页。

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

IBM DB2 主机统一数据库

将 DB2 主机设置为统一数据库

要设置 DB2 主机用作 MobiLink 统一数据库，必须运行设置过程，此过程将添加 MobiLink 同步所需的 MobiLink 系统表、存储过程、触发器和视图。可以使用 SQL 或 JCL 方法执行此任务。

注意
运行安装脚本的数据库用户是具有更改 MobiLink 系统表权限的唯一用户，这是配置 MobiLink 应用程序所必需的。请参见“所需权限”一节第 28 页。

MobiLink 服务器用于连接统一数据库的 RDBMS 用户必须能够使用 MobiLink 系统表、过程等，但不能使用任何限定符（例如，SELECT * from ml_user）。请参见“MobiLink 服务器系统表”第 653 页。

♦ 设置 DB2 主机环境的通用步骤

1. 为 MobiLink 模式创建缓冲池，其页面大小不小于 8K 且具有行级锁定功能。需要行级锁定来处理相同表的并发同步。对于本示例，将缓冲池命名为 *BP8K*。
2. 为 MobiLink 模式创建一个名为 *MLTB8K* 的表空间，其缓冲池的页面大小为 8 KB。例如：

```
create tablespace MLTB8K in IANY bufferpool BP8K locksize row
grant use of tablespace IANY.MLTB8K to public
```

3. 如果还没有表空间，可为 MobiLink 模式过程创建 "负载管理器" 环境，其名称类似于 *MLWLM*。
4. 使用随附于 DB2 数据库的 ODBC 驱动程序为 DB2 主机统一数据库设置 ODBC DSN。请参见：
   - Recommended ODBC Drivers for MobiLink
   - IBM DB2 主机文档

♦ 使用 SQL 创建 MobiLink 系统表

注意
SQL 方法需要有使用 DSNTPSMP 创建存储过程的能力。如果您尚未启用 SQL 存储过程，则使用 JCL 方法。

1. 使用在“将 DB2 主机设置为统一数据库”一节第 15 页中列举的通用步骤设置 DB2 主机环境。
2. 修改 *syncd2m.sql* 安装脚本，它位于 *install-dir\MobiLink\setup* 中。

   注意
   确保继续操作之前保存了原始 *syncd2m.sql* 文件的备份副本。

   在 *syncd2m.sql* 文件中，
   - 使用表空间名 MLTB8K 替换 {MLTABLESPACE} 的所有实例。
   - 使用 [负载管理器] MLWLM 替换 {WLMENV}。

3. 使用以下命令行运行 *syncd2m.sql* 安装脚本：

```
dbisql -c "uid=user-id;pwd=password;DSN=dsn-name" -nogui syncd2m.sql
```

将生成消息日志文件 *syncd2m.txt*。

4. 打开 *syncd2m.txt* 以验证 DSNTPSMP 调用成功。

♦ 使用 JCL 创建 MobiLink 系统表

1. 使用在“将 DB2 主机设置为统一数据库”一节第 15 页中列举的通用步骤设置 DB2 主机环境。
2. 修改 *syncd2m_jcl.sql* 安装脚本，它位于 *install-dir\MobiLink\setup* 中。

> **注意**
> 确保在继续操作之前保存了原始 *syncd2m_jcl.sql* 文件的备份副本。

在 *syncd2m_jcl.sql* 文件中，

- 将所有出现的 {MLTABLESPACE} 替换为您限定的表空间，例如 MYDB.MYTS。
- 将所有出现的 {WLMENV} 替换为与 DB2 实例相关联的 "负载管理器" 的名称。

3. 启动 DBISQL 并将它连接到 DB2 主机。
4. 要在 DB2 主机中创建 Mobilink 表并定义 Mobilink 过程，可运行 *syncd2m_jcl.sql* 安装脚本经过编辑的副本，它位于 *install-dir\MobiLink\setup* 中。
5. 从 *%SQLANY%\MobiLink\setup* 目录中，通过 FTP 连接到主机并运行以下命令：

```
bin
    hash
    cd xmit
    quote site recfm=fb lrecl=80
    quote site cyl
    put d2mload.xmit
    put d2mdbrm.xmit
    quit
```

6. 主机上的两个 xmit 文件如下所示：
   - USERID.XMIT.D2MLOAD.XMIT
   - USERID.XMIT.D2MDBRM.XMIT

   USERID 是在通过 FTP 进行连接时您所给出的用户名。
7. 打开终端会话，然后从 "ISPF 命令 Shell" 运行以下命令：

```
RECEIVE INDATASET('USERID.XMIT.D2MLOAD.XMIT')
    RECEIVE INDATASET('USERID.XMIT.D2MDBRM.XMIT')
```

输出如下：

- USERID.ML.LOADLIB
- USERID.ML.DBRMLIB

8. 复制 *d2mrelod.jcl* 文件并按以下方式进行修改：

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

- 将 USERID 更改为主机用户 ID。
- 将 DSNDB0T 更改为您的 DB2 DSN。

9. 运行 *d2mrelod.jcl* 脚本经过编辑的副本，它位于 *install-dir\MobiLink\setup* 中。
10. 复制 *d2mbdpk.jcl* 文件并按以下方式进行修改：
    - 将 USERID 更改为主机用户 ID。
    - 将 DB0T 更改为您的 DB2 SSID。
11. 通过运行 *d2mbdpk.jcl* 经过编辑的副本将所有 SQL 过程绑定在一起。以下是 SQL 过程到装载模块名称的映射参考。

过程名称	装载模块名称
ml_add_user	mlaub
ml_delete_user	mldub
ml_del_sstate	mldssb
ml_reset_sstate	mlrssb
ml_del_sstate_b4	mldssbb
ml_add_lcs_chk	mlalcscb
ml_add_lcs	mlalcsb
ml_add_cs	mlacsb
ml_add_jcs	mlajcsb
ml_add_dcs	mladcsb
ml_add_lts_chk	mlaltscb
ml_add_lts	mlaltsb
ml_add_ts	mlatsb
ml_add_jts	mlajtsb
ml_add_dts	mladtsb
ml_add_property	mlapb
ml_add_column	mlacb
ml_set_device	mlsdb

过程名称	装载模块名称
ml_set_device_nt	mlsdnb
ml_set_dev_addr	mlsdab
ml_set_dev_addr_int	mlsdanb
ml_set_listening	mlslb
ml_set_listen_nt	mlslnb
ml_set_sis_sstate	mlssssb
ml_del_dev_addr	mlddab
ml_del_listen	mldlb
ml_delete_device	mlddb

DB2 主机已知问题

- **DB2 主机不能以 [模型] 模式工作** 使用 [**创建同步模型向导**] 时，不能使用 DB2 主机作为统一数据库。
- **SELECT 语句需要 FOR READ ONLY 子句** 缺省情况下 DB2 主机中的 SELECT 语句会对更新开放，这意味着数据库预期在 SELECT 语句之后为 UPDATE 语句而获取写锁定。

  为避免该写锁定并增强并发，将 FOR READ ONLY 附加到不在 UPDATE 语句之前的所有 SELECT 语句上。尽可能使用 SELECT 语句中的 FOR READ ONLY，特别是在 download_cursor 和 download_delete_cursor 脚本中。
- **Sysplex 需要时间同步** 在 Sysplex 中运行 DB2 主机统一数据库时，Sysplex 中的所有 LPAR 的时钟必须同步。时钟不同步会导致数据库同步过程中数据丢失。
- **数字近似** 近似数字拥有不同的可能值。以下是一个示例表。

类型	输入值	DB2 主机值	ASA 值
Real	123.456	123.4559936523	123.4560012817
Float	123.456	123.45599999999999	123.4560012817
Double	123.456	123.45599999999999	123.456

推荐方法是避免双精度和浮点列与 DB2 主机统一数据库同步。

隔离级别

请参见“MobiLink 隔离级别”一节第 156 页。

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

Microsoft SQL Server 统一数据库

将 Microsoft SQL Server 设置为统一数据库

> **注意**
> 运行安装脚本的数据库用户必须能够创建表、触发器和存储过程，因此必须具备 db_owner 角色。

要设置 Microsoft SQL Server 用作 MobiLink 统一数据库，必须运行设置过程，此过程将添加 MobiLink 同步所需的 MobiLink 系统表、存储过程、触发器和视图。可通过多种方法实现这一点：

- 运行 *syncmss.sql* 安装脚本，它位于 *install-dir\MobiLink\setup* 中。
- 在 Sybase Central 的 MobiLink 插件中，选择 [**模式**] » [**管理**]；连接到服务器数据库；右击数据库名并选择 [**检查 MobiLink 系统设置**]。如果需要对您的数据库进行设置，系统将提示您继续。如果想使用一个现有的 MobiLink 系统设置，则 default_schema 应该为 MobiLink 系统设置的模式。
- 使用 [**创建同步模型向导**] 或 [**部署同步模型向导**] 时，将在连接到服务器数据库时检查系统设置。如果需要对您的数据库进行设置，系统将提示您继续。

> **注意**
> 运行安装脚本的数据库用户是具有更改 MobiLink 系统表权限的唯一用户，这是配置 MobiLink 应用程序所必需的。请参见“所需权限”一节第 28 页。
>
> MobiLink 服务器用于连接统一数据库的 RDBMS 用户必须能够使用 MobiLink 系统表、过程等，但不能使用任何限定符（例如，SELECT * from ml_user）。请参见“MobiLink 服务器系统表”第 653 页。

ODBC 驱动程序

必须使用随附于 SQL Server 数据库的 ODBC 驱动程序为 SQL Server 统一数据库设置 ODBC DSN。请参见：

- Recommended ODBC Drivers for MobiLink
- Microsoft SQL Server 文档

SQL Server 问题

- **SET NOCOUNT ON** 对于 Microsoft SQL Server，应将 SET NOCOUNT ON 指定为所有存储过程或通过 ODBC 执行的 SQL 批处理中的第一个语句。缺少此选项，网络缓冲区可能溢出，结果以静默方式丢失数据。这被称为 SQL Server 问题。
- **过程调用** Microsoft SQL Server 要求使用 ODBC 语法编写带有参数的过程调用：

  ```
  { CALL procedure_name( {ml param1}, {ml param2}, ... ) }
  ```

- **CHAR 列** 在 Microsoft SQL Server 中，CHAR 数据类型长度固定并填充空白以达到字符串的全长。在 MobiLink 远程数据库（SQL Anywhere 或 UltraLite）中，CHAR 与 VARCHAR 相同：值不填充空白以达到固定宽度。我们强烈建议在统一数据库中使用 VARCHAR 而不是 CHAR。

Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

如果必须使用 CHAR，在同步过程中可以使用 mlsrv11 -b 命令行选项删除字符串的尾随空白。此选项对于用于检测冲突的字符串比较十分重要。

请参见“-b 选项”一节第 48 页。

- **数据类型映射** 列的数据类型必须在统一数据库和远程数据库之间正确映射。有关详细信息，请参见“Microsoft SQL Server 数据映射”一节第 720 页。
- **示例数据库问题** SQL 服务器 AdventureWorks 示例数据库包含计算列。您无法同步计算列。可以将列设置为仅下载，或从同步中排除该列。
- **在 upload_update 脚本中实现冲突检测** SQL Server NOCOUNT 选项的行为表示 MobiLink 服务器在某些时候不能准确地评估出上载脚本更改了多少行。对于 SQL Server，在 upload_update 脚本中实现存储过程的冲突检测会更安全。

隔离级别

请参见“MobiLink 隔离级别”一节第 156 页。

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

MySQL 统一数据库

MobiLink 服务器支持 MySQL Community 和 Enterprise 服务器 5.1.22 或更高版本。QAnywhere 和 MobiLink 模型不支持 MySQL。

将 MySQL 设置为统一数据库

要设置 MySQL 用作 MobiLink 统一数据库，必须运行设置过程，此过程将添加 MobiLink 同步所需的 MobiLink 系统表、存储过程、触发器和视图。可通过两种方法实现这一点：

- 使用 MySQL 命令行工具或 MySQL Query Browser，运行 *syncmys.sql* 安装脚本，它位于 *install-dir\MobiLink\setup* 中。确保您的 MySQL 用户 ID 拥有创建触发器的权限。
- 在 Sybase Central 的 MobiLink 插件中，选择 [**模式**] » [**管理**] 并连接到服务器数据库。右击数据库名，选择 [**检查 MobiLink 系统设置**]。如果需要对您的数据库进行设置，系统将提示您继续。请注意，如果想使用一个现有的 MobiLink 系统设置，则 default_schema 应该为 MobiLink 系统设置的模式。

注意

运行安装脚本的数据库用户是具有更改 MobiLink 系统表权限的唯一用户，这是配置 MobiLink 应用程序所必需的。请参见“所需权限”一节第 28 页。

MobiLink 服务器用于连接统一数据库的 RDBMS 用户必须能够使用 MobiLink 系统表、过程等，但不能使用任何限定符（例如，SELECT * from ml_user）。请参见“MobiLink 服务器系统表”第 653 页。

ODBC 驱动程序

必须使用 MySQL web 站点所提供的 ODBC 驱动程序为 MySQL 统一数据库设置 ODBC DSN。MobiLink 服务器支持 MySQL ODBC 驱动程序 5.1.3 或更高版本。请参见：

- Recommended ODBC Drivers for MobiLink

要指定 Unix 中的 ODBC 配置文件，执行以下操作之一：

- 将 *ODBC.INI* 文件置于当前用户的主目录中。
- 创建 [**ODBCINI**] 环境变量并将其置于 *ODBC.INI* 文件的目录位置。

如果某些同步脚本包含用分号分隔的批处理 SQL 命令，那么，当您配置 MobiLink 服务器的 DSN 以连接 MySQL 数据库时，可能需要选中在 [**MySQL Connector/ODBC Data Source Configuration**] 窗口的 [**Flags 3**] 页面上的 [**Allow Multiple Statements**] 复选框。

MySQL 问题

- **存储引擎** MobiLink 服务器要求缺省存储引擎遵从 ACID。如果缺省存储引擎不遵从 ACID，则应确保使用遵从 ACID 的存储引擎（如 InnoDB 和 Falcon）来创建所有 MobiLink 服务器表。
- **存储过程** 在存储过程调用中不能使用 INOUT 或 OUT 参数。需要这些参数的过程必须作为返回 OUT 值的函数实现。

需要 INOUT 参数的服务器事件，如 authenticate_user 和 modify_user，必须作为函数实现并且使用 SELECT 语句运行，而不是 CALL 语句。

因为用户定义的命名参数在服务器事件运行之后不会修改，因此不受支持。

- **游标脚本** 事件 upload_fetch、download_cursor 和 download_delete_cursor 必须使用 SELECT 语句来调用，而这些事件由 MobiLink 服务器以读取已提交的隔离级别来运行。MySQL ODBC 驱动程序中的错误会使服务器读取未提交操作，如 INSERT、UPDATE 和 DELETE 语句，这样就会导致统一数据库和远程数据库之间的数据不一致。

  为解决此问题，将 LOCK IN SHARE MODE 子句附加到 SELECT 语句。例如，

  ```
  SELECT column1 FROM table1 WHERE id > 0 LOCK IN SHARE MODE
  ```

  该子句会防止 SELECT 语句执行未提交操作。

- **批量上载** MobiLink 服务器依赖于 MySQL ODBC 驱动程序，而该程序当前不支持批量上载。
- **MLSD** MobiLink 服务器依赖于 MySQL ODBC 驱动程序，而该程序当前不支持 MSDTC。
- **用于 Unix 的 64 位 MobiLink 服务器上的 SQLLEN 数据类型** MySQL ODBC 驱动程序将 SQLLEN 定义为一个 32 位整数，导致与将 SQLLEN 定义为一个 64 位整数的 64 位 MobiLink 服务器存在差异。如果在 64 位 Unix 环境下运行 MobiLink，则必须将以下内容添加到 ODBC 配置文件，

  ```
  length32=1
  ```

  该条目强制服务器将 SQLLEN 读取为 32 位整数。您的配置应与以下示例类似：

  ```
  [a_mysql_dsn]
  Driver=full_path/libmyodbc5.so
  server=host_name
  uid=user_name
  pwd=user_password
  database=database_name
  length32=1
  ```

- **MySQL 服务器配置** MobiLink 同步脚本作为文本存储在 ml_script 表中，需要时可对其进行检索。可能需要在 *my.ini* 文件中将 max_allowed_packet 设置为 16m 或更大。

隔离级别

请参见“MobiLink 隔离级别”一节第 156 页。

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

Oracle 统一数据库

将 Oracle 设置为统一数据库

要设置 Oracle 用作 MobiLink 统一数据库，必须运行设置过程，此过程将添加 MobiLink 同步所需的 MobiLink 系统表、存储过程、触发器和视图。可通过多种方法实现这一点：

- 运行 *syncora.sql* 安装脚本，它位于 *install-dir\MobiLink\setup* 中。
- 在 Sybase Central 的 MobiLink 插件中，选择 [**模式**] » [**管理**] 并连接到服务器数据库。右击数据库名，选择 [**检查 MobiLink 系统设置**]。如果需要对您的数据库进行设置，系统将提示您继续。请注意，如果您为现有 MobiLink 系统设置取了别名，则应该以其模式具有 MobiLink 系统设置的用户身份进行连接。
- 使用 [**创建同步模型向导**] 或 [**部署同步模型向导**] 时，将在连接到服务器数据库时检查系统设置。如果需要对您的数据库进行设置，系统将提示您继续。

注意

运行安装脚本的数据库用户是具有更改 MobiLink 系统表权限的唯一用户，这是配置 MobiLink 应用程序所必需的。请参见“所需权限”一节第 28 页。

MobiLink 服务器用于连接统一数据库的 RDBMS 用户必须能够使用 MobiLink 系统表、过程等，但不能使用任何限定符（例如，SELECT * from ml_user）。请参见“MobiLink 服务器系统表”第 653 页。

ODBC 驱动程序

必须为 Oracle 统一数据库设置 ODBC DSN。请参见：

- “iAnywhere Solutions Oracle 驱动程序”一节第 745 页
- Recommended ODBC Drivers for MobiLink

Oracle 问题

- **存储过程** 如果在 Oracle 中使用存储过程，则必须为 Oracle ODBC 驱动程序选择 [过程返回结果] 选项。

  请参见“iAnywhere Solutions Oracle 驱动程序”一节第 745 页。

- **会话范围的变量** Oracle 不提供会话范围的变量。但可以将会话范围的信息存储在 Oracle 程序包的变量中。Oracle 程序包允许创建、修改和删除这些变量；这些变量可以与当前 Oracle 程序包持续同样长的时间。
- **自动增量方法** 为了维护主键的唯一性，您可以用 Oracle 序列生成与自动增量字段的键列表相似的键列表。CustDB 示例数据库提供了代码示例，在 *Samples\MobiLink\CustDB\custora.sql* 中可以找到它们。但是，与自动增量不同的是，您必须显式参考该序列。如果 INSERT 语句中未引用此列，自动增量将自动插入列值。
- **Oracle 不支持空字符串** 在 Oracle 中，空字符串被视为 NULL。在 SQL Anywhere 和 UltraLite 中，空字符串具有空值之外的其它含义。因此，拥有 Oracle 统一数据库后，应避免在客户端数据库中使用空字符串。

- **CHAR 列** 在 Oracle 中，CHAR 数据类型长度固定并填充空白以达到字符串的全长。在 MobiLink 远程数据库（SQL Anywhere 或 UltraLite）中，CHAR 与 VARCHAR 相同：值不填充空白以达到固定宽度。强烈建议在统一数据库中使用 VARCHAR 而不是 CHAR。如果必须使用 CHAR，在同步过程中可以使用 mlsrv11 -b 命令行选项删除字符串的尾随空白。此选项对于用于检测冲突的字符串比较十分重要。

  请参见“-b 选项”一节第 48 页。

- **数据类型映射** 列的数据类型必须在统一数据库和远程数据库之间正确映射。有关详细信息，请参见“Oracle 数据映射”一节第 731 页。

隔离级别

请参见“MobiLink 隔离级别”一节第 156 页。

使用 Oracle varray

iAnywhere Solutions 11 - Oracle ODBC 驱动程序支持在存储过程中使用 Oracle varray。与在存储过程编写的上载脚本（upload_insert、upload_update 和 upload_delete）中不使用 varray 相比，在存储过程编写的上载脚本中使用 varray 可以提高 MobiLink 服务器的性能。对于 INSERT、UPDATE 和 DELETE 等简单 SQL 语句，在不使用存储过程的情况下通常都提供了最佳性能，但是使用存储过程（包括 varray 技术）可提供简单语句无法提供的应用业务逻辑的机会。

varray 示例

以下是使用 varray 的简单示例：

1. 创建一个包含 3 列名为 my_table 的表。

   ```
   create table my_table ( pk integer primary key, c1 number(20), c2
   varchar2(4000) )
   ```

2. 使用 varrays 创建用户定义的收集类型。

   ```
   create type my_integer is varray(100) of integer;
   create type my_number is varray(100) of number(20);
   create type my_varchar is varray(100) of varchar2(8000);
   ```

   my_varchar 被定义为 varray，它包含 100 个元素，每个元素都是 varchar2 类型数据，其宽度为 8000。宽度是为 my_table 所指定的宽度大小的两倍。

3. 创建插入存储过程。

   ```
   create or replace procedure my_insert_proc( pk_v my_integer, c1_v
   my_number, c2_v my_varchar )
   is
   c2_value my_varchar;
   begin
         c2_value := c2_v;  -- Work around an Oracle bug
         FORALL i in 1 .. pk_v.COUNT
               insert into my_table ( pk, c1, c2 ) values( pk_v(i), c1_v(i),
   c2_value(i) );
   end;
   ```

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

varray 限制

当在存储过程中使用 varray 时有以下限制：

- DSN 必须使 [**启用 Microsoft 分布式事务**] 复选框未选中。
- 不支持 BLOB 和 CLOB varray。
- 包含 varray 的存储过程必须是独立的过程，不能是封装的过程。
- 如果 varray 的数据类型是 CHAR、VARCHAR、NCHAR 或 NVARCHAR，则用户定义的 varray 类型长度必须是为表列所指定的长度大小的两倍。
- 由 MobiLink 服务器发送到 Oracle 统一数据库的 varray 中的行数由 -s 选项设置，不是在 varray 类型中声明的 varray 的大小。-s 选项设置的值一定不要大于同步脚本正在使用的最小 varray 类型大小。如果大于最小 varray 类型大小，则 MobiLink 服务器会发出错误。请参见“-s 选项”一节第 85 页。

SQL Anywhere 统一数据库

将 SQL Anywhere 设置为统一数据库

要设置 SQL Anywhere 用作 MobiLink 统一数据库，必须运行设置过程，此过程将添加 MobiLink 同步所需的 MobiLink 系统表、存储过程、触发器和视图。可通过多种方法实现这一点：

- 运行 *syncsa.sql* 安装脚本，它位于 *install-dir\MobiLink\setup* 中。
- 在 Sybase Central 的 MobiLink 插件中，选择 [**模式**] » [**管理**] 并连接到服务器数据库。右击数据库名，选择 [**检查 MobiLink 系统设置**]。如果需要对您的数据库进行设置，系统将提示您继续。
- 使用 [**创建同步模型向导**] 或 [**部署同步模型向导**] 时，将在连接到服务器数据库时检查系统设置。如果需要对您的数据库进行设置，系统将提示您继续。

注意

运行安装脚本的数据库用户是具有更改 MobiLink 系统表权限的唯一用户，这是配置 MobiLink 应用程序所必需的。请参见“所需权限”一节第 28 页。

MobiLink 服务器用于连接统一数据库的 RDBMS 用户必须能够使用 MobiLink 系统表、过程等，但不能使用任何限定符（例如，SELECT * from ml_user）。请参见“MobiLink 服务器系统表”第 653 页。

设置 ODBC 驱动程序

必须为 SQL Anywhere 统一数据库设置 ODBC DSN。SQL Anywhere 的 ODBC 驱动程序与 SQL Anywhere 一起安装。

有关 SQL Anywhere ODBC 驱动程序的信息，请参见“创建 ODBC 数据源”一节 《SQL Anywhere 服务器 - 数据库管理》。

隔离级别

请参见“MobiLink 隔离级别”一节第 156 页。

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

MobiLink 服务器

目录

运行 MobiLink 服务器 28
停止 MobiLink 服务器 30
记录 MobiLink 服务器操作 31
在当前会话外运行 MobiLink 服务器 33
在服务器群中运行 MobiLink 服务器 37
MobiLink 服务器启动故障排除 38

运行 MobiLink 服务器

所有 MobiLink 客户端都通过 MobiLink 服务器进行同步。而不直接连接数据库服务器。必须先启动 MobiLink 服务器，然后再进行 MobiLink 客户端同步。

有关 mlsrv11 命令行选项的列表，请参见“MobiLink 服务器选项”第 39 页。

MobiLink 服务器通过 ODBC 打开与统一数据库服务器之间的连接。然后，它接受来自远程应用程序的连接并对同步过程进行控制。

♦ 启动 MobiLink 服务器

- 运行 mlsrv11。使用 -c 选项为统一数据库指定 ODBC 连接参数。

有关连接参数的信息，请参见“-c 选项”一节第 50 页。

必须指定连接参数。还提供其它选项，但这些选项是可选的。使用这些选项可指定服务器的工作方式。例如，可指定高速缓存大小和记录选项。

有关 mlsrv11 选项的详细信息，请参见“mlsrv11 语法”一节第 41 页。

注意
mlsrv11 选项允许您指定 MobiLink 服务器的工作方式。若要控制服务器的操作，请定义发生同步事件时调用的脚本。请参见“同步事件”第 319 页。

示例

以下命令使用 ODBC 数据源 *SQL Anywhere 11 CustDB* 启动 MobiLink 服务器，以标识统一数据库。请在一行中输入整个命令。

```
mlsrv11
  -c "dsn=SQL Anywhere 11 CustDB;uid=DBA;pwd=sql"
  -zs MyServer
  -o mlsrv.log
  -vcr
  -x tcpip(port=3303)
  -xo tcpip
```

在此例中，-c 选项提供包含 ODBC 数据源名 (DSN) 以及验证的连接字符串。-zs 选项提供服务器名。-o 选项指定应将日志文件命名为 *mlsrv.log*。由于使用了 -vcr 选项，*mlsrv.log* 中包含详细内容。-x 选项为版本 10 和 11 客户端打开端口，而 -xo 选项为版本 8 和 9 客户端打开端口。您必须使用 -x 选项或 -xo 选项指定一个端口，否则命令失败，因为这两个选项都使用此缺省端口。

也可以将 MobiLink 服务器作为 Windows 服务或 Unix 守护程序来启动。请参见“在当前会话外运行 MobiLink 服务器”一节第 33 页。

所需权限

必须为 MobiLink 服务器指定一个数据库用户以与数据库服务器进行连接。使用 mlsrv11 -c 选项或在 DSN 中指定数据库用户。

此数据库用户必须拥有对 MobiLink 系统表进行选择、插入、更新和删除操作的完全权限，还必须具有执行 MobiLink 系统过程的权限。缺省情况下，运行 MobiLink 安装脚本的数据库用户拥有上

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

述权限。如果希望通过其他数据库用户运行 MobiLink 服务器，则必须为该用户授予 ml_* 表和 ml_add_*_script 系统过程的相关权限。

例如，在 SQL Anywhere 统一数据库中，您可按如下方式授予所需权限：

```
CREATE USER DBUser IDENTIFIED BY SQL;
GRANT ALL ON dbo.ml_user to DBUser;
...
GRANT EXECUTE ON dbo.ml_add_table_script TO DBUser;
...
```

必须为每个 MobiLink 系统表和系统过程授予权限。有关全部 MobiLink 系统表和系统过程的列表，请参见“MobiLink 服务器系统表”第 653 页和“MobiLink 服务器系统过程”第 623 页。

该数据库用户还需要有关在 MobiLink 脚本中引用的所有表的适当权限，以及执行 MobiLink 脚本中所引用的任何过程的权限。

有关设置权限的详细信息，请参见“GRANT 语句”一节 《SQL Anywhere 服务器 - SQL 参考》。

有关安装脚本的详细信息，请参见“建立统一数据库”一节第 6 页。

Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

停止 MobiLink 服务器

可以从启动 MobiLink 服务器的计算机来停止 MobiLink 服务器。可用以下方法停止 MobiLink 服务器：

- 使用 MobiLink 停止实用程序 (mlstop)。
- 在 MobiLink 服务器窗口中单击 [关闭]。
- 在 Windows 中，右击系统任务栏中的图标并选择 [关闭]。
- 在 Unix 上运行而没有 MobiLink 服务器窗口时，请键入 Q。
- 使用 MobiLink 服务器 API 中的 shutdown 方法。

另请参见

- “MobiLink 停止实用程序 (mlstop)”一节第 649 页
- 用于 Java 的服务器 API： “shutdown 方法”一节第 537 页
- 用于 .NET 的服务器 API： “ShutDown 方法”一节第 591 页

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

记录 MobiLink 服务器操作

在开发过程中和进行故障排除时，对服务器执行的操作进行记录特别有帮助。不建议对生产环境的正常运行使用详细输出模式，因为详细输出会降低性能。

将输出记录到文件中

将记录输出发送到 MobiLink 服务器消息窗口。此外，也可以使用 -o 选项将输出发送到消息日志文件中。以下命令会将输出发送到名为 *mlsrv.log* 的消息日志文件中。

```
mlsrv11 -o mlsrv.log -c ...
```

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 选项指定应使用日志文件。
- 如果想先删除日志文件以前的内容，然后再向其发送消息，请使用 -ot 选项指定应使用日志文件。
- 除 -o 或 -ot 之外，还可以使用 -on 选项来指定一个大小，日志文件达到该大小时将以扩展名 .old 重命名日志文件，然后使用原来的文件名启动新文件。
- 除 -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将启动新的日志文件，并根据日期和顺序号为新文件指定新名称。

请参见：

- “-o 选项”一节第 73 页
- “-on 选项”一节第 74 页
- “-os 选项”一节第 76 页
- “-ot 选项”一节第 77 页

控制记录输出的数量

可以使用 -v 选项控制记录到消息日志文件中并显示在 MobiLink 服务器窗口中的信息。请参见“-v 选项”一节第 97 页。

控制报告哪些警告消息

还可以控制报告哪些警告消息。

有关详细信息，请参见：

- “-zw 选项”一节第 116 页
- “-zwd 选项”一节第 117 页
- “-zwe 选项”一节第 118 页

查看 MobiLink 服务器日志

您可以通过下列方法查看 MobiLink 日志：

- 在 MobiLink 服务器消息窗口中
- 通过打开日志文件
- 使用 Sybase Central 中的 MobiLink 服务器日志文件查看器

要在 MobiLink 服务器消息窗口外查看日志信息，您必须将信息记录到文件。请参见“将输出记录到文件中”一节第 31 页。

MobiLink 服务器日志文件查看器

要查看 MobiLink 服务器日志，打开 Sybase Central，然后选择 [**工具**] » [**MobiLink 11**] » [**MobiLink 服务器日志文件查看器**]。将提示您选择要查看的日志文件。按住 shift 键，可同时打开多个日志文件。

MobiLink 服务器日志文件查看器将读取 MobiLink 日志文件中存储的信息。它不连接到 MobiLink 服务器，也不更改日志文件的组成。

MobiLink 服务器日志文件查看器允许您过滤所查看的信息。此外，它根据日志中的信息提供统计信息。

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

在当前会话外运行 MobiLink 服务器

可以将 MobiLink 服务器设置为始终可用。为了使之更简单，可以运行适用于 Windows 和 Unix 的 MobiLink 服务器，这样当您注销计算机时 MobiLink 服务器仍然保持运行。执行此操作的方法取决于所使用的操作系统。

- **Unix 守护程序** 可以使用 mlsrv11 -ud 选项将 MobiLink 服务器作为守护程序运行。这将使 MobiLink 服务器在后台运行，而且在您注销后它仍会继续运行。
- **Windows 服务** 可以将 Windows MobiLink 服务器作为服务运行。

  要停止正作为服务运行的 MobiLink 服务器，可使用 mlstop、dbsvc 或 Windows 服务管理器。

另请参见

- “-ud 选项”一节第 94 页
- “用于 Linux 的服务实用程序 (dbsvc)”一节 《SQL Anywhere 服务器 - 数据库管理》
- “在当前会话外部运行服务器”一节 《SQL Anywhere 服务器 - 数据库管理》

将 Unix MobiLink 服务器作为守护程序运行

在 Unix 上要使 MobiLink 服务器在后台运行并独立于当前会话，可以将其作为守护程序运行。

♦ **将 Unix MobiLink 服务器作为守护程序运行**

- 启动 MobiLink 服务器时使用 -ud 选项。例如：

```
mlsrv11 -c "dsn=SQL Anywhere 11 Demo;uid=DBA;pwd=sql" -ud
```

请参见“-ud 选项”一节第 94 页。

另请参见

- “用于 Linux 的服务实用程序 (dbsvc)”一节 《SQL Anywhere 服务器 - 数据库管理》

将 Windows MobiLink 服务器作为服务运行

若要在后台运行 Windows MobiLink 服务器并使其独立于当前会话，您可以将其作为服务运行。

可以通过命令行或 Sybase Central 中的 [**服务**] 选项卡执行以下服务管理任务：

- 添加、编辑和删除服务。
- 启动和停止服务。
- 修改用于控制服务的参数。

另请参见

- “用于 Windows 的服务实用程序 (dbsvc)”一节 《SQL Anywhere 服务器 - 数据库管理》

Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

添加、修改或删除服务

Sybase Central 中的服务图标使用表示服务处于运行还是停止状态的图标来显示每个服务的当前状态。

♦ 添加新服务 (Sybase Central)

1. 在 Sybase Central 中，单击左窗格内的 MobiLink 11。
2. 在右窗格中，单击 [**服务**] 选项卡。
3. 在右窗格中，右击并选择 [**新建**] » [**服务**]。
4. 请按照 [**创建服务向导**] 中的说明进行操作。

也可以使用 dbsvc 实用程序来创建服务。请参见“用于 Windows 的服务实用程序 (dbsvc)”一节《SQL Anywhere 服务器 - 数据库管理》。

♦ 删除服务 (Sybase Central)

- 选择服务，然后单击 [**编辑**] » [**删除**]。

♦ 更改服务的参数

- 右击服务并选择 [**属性**]。

对一个服务配置所做的更改将在下次启动服务时生效。

设置启动选项

下面的选项用于控制 MobiLink 服务的启动行为。可以在 [**服务属性**] 窗口的 [**常规**] 选项卡上设置它们。

- **自动** 如果选择 [**自动**]，则每当 Windows 操作系统启动时，服务就会启动。此设置适用于数据库服务器和其它始终处于运行状态的应用程序。
- **手工** 如果选择 [**手工**]，则只有具备管理员权限的用户才能启动服务。有关管理员权限的信息，请参见 Windows 文档。
- **已禁止** 如果选择 [**已禁用**]，则服务不启动。

启动选项将在下次启动 Windows 时应用。

指定命令行选项

[**服务属性**] 窗口的 [**配置**] 选项卡提供一个 [**文件名**] 文本框（用于输入程序可执行文件路径）以及一个 [**参数**] 文本框（用于为服务输入命令行选项）。不要在 [**参数**] 框中键入程序可执行文件的名称。

例如，若要启动具有详细记录的 MobiLink 同步服务，请在 [参数] 框中键入以下内容：

```
-c "dsn=SQL Anywhere 11 Demo;uid=DBA;pwd=sql"
-vc
```

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

服务的命令行选项与可执行文件的命令行选项相同。请参见“MobiLink 服务器选项”第 39 页。

设置帐户选项

可以选择服务藉以运行的帐户。大多数服务都在特殊的 LocalSystem 帐户下运行，LocalSystem 帐户是服务的缺省选项。打开 [**服务属性**] 窗口中的 [**帐户**] 选项卡并输入另一帐户的信息，就可以将服务设置为使用该帐户登录。

如果选择使用非 LocalSystem 帐户运行服务，此帐户必须拥有 "作为服务登录" 权限。有关高级特权的信息，请参见 Microsoft Windows 文档。

服务的图标是否显示在任务栏或桌面上取决于所选择的帐户，以及是否选中了 [**允许服务与桌面交互**]，如下所述：

- 如果服务以 LocalSystem 的名义运行，并且 [**服务属性**] 窗口中的 [**允许服务与桌面交互**] 处于选中状态，则在运行该服务的计算机上登录 Windows XP/200x 的每个用户的桌面上都会出现一个图标。任何用户都可以打开应用程序窗口并停止作为服务运行的程序。
- 如果服务以 LocalSystem 的名义运行，而 [**服务属性**] 窗口中的 [**允许服务与桌面交互**] 处于未选中状态，则任何用户的桌面上都不会出现图标。只有拥有更改服务状态权限的用户才可以停止服务。
- 如果服务使用另一个帐户运行，则桌面上不会出现图标。只有拥有更改服务状态权限的用户才可以停止服务。

更改可执行文件

要更改与 Sybase Central 中服务相关的程序可执行文件，请单击 [**服务属性**] 窗口中的 [**配置**] 选项卡，然后在 [**文件名**] 框中键入新路径和文件名。

如果将可执行文件移动到一个新的目录，您必须修改这一项。

启动和停止

- **启动或停止服务**

1. 在 Sybase Central 中，单击左窗格中的 MobiLink 11，然后在右窗格中打开 [**服务**] 选项卡。
2. 右击服务并选择 [**启动**] 或 [**停止**]。

如果启动某个服务，则它会一直运行，直到将其停止。关闭 Sybase Central 或注销并不会停止该服务。

停止服务会关闭与数据库的所有连接并停止数据库服务器。对其它应用程序而言，此程序将关闭。

一次运行多个服务

尽管可以将 [控制面板] 中的 Windows 服务管理器用于某些任务，但却无法从 Windows 服务管理器安装或配置 MobiLink 服务。可以使用 Sybase Central 来执行 MobiLink 的所有服务管理。

如果从 Windows 的 [控制面板] 打开 Windows 服务管理器，就可以看到一个服务列表。在用户于安装该服务时提供的 [服务名] 前面加上 SQL Anywhere，便构成了 SQL Anywhere 服务的名称。所有已安装的服务都会出现在该列表中。

本部分介绍有关一次运行多个服务的特定主题。

服务依赖性

在某些情况下您可能需要运行多个可执行文件作为服务，而这些可执行文件可能相互依赖。例如，若要同步，必须运行 MobiLink 服务器和数据库服务器。

在此类情况下，服务必须按正确的顺序启动。如果 MobiLink 同步服务先于统一数据库服务器启动，它将因无法找到统一数据库服务器而失败。正确的顺序是：当启动 MobiLink 服务器时，数据库服务器已处于运行状态。（如果统一数据库服务器位于另一计算机上，则此规则不适用。）

使用服务组可以防止这些问题出现，可使用 Sybase Central 管理服务组。

服务组

可以将系统上的每个服务都指派为服务组的成员。缺省情况下，每项服务都从属于一个组。MobiLink 服务器的缺省组是 SQLANYMobiLink。

在对服务进行配置以确保其按正确的顺序启动之前，必须检查这些服务是否为某一适当组的成员。可以通过 Sybase Central 检查服务所属的组及更改此组。

♦ 检查并更改服务所属的组

1. 在 Sybase Central 中，单击左窗格中的 MobiLink 11，然后在右窗格中打开 [**服务**] 选项卡。
2. 右击服务并选择 [**属性**]。
3. 单击 [**依赖性**] 选项卡。最上面的文本框显示该服务所属组的名称。
4. 单击 [**更改**] 以显示系统上可用组的列表。
5. 选择某个组或键入新组的名称。
6. 单击 [**确定**]，将服务指派到该组。

管理服务依赖性

可以使用 Sybase Central 指定服务的依赖性。例如：

- 您可以确保在当前服务之前已经至少启动一个组。
- 您可以确保在当前服务之前启动任一服务。

♦ 向依赖性列表中添加服务或组

1. 在 Sybase Central 中，单击左窗格中的 MobiLink 11，然后在右窗格中打开 [**服务**] 选项卡。
2. 右击服务并选择 [**属性**]。
3. 单击 [**依赖性**] 选项卡。
4. 单击 [**添加服务**] 或 [**添加服务组**] 以便向依赖性列表中添加服务或组。
5. 从列表中选择一个服务或组。
6. 单击 [**确定**] 将服务或组添加到依赖性列表。

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

在服务器群中运行 MobiLink 服务器

需要单独授予许可的组件
共享状态模式是 MobiLink 高可用性选项的一项功能，它需要单独的许可证。请参见“单独授权的组件”一节 《SQL Anywhere 11 - 简介》。

MobiLink 服务器群是一个环境，该环境中存在一个以上的 MobiLink 服务器，它们同时将同一组远程数据库与同一个统一数据库进行同步。通常，大规模部署或故障转移功能需要这样的设置。这些 MobiLink 服务器群部署要求 MobiLink 在共享状态模式下运行，并且如果使用了 HTTP 通信链接，还可能需要使用中继服务器。对于基于 TCP 的流，TCP 负载平衡器将起作用。使用多个服务器时，无法使用可重新启动的下载。

缺省情况下，MobiLink 不在共享服务器状态下运行。

♦ 启用共享服务器状态

1. 使用 -zs 命令行选项为每个 MobiLink 服务器赋予唯一的名称。这些名称用于在统一数据库中管理群的状态。请参见“-zs 选项”一节第 112 页。
2. 使用 -ss 选项在共享服务器状态模式下启动 MobiLink。如果设置了此选项，则 MobiLink 服务器在启动时会将以下消息输出到日志：**此服务器正将共享服务器状态用于资源锁定**。请参见“-ss 选项”一节第 90 页。
3. 如果要对服务器启动的同步使用通告程序，请使用 -lsc 选项以指定本地服务器连接设置。这些设置传递到群中的其它服务器上，以使它们可以相互连接以共享通知的处理。例如，如果在主机 **farm_host22** 上运行：

   ```
   mlsrv11 -x tcpip(port=3245) -zs -ss server5 -lsc
   tcpip(host=farm_host22;port=3245) -c ...
   ```

   如果未使用通告程序，则不需要 -lsc 选项。

另请参见

- “-lsc 选项”一节第 68 页
- “-zs 选项”一节第 112 页
- “-ss 选项”一节第 90 页

MobiLink 服务器启动故障排除

本部分介绍启动 MobiLink 服务器时的一些常见问题。

确保网络通信软件处于运行状态

运行 MobiLink 服务器之前，必须安装适当的网络通信软件并使之处于运行状态。如果您运行的是可靠的网络软件并且只安装了一个网络，要做到这一点应该会比较简单。运行 MobiLink 服务器之前，应确认其它需要网络通信的软件是否工作正常。

如果是在 TCP/IP 协议下运行，则可能需要确认 ping 和 telnet 的工作是否正常。许多 TCP/IP 协议栈都随附有 ping 和 telnet 应用程序。

调试网络通信启动问题

如果在网络上建立连接时遇到问题，则在客户端和服务器均可以使用调试选项来诊断问题。启动信息显示在服务器窗口中：可以使用 -o 选项将结果记录到输出文件中。

请参见“记录 MobiLink 服务器操作”一节第 31 页。

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

MobiLink 服务器选项

目录

mlsrv11 语法 41
@data 选项 46
-a 选项 47
-b 选项 48
-bn 选项 49
-c 选项 50
-cm 选项 51
-cn 选项 52
-cr 选项 53
-cs 选项 54
-ct 选项 55
-dl 选项 56
-dr 选项 57
-ds 选项 58
-dsd 选项 59
-dt 选项 60
-e 选项 61
-esu 选项 62
-et 选项 63
-f 选项 64
-fips 选项 65
-fr 选项 66
-ftr 选项 67
-lsc 选项 68
-m 选项 69
-nba 选项 70
-nc 选项 71
-notifier 选项 72
-o 选项 73
-on 选项 74
-oq 选项 75

-os 选项 76
-ot 选项 77
-ppv 选项 78
-q 选项 82
-r 选项 83
-rd 选项 84
-s 选项 85
-sl dnet 选项 86
-sl java 选项 87
-sm 选项 89
-ss 选项 90
-tc 选项 91
-tf 选项 92
-tx 选项 93
-ud 选项 94
-ui 选项 95
-ux 选项 96
-v 选项 97
-w 选项 100
-wu 选项 101
-x 选项 102
-xo 选项 107
-zp 选项 111
-zs 选项 112
-zt 选项 113
-zu 选项 114
-zus 选项 115
-zw 选项 116
-zwd 选项 117
-zwe 选项 118

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

mlsrv11 语法

功能

启动 MobiLink 服务器。

语法

mlsrv11 -c "*connection-string***"** [*options*]

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 46 页。
-a	禁用出现同步错误时的自动重新连接。请参见“-a 选项”一节第 47 页。
-b	修整字符串的空格填充。请参见“-b 选项”一节第 48 页。
-bn *size*	指定比较 BLOB 以进行冲突检测时要考虑的最大字节数。请参见“-bn 选项”一节第 49 页。
-c *"keyword=value; ..."*	为统一数据库提供 ODBC 数据库连接参数。请参见“-c 选项”一节第 50 页。
-cm *size*	指定服务器内存高速缓存大小。请参见“-cm 选项”一节第 51 页。
-cn *connections*	设置与统一数据库服务器的最大同时连接数。请参见“-cn 选项”一节第 52 页。
-cr *count*	设置数据库连接的最大重试次数。请参见“-cr 选项”一节第 53 页。
-cs *"keyword=value; ..."*	为 MobiLink 系统数据库 (MLSD) 提供系统数据库连接参数。
-ct *connection-timeout*	设置一个连接在闲置多长时间后超时。请参见“-ct 选项”一节第 55 页。
-dl	在 MobiLink 服务器消息窗口显示所有日志消息。请参见“-dl 选项”一节第 56 页。
-dr	仅适用于 Adaptive Server Enterprise。请确保参与同步的表没有使用 DataRow 锁定模式。请参见“-dr 选项”一节第 57 页。

选项	说明
-ds *size*	指定可用于存储所有可重新启动的下载的最大数据量。请参见“-ds 选项”一节第 58 页。
-dsd	禁用快照隔离，其为 SQL Anywhere 和 Microsoft SQL Server 统一数据库的缺省下载隔离级别。请参见“-dsd 选项”一节第 59 页。
-dt	仅检测当前数据库中的事务。请参见“-dt 选项”一节第 60 页。
-e *filename*	将发送的远程错误日志存入指定文件。请参见“-e 选项”一节第 61 页。
-esu	对上载使用快照隔离。请参见“-esu 选项”一节第 62 页。
-et *filename*	将发送的远程错误日志存入指定文件，但如果文件存在，则会首先删除其内容。请参见“-et 选项”一节第 63 页。
-f	假设同步脚本没有发生变化。请参见“-f 选项”一节第 64 页。
-fips	强制所有安全 MobiLink 流符合 FIPS 标准。请参见“-fips 选项”一节第 65 页。
-fr	如果缺少表数据脚本，同步不会终止而只是发出一个警告。请参见“-fr 选项”一节第 66 页。
-ftr *path*	创建一个文件（mlfiletransfer 实用程序将使用这些文件）位置。请参见“-ftr 选项”一节第 67 页。
-lsc *protocol*[*protocol-options*	指定本地服务器连接信息。请参见“-lsc 选项”一节第 68 页。
-m [*filename*]	启用 QAnywhere 消息传递。请参见“-m 选项”一节第 69 页。
-nba { + | - }	设置服务器操作的下载确认模式。请参见“-nba 选项”一节第 70 页。
-nc *connections*	设置并发连接的最大数目。请参见“-nc 选项”一节第 71 页。
-notifier	为服务器启动的同步启动通告程序。请参见“-notifier 选项”一节第 72 页。

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

选项	说明
-o *logfile*	将消息记录到一个文件。请参见“-o 选项”一节第 73 页。
-on *size*	设置日志文件的最大大小。请参见“-on 选项”一节第 74 页。
-oq	出现启动错误时阻止窗口弹出。请参见“-oq 选项”一节第 75 页。
-os *size*	输出文件大小的最大值。请参见“-os 选项”一节第 76 页。
-ot *logfile*	将消息记录到一个文件，但首先会删除文件的内容。请参见“-ot 选项”一节第 77 页。
-q	最小化 MobiLink 服务器消息窗口。请参见“-q 选项”一节第 82 页。
-r *retries*	重试被死锁的上载的最大次数。请参见“-r 选项”一节第 83 页。
-rd *delay*	设置重试被死锁的事务前的最大延迟时间（以秒为单位）。请参见“-rd 选项”一节第 84 页。
-s *count*	指定一次读取或发送的最大行数。请参见“-s 选项”一节第 85 页。
-sl dnet *script-options*	设置 NET CLR 选项并在启动时强制装载虚拟机。请参见“-sl dnet 选项”一节第 86 页。
-sl java *script-options*	设置 Java 虚拟机选项并在启动时强制装载虚拟机。请参见“-sl java 选项”一节第 87 页。
-sm *number*	设置可活动处理的最大同步数。请参见“-sm 选项”一节第 89 页。
-ss	使服务器进入共享服务器状态模式。请参见“-ss 选项”一节第 90 页。
-tc *minutes*	为 SQL 脚本的执行设置倒数计时器。请参见“-tc 选项”一节第 91 页。
-tf	当倒数计时器到期时，SQL 脚本执行失败（不适用于 Oracle）。请参见“-tf 选项”一节第 92 页。

选项	说明
-tx *count*	用于事务性上载，对事务组进行批处理，然后将它们一起提交。请参见“-tx 选项”一节第 93 页。
-ud	在 Unix 平台上作为守护程序运行。请参见“-ud 选项”一节第 94 页。
-ui	适用于支持 X 窗口的 Linux，如果没有提供可用的显示则以 shell 模式启动 MobiLink 服务器。请参见“-ui 选项”一节第 95 页。
-ux	打开 MobiLink 服务器消息窗口。请参见“-ux 选项”一节第 96 页。
-v [*levels*]	控制写入日志文件的消息类型。请参见“-v 选项”一节第 97 页。
-w *count*	设置数据库工作线程的数目。请参见“-w 选项”一节第 100 页。
-wu *count*	设置可以同时处理上载的最大数据库工作线程数目。请参见“-wu 选项”一节第 101 页。
-x *protocol*[**(***network-parameters***)**]	指定通信协议。还可以指定网络参数，形式为 *parameter=value*，多个参数间用分号隔开。请参见“-x 选项”一节第 102 页。
-xo *protocol*[**(***network-parameters***)**]	为版本 8 和 9 客户端指定通信协议。还可以指定网络参数，形式为 *parameter=value*，多个参数间用分号隔开。请参见“-xo 选项”一节第 107 页。
-zp	如果统一数据库和远程数据库发生时间戳冲突，利用此选项可将精度比最低精度高的时间戳值用于检测冲突。请参见“-zp 选项”一节第 111 页。
-zs *name*	指定一个服务器名。请参见“-zs 选项”一节第 112 页。
-zt *number*	指定用来运行 MobiLink 服务器的处理器的最大数量。请参见“-zt 选项”一节第 113 页。
-zu { + | - }	在未定义 authenticate_user 脚本时，控制用户的自动添加。请参见“-zu 选项”一节第 114 页。
-zus	使 MobiLink 调用不存在上载的表的上载脚本。请参见“-zus 选项”一节第 115 页。

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

选项	说明
-zw 1,...5	控制要显示的警告消息级别。请参见“-zw 选项”一节 第 116 页。
-zwd *code*	禁用特定的警告代码。请参见“-zwd 选项”一节 第 117 页。
-zwe *code*	启用特定的警告代码。请参见“-zwe 选项”一节 第 118 页。

说明

MobiLink 服务器通过 ODBC 打开与统一数据库服务器之间的连接。然后就可以接受来自客户端应用程序的连接，并控制同步过程。

必须使用 -c 选项为统一数据库提供连接参数。命令行选项可按任意顺序指定。c 选项在此处作为第一项出现在命令字符串中，这仅仅是个约定。它可以出现在选项列表的任意位置，但必须位于连接字符串之前。

除非已将 ODBC 数据源配置为自动启动统一数据库，否则在启动 MobiLink 服务器之前请确保该数据库处于运行状态。

另请参见

- “MobiLink 服务器”第 27 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

mlsrv11 -c "*connection-string***" @***data* ...

注释

使用此选项从指定的环境变量或配置文件中读入 mlsrv11 命令行选项。如果同时存在名称相同的环境变量和配置文件，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》。

Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

-a 选项

指示 MobiLink 服务器在发生同步错误时不重新进行连接。

语法

mlsrv11 -c "*connection-string***" -a** ...

注释

如果同步过程中出现错误，MobiLink 服务器将自动断开与统一服务器的连接，然后重新建立连接。重新连接可以确保随后的同步从已知状态开始。如果不需要进行重新连接，则您可以使用此命令行选项禁用该操作。状态信息的维护取决于程序员的要求，并可随程序员配置与 DBMS 一起工作的 MobiLink 脚本的方式而变化。即使所用数据库是 Oracle、SQL Anywhere 数据库或其它受支持的产品，情况也是如此。某些状态信息可能需要根据客户端应用程序而重新进行初始化。

Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

-b 选项

对于 VARCHAR、CHAR、LONG VARCHAR 或 LONG CHAR 类型的列，请在同步过程中删除字符串的尾随空白。

语法

mlsrv11 -c "*connection-string***" -b** ...

注释

注意
建议在统一数据库中使用 VARCHAR 而不是 CHAR，这样就不会出现此问题。

此选项帮助消除 SQL Anywhere CHAR 数据类型和统一数据库所使用的 CHAR 或 VARCHAR 数据类型之间的差异。SQL Anywhere CHAR 数据类型等效于 VARCHAR。但是，在大多数非 SQL Anywhere 统一数据库中，CHAR(n) 数据类型会填充空白以达到 n 个字符。

指定 -b 选项后，如果远程数据库上的列是一个字符串，那么对于 CHAR、VARCHAR、LONG CHAR 或 LONG VARCHAR 类型的列，MobiLink 服务器将删除字符串的尾随空白。此操作在过滤当前同步过程中上载的行之前进行。随后将剪裁的数据下载到远程数据库中。

此选项也可用来检测冲突的更新。对于每个上载更新行，MobiLink 服务器会针对给定的主键从统一数据库中读取行，将该行与更新的前映像进行比较，然后确定更新是否为冲突的更新。如果使用 -b，MobiLink 会在进行比较之前从 CHAR、VARCHAR、LONG CHAR 或 LONG VARCHAR 类型的列中剪裁尾随空白。

另请参见

- “CHAR 列”一节第 8 页
- “NVARCHAR 数据类型”一节 《SQL Anywhere 服务器 - SQL 参考》

示例

如果不使用 -b 选项，那么从 SQL Anywhere 或 UltraLite 远程数据库上载到统一数据库的 CHAR(10) 列中的 'abc' 主键值将变成后面带有 7 个空格的 'abc'。如果下载了同一行，则它在远程数据库中显示为后面带有 7 个空格的 'abc'。远程数据库在未用空白填充的情况下包含两行：后面都带有 7 个空格的两个 'abc' 行。此时，远程数据库中有一个重复的行。

如果使用 -b 选项，那么从 SQL Anywhere 或 UltraLite 远程数据库上载到统一数据库的 CHAR(10) 列中的 'abc' 主键值将变成后面带有 7 个空格的 'abc'。7 个空格仍会将该值填充至 10 个字符，但如果下载了同一行，MobiLink 服务器会去掉尾随空白，该值在远程数据库中显示为 'abc'。-b 选项可以修正重复行的问题。

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

-bn 选项

设置要在冲突检测过程中比较的 BLOB 最大字节数。

语法

mlsrv11 -c "*connection-string***" -bn** *size* ...

注释

当两个 BLOB 包含相似或相同值时，由于涉及的数据量较大，因此为进行过滤或冲突检测而进行的比较操作将十分耗费资源。该选项将通知 MobiLink 服务器在进行比较时仅考虑两个 BLOB 中第一个 *size* 字节的内容。缺省情况是整个比较两个 BLOB。

在某些情况下，限制比较数据的最大值可以显著地加快同步过程的速度；但也可能会导致错误。例如，如果两个大的 BLOB 仅在最后几个字节有区别，那么 MobiLink 服务器可能认为它们相同，而事实上它们并不相同。

Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

-c 选项

为统一数据库指定连接参数。

语法

mlsrv11 -c "*connection-string***"** ...

注释

连接字符串必须为 MobiLink 服务器连接到统一数据库提供足够的信息。连接字符串是必需的。

连接字符串必须指定连接参数，形式为 *keyword=value*，以分号隔开，参数与参数间不留空格。

如果命令行中没有指定连接参数，则 ODBC 数据源说明中必须包含连接参数。检查 RDBMS 和 ODBC 数据源以确定所需的连接数据。

有关 SQL Anywhere 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》。

有关如何隐藏口令的信息，请参见“文件隐藏实用程序 (dbfhide)”一节 《SQL Anywhere 服务器 - 数据库管理》。

示例

```
mlsrv11 -c "dsn=odbcname;uid=DBA;pwd=sql"
```

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

-cm 选项

设置服务器内存高速缓存的最大大小。

语法

mlsrv11 -c "*connection-string***" -cm** *size*[**k** | **m** | **g**] ...

注释

服务器用于保存表数据、网络缓冲区、缓存的下载数据及用于同步的其它结构的最大内存量。如果服务器拥有的数据超过此内存池可保存的数据，则会在磁盘上存储数据。

size 为需要保留的内存量（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。缺省值为 **50M**。

-cn 选项

设置统一数据库同时连接的最大连接数目。

语法

mlsrv11 -c "*connection-string***" -cn** *value* ...

注释

指定 MobiLink 服务器与统一数据库之间可同时建立的最大连接数目。最小数目及缺省值比数据库工作线程数目多一个。如果提供的数目过小，系统将发出警告。

一个 MobiLink 数据库连接仅用于使用一个脚本版本的同步。当 MobiLink 服务器正在使用 -cn 选项所允许的所有数据库连接时，如果某个同步正在等待执行，但目前不存在用于其脚本版本的数据库连接，则 MobiLink 服务器会断开一个连接，然后为待执行同步的脚本版本创建一个新的数据库连接。

使用大于数据库工作线程数目的值可以提高性能，尤其是在统一数据库的连接速度较慢或正在使用多个脚本版本的情况下更是如此。合适的数据库连接数目最大值为脚本版本的数目与数据库工作线程数目的乘积再加一。如果连接数目大于此最佳值，则不一定会提高同步速度，并对 MobiLink 服务器和统一数据库的资源造成不必要的消耗。

另请参见

- “-w 选项”一节第 100 页

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

-cr 选项

设置数据库连接的最大重试次数。

语法

mlsrv11 -c "*connection-string***" -cr** *value* ...

注释

设置连接失败时 MobiLink 服务器退出之前尝试连接到数据库的最大次数。缺省值为三次连接重试。

-cs 选项

指定 MobiLink 系统数据库 (MLSD) 的连接参数。

语法

mlsrv11 -c "*connection-string***" -cs "***connection-string***"** ...

注释

可将 MobiLink 服务器系统对象（例如系统表、过程、触发器和视图）存储在统一数据库以外的其它数据库中。存储 MobiLink 系统对象的数据库被称为 MLSD。

当在命令行指定了此命令选项时，MobiLink 服务器会连接到 MLSD，并在保持同步状态的同时读取用户定义的脚本，如 ML 用户名、远程 ID、进度偏移、上次上载和下载的时间戳等等。MobiLink 服务器使用初始 -c 命令行选项连接到统一数据库，并使用这些连接从客户端数据库上载数据，同时将数据下载到客户端数据库。统一数据库中无需包括任何 MobiLink 服务器系统对象，所有用户定义的脚本（包括错误报告和错误处理脚本）可从 MLSD 中读取然后在统一数据库中执行。

使用此选项时，MobiLink 服务器会使用 Microsoft 分布式事务协调器 (MSDTC)。

统一数据库和 MLSD 可以是任何一种受支持的 MobiLink 统一数据库。但是，相应的 ODBC 驱动程序必须支持 Microsoft 分布式事务。

统一数据库和 MLSD 必须要有事务日志才能使用 MSDTC。

此选项只能在 Windows 操作系统上使用。

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

-ct 选项

设置连接闲置多久（以分钟为单位）之后将超时并被 MobiLink 服务器断开。

语法

mlsrv11 -c "*connection-string***" -ct** *connection-timeout* ...

注释

服务器将在 MobiLink 数据库连接闲置了指定时间段之后将其释放。此超时时间可以通过 -ct 选项来设置。缺省超时时间为 60 分钟。

-dl 选项

在屏幕上显示所有 MobiLink 服务器消息。

语法

mlsrv11 -c "*connection-string***" -v -dl** ...

注释

在 MobiLink 服务器消息窗口显示所有 MobiLink 服务器消息。缺省情况下，如果 MobiLink 服务器消息日志文件将被输出（使用 -o 选项），则窗口中只显示部分消息。如果有大量消息，则使用该选项可能降低性能。

另请参见

- “-o 选项”一节第 73 页
- “将数据库服务器消息记录到文件”一节 《SQL Anywhere 服务器 - 数据库管理》

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

-dr 选项

仅适用于 Adaptive Server Enterprise。请确保参与同步的表没有使用 DataRow 锁定模式。

语法

mlsrv11 -c "*connection-string***" -dr** ...

注释

只有在所有的同步表都不是通过使用 DataRow 锁定模式创建时，才应该使用此选项。

使用此选项可减少 MobiLink 服务器所发送的重复数据。

另请参见

- “MobiLink 隔离级别”一节第 156 页

-ds 选项

用于可重新启动的下载。指定 MobiLink 服务器可用于存储所有可重新启动下载的最大数据量。

语法

mlsrv11 -c "*connection-string***" -ds** *size*[**k** | **m** | **g**] ...

注释

MobiLink 服务器保存客户端尚未接收的下载数据以用于可重新启动的下载。此选项限制服务器为所有组合的同步所保存的数据量。

如果 *size* 太小，服务器可能会释放下载数据，造成无法重新启动下载。服务器仅在发生以下情况之一时才释放下载数据：

- 用户成功完成下载。
- 用户通过新的同步请求进行恢复，但未启用下载恢复。
- 进来的请求需要使用高速缓存。此时将优先清除最早未成功的下载。

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。缺省值为 **10m**。

另请参见

- “恢复失败的下载”一节第 149 页
- “-dc 选项”一节 《MobiLink - 客户端管理》

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

-dsd 选项

禁用快照隔离。

语法

mlsrv11 -c "*connection-string***" -dsd** ...

注释

当统一数据库为 SQL Anywhere（版本 10 或更高版本）或 Microsoft SQL Server（2005 或更高版本）时，下载的缺省隔离级别为快照隔离。如果统一数据库为这些数据库的早期版本，则缺省下载隔离级别为读取已提交的。

您也可以在脚本中更改缺省隔离级别。但对于 SQL Anywhere 版本 10 和 Microsoft SQL Server 2005 及更高版本的数据库，是在上载和下载事务开始时设置隔离级别。这意味着如果在 begin_connection 脚本中设置隔离级别，它可能在 begin_upload 和 begin_download 脚本中被覆盖。

此选项仅适用于 SQL Anywhere 版本 10 和 Microsoft SQL Server 2005 统一数据库。

另请参见

- “MobiLink 隔离级别”一节第 156 页
- “-dt 选项”一节第 60 页
- “-esu 选项”一节第 62 页

-dt 选项

仅适用于 Microsoft SQL Server 和 Adaptive Server Enterpris 数据库。使 MobiLink 仅检测当前数据库中的事务。

语法

mlsrv11 -c "*connection-string***" -dt** ...

注释

此选项使 MobiLink 忽略除当前数据库中的事务之外的所有事务。这可增加吞吐量并减少行的重复下载。

此选项在以下情况使用：

- 统一数据库正在其上还运行了其它数据库的 Microsoft SQL Server 或 Adaptive Server Enterprise 上运行。
- 在 Microsoft SQL Server 中使用了上载或下载的快照隔离。
- 与 Adaptive Server Enterprise 同步表时使用了 DataRow 锁定模式。
- 上载或下载脚本没有访问服务器上的任何其它数据库。

此选项仅适用于使用快照隔离的 Microsoft SQL Server 数据库，和对同步中涉及的表使用 DataRow 锁定模式的 Adaptive Server Enterprise 数据库。

另请参见

- “MobiLink 隔离级别”一节第 156 页
- “-dsd 选项”一节第 59 页
- “-esu 选项”一节第 62 页

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

-e 选项

存储从 SQL Anywhere MobiLink 客户端发送的错误日志。

语法

mlsrv11 -c "*connection-string***" -e** *filename* ...

注释

如果不使用 -e 选项，来自 SQL Anywhere MobiLink 客户端的错误日志会存储到名为 *mlsrv11.mle* 的文件中。而 -e 选项指示 MobiLink 服务器将错误日志存储到指定文件。缺省情况下，在远程站点出现错误时，dbmlsync 会将多达 32 千字节的远程日志消息发送到 MobiLink 服务器。

此选项提供对远程错误日志的集中访问，以帮助诊断同步问题。

从远程站点传送的信息量可由 dbmlsync 扩展选项 ErrorLogSendLimit 来控制。

另请参见

- “-et 选项”一节第 63 页
- “ErrorLogSendLimit (el) 扩展选项”一节 《MobiLink - 客户端管理》

-esu 选项

对上载使用快照隔离。

语法

mlsrv11 -c "*connection-string***" -esu** ...

注释

缺省情况下，MobiLink 对上载使用 SQL_TXN_READ_COMMITTED 隔离级别。多数情况下，这是最佳隔离级别。

如果对上载使用快照隔离，上载更新期间可能产生快照事务冲突。如果发生此种情况，MobiLink 服务器将回滚整个上载并重新尝试上载。在这种情况下，您可能要调整 MobiLink 服务器选项 -r 或 -rd 的设置，以指定重试之间的延迟时间及最大重试次数。

您可以在脚本中更改缺省隔离级别。要更改上载隔离级别，通常使用 begin_upload 脚本。

此选项仅适用于 SQL Anywhere 版本 10 和 Microsoft SQL Server 2005 统一数据库。

另请参见

- “MobiLink 隔离级别”一节第 156 页
- “-dsd 选项”一节第 59 页
- “-dt 选项”一节第 60 页
- “-r 选项”一节第 83 页
- “-rd 选项”一节第 84 页

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

-et 选项

在截断现有文件后，将自 SQL Anywhere MobiLink 客户端发送的错误日志存储到指定的文件中。

语法

mlsrv11 -c "*connection-string***" -et** *filename* ...

注释

-et 选项与 -e 选项功能类似，区别在于前者在新错误添加到错误日志文件之前截断该文件。

从远程站点传送的信息量可由 dbmlsync 扩展选项 ErrorLogSendLimit 来控制。

另请参见

- “ErrorLogSendLimit (el) 扩展选项”一节 《MobiLink - 客户端管理》
- “-e 选项”一节第 61 页

Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

-f 选项

仅装载同步脚本一次，从而可提高性能。

语法

mlsrv11 -c "*connection-string***" -f** ...

注释

如果不使用 -f 选项，MobiLink 服务器将在常规操作过程中检查同步脚本是否发生了变化。这样的检查在开发过程中是有益的，但在生产环境中进行这样的检查将造成不必要的性能损失。如果使用 -f 选项，MobiLink 服务器仅将为每个 MobiLink 会话装载一次同步脚本。

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

-fips 选项

强制所有安全 MobiLink 流符合 FIPS 标准。

语法

mlsrv11 -c *connection-string*" **-fips** ...

注释

如果指定此选项，则强制所有 MobiLink 加密均使用 FIPS 认可的算法。当指定了 -fips 选项时，您仍然可以使用未加密的连接，但不能使用简单加密。

使用此选项时，无论是否指定 FIPS 认可的算法，都将对连接使用此类算法。例如，如果启动 MobiLink 服务器时使用选项 -fips 和选项 -x tls(...;fips=no;...)，将忽略 fips=no 设置，并使用 fips=yes 启动服务器。

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》。

对于 MobiLink 传送层安全性，-fips 选项会使服务器使用 FIPS 认可的 RSA 加密编码器（即使指定的 RSA 未经 FIPS 认可）。如果指定了 ECC，则会因 FIPS 认可的椭圆曲线算法不可用而发生错误。

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》
- “FIPS 认可的加密技术”一节 《SQL Anywhere 服务器 - 数据库管理》

-fr 选项

如果缺少所需的表数据脚本，同步不会中止而只是发出一个警告。

语法

mlsrv11 -c "*connection-string***" -fr** ...

注释

缺省情况下，如果缺少所需的同步脚本，MobiLink 服务器将中止。此选项会导致 MobiLink 发出警告而非中止。

另请参见

- “必需的脚本”一节第 305 页
- “仅上载同步和仅下载同步”一节第 130 页
- “同步事件”第 319 页
- “直接行处理”第 609 页

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

-ftr 选项

创建一个文件（mlfiletransfer 实用程序将使用这些文件）位置。

语法

mlsrv11 -c "*connection-string***" -ftr** *path ...*

注释

此选项设置文件传输根目录。要传输给用户的文件可放置在根目录中，或放在带有用户名的子目录中。MobiLink 首先在文件传输根目录下具有连接客户端用户名的子目录中查找请求的文件。如果文件不在此子目录中，MobiLink 会在文件传输根目录下查找。

如果要使用 mlfiletransfer 实用程序，该选项是必需的。

另请参见

- “MobiLink 文件传输实用程序 (mlfiletransfer)”一节 《MobiLink - 客户端管理》
- “authenticate_file_transfer 连接事件”一节第 331 页

Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

-lsc 选项

指定本地服务器连接信息。会将此信息传递给服务器群中的其它服务器。

语法

mlsrv11 -c "*connection-string***" -lsc** *protocol*[*protocol-options*] ...

protocol : **tcpip** | **tls** | **http** | **https**

protocol-options : **(** *option***=***value*; ... **)**

注释

仅在 MobiLink 服务器群中运行通告程序时才需要使用此选项。当其它服务器需要连接到本地 MobiLink 服务器时将此信息传递给它们。

例如，如果有一个正运行于名为 server_rack10 主机上的服务器，则命令行可为：

```
mlsrv11  -x tcpip(port=200) -zs -ss server5 -lsc
tcpip(host=server_rack10;port=200) -c ...
```

在此示例中，另一个服务器将使用统一数据库中的共享状态来获取连接字符串 `tcpip(host=server_rack10;port=200)`，然后使用这个连接字符串连接到刚才启动的服务器。

另请参见

- “在服务器群中运行 MobiLink 服务器”一节第 37 页
- “-zs 选项”一节第 112 页
- “-ss 选项”一节第 90 页
- “MobiLink 服务器群中的通告程序”一节 《MobiLink - 服务器启动的同步》

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

-m 选项

启用 QAnywhere 消息传递。

语法

mlsrv11 -c "*connection-string***" -m** [*message-properties-file*] ...

注释

不建议使用可选的 *message-properties-file*。现在属性通过 Sybase Central 指定并存储在数据库中，或使用服务器管理请求来指定。

在 *message-properties-file* 中，每个属性必须单占一行，并且包含一个属性名、一个 = 符号以及一个属性值。

有关可设置属性的列表，请参见“服务器属性”一节 《QAnywhere》。

另请参见

- “QAnywhere 技术简介” 《QAnywhere》
- “启动启用了 MobiLink 的 QAnywhere”一节 《QAnywhere》
- “服务器管理请求” 《QAnywhere》

示例

要在使用示例服务器消息存储库 (*samples-dir\QAnywhere\server\qanyserv.db*) 时启动 QAnywhere 消息传递，请运行以下命令：

```
mlsrv11 -m -c "dsn=QAnywhere 10 Demo"
```

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》。

-nba 选项

设置服务器操作的下载确认模式。

语法

mlsrv11 -c "*connection-string***" -nba{ + | - }** ...

注释

如果指定了下载确认，可以选择两种模式：非阻塞（缺省模式，通过 -nba+ 来设置）或阻塞（通过 -nba- 来设置）。现已不建议使用阻塞下载确认 (-nba-)。应尽量使用非阻塞下载确认。

推荐使用非阻塞下载确认，因为它比阻塞下载确认有显著的性能优势。但是，以下情况下不能使用非阻塞下载确认：

- 版本 10.0.0 以前的客户端不支持非阻塞下载确认。

如果使用 mlsrv11 -m 选项启用了 QAnywhere 消息传递，则无法使用阻塞下载确认。无法同时指定 -m 和 -nba-。

另请参见

- dbmlsync： “SendDownloadACK (sa) 扩展选项”一节 《MobiLink - 客户端管理》
- UltraLite： “发送下载确认同步参数”一节 《UltraLite - 数据库管理和参考》
- “nonblocking_download_ack 连接事件”一节第 438 页
- “publication_nonblocking_download_ack 连接事件”一节第 442 页

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

-nc 选项

设置并发网络连接的最大数目。

语法

mlsrv11 -c "*connection-string***" -nc** *connections* ...

注释

达到该限制量时，MobiLink 服务器会拒绝新的同步连接。在客户端，使用系统错误代码发出的通信错误表示连接已遭到拒绝。

缺省值为 1024。

要限制非持久性 HTTP/HTTPS 的并发同步的数量，请将 -nc 设置为远高于 -sm。达到 -sm 限制时，MobiLink 服务器将向远程客户端提供一个 HTTP 错误 503（服务不可用）。但是，如果达到 -nc 限制，将会发出套接字错误。-nc 和 -sm 之间的差值越大，拒绝的连接生成 HTTP 503 错误的可能性就越大，而描述性套接字错误的可能性则会越小。例如，可将 -sm 设置为 100，而将 -nc 设置为 1000。请参见“-sm 选项”一节第 89 页。

Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

-notifier 选项

为服务器启动的同步启动通告程序。

语法

mlsrv11 -c "*connection-string***" -notifier** [*notifier-properties-file*] ...

注释

如果指定通告程序的配置文件名，或者不指定文件名但拥有名为 *config.notifier* 的缺省通告程序属性文件，那么将使用该文件对通告程序进行配置。这会替换存储在统一数据库的 ml_properties 表中的任何配置信息。

否则，MobiLink 将使用存储在统一数据库的 ml_properties 表中的配置信息。

在使用 -notifier 选项时，将启动已启用的每个通告程序。

有关启用通告程序的详细信息，请参见“通告程序属性”一节 《MobiLink - 服务器启动的同步》。

另请参见

- “用于服务器启动的同步的 MobiLink 服务器设置” 《MobiLink - 服务器启动的同步》
- “使用通告程序配置文件配置服务器端设置”一节 《MobiLink - 服务器启动的同步》
- “通告程序”一节 《MobiLink - 服务器启动的同步》
- “MobiLink 服务器群中的通告程序”一节 《MobiLink - 服务器启动的同步》

Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

-o 选项

将输出消息记录到 MobiLink 服务器消息日志文件，并限制记录到 MobiLink 服务器消息窗口的数据。

语法

mlsrv11 -c "*connection-string***" -o** *logfile ...*

注释

将所有日志消息写入指定文件。请注意 MobiLink 服务器窗口（如果出现）通常只显示记录的部分消息。

在同步期间发生错误时，MobiLink 服务器将在其输出文件中给出完整的错误上下文。错误上下文可能包括以下信息：

- **用户名** 这是由 MobiLink SQL Anywhere 应用程序在同步过程中提供的实际用户名。
- **修改过的用户名** 这就是 modify_user 脚本修改过的用户名。
- **事务** 它会列出出现错误的事务。事务可以是 authenticate_user、begin_synchronization、upload、prepare_for_download、download 或 end_synchronization。
- **表名** 显示表名称（如果存在）或为空。
- **行操作** 操作可以是 INSERT、UPDATE、DELETE 或 FETCH。
- **行数据** 显示引起错误的行的所有列值。
- **脚本版本** 当前用于同步的脚本版本。
- **脚本** 这是导致错误的脚本。

无论选择哪一种详细级别，错误上下文信息都会出现在日志中。

另请参见

- “-os 选项”一节第 76 页
- “-dl 选项”一节第 56 页
- “-ot 选项”一节第 77 页
- “-on 选项”一节第 74 页
- “-v 选项”一节第 97 页

Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

-on 选项

指定 MobiLink 服务器消息日志文件的最大大小，达到该大小后，将用扩展名 .old 重命名该文件并起用一个新文件。

语法

mlsrv11 -c "*connection-string***" -on** *size* [**k** | **m**]...

注释

size 是消息日志文件大小的最大值，以字节为单位。使用后缀 k 或 m 分别指定以千字节或兆字节为单位。最小值限制为 10 KB。

日志文件达到指定大小后，MobiLink 服务器将用扩展名 .old 重命名输出文件，并以原始名称起用一个新文件。

> **注意**
> 如果 .old 文件已经存在，则将覆盖它。要避免丢失旧的日志文件，请使用 -os 选项。

此选项不能与 -os 选项一起使用。

另请参见

- “-o 选项”一节第 73 页
- “-ot 选项”一节第 77 页
- “-on 选项”一节第 74 页
- “-os 选项”一节第 76 页
- “-v 选项”一节第 97 页

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

-oq 选项

在 Windows 上，用于在发生启动错误时禁止出现错误窗口。

语法

mlsrv11 -c "*connection-string***" -oq** ...

注释

缺省情况下，MobiLink 服务器在发生启动错误时显示一个窗口。-oq 选项可禁止显示此窗口。

-os 选项

设置 MobiLink 服务器消息日志文件的最大大小，之后将创建并使用一个具有新名称的新日志文件。

语法

mlsrv11 -c "*connection-string***" -os** *size* [**k** | **m**] ...

注释

size 是记录输出消息的文件大小的最大值。缺省单位是字节。使用后缀 k 或 m 分别指定以千字节或兆字节为单位。最小值限制为 10 KB。

在 MobiLink 服务器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超过了指定大小，那么 MobiLink 服务器会将消息日志文件重命名为 *yymmddxx.mls*，其中 *xx* 是 00 到 99 之间的一个数字，*yymmdd* 代表当前的年、月、日。

可以使用该选项删除旧的消息日志文件，从而释放磁盘空间。最新输出总是附加到由 -o 或 -ot 指定的文件中。

不能将此选项与 -on 选项一起使用。

注意
此选项会使日志文件的数目没有限制。为避免此情况，请使用 -o 或 -on。

另请参见

- “-o 选项”一节第 73 页
- “-on 选项”一节第 74 页
- “-ot 选项”一节第 77 页
- “-v 选项”一节第 97 页

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

-ot 选项

将输出消息记录到 MobiLink 服务器消息日志文件中，但首先删除其内容。

语法

mlsrv11 -c "*connection-string***" -ot** *logfilename* ...

注释

缺省情况下这些消息将输出到屏幕。

另请参见

- “-on 选项”一节第 74 页
- “-os 选项”一节第 76 页
- “-v 选项”一节第 97 页
- “-o 选项”一节第 73 页

Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

-ppv 选项

使 MobiLink 根据指定的时间段打印新的定期监控值。时间段以秒为单位。

语法

mlsrv11 -c "*connection-string***" -ppv** *period ...*

注释

通过这些值可了解服务器的状态，而且它们还有助于确定 MobiLink 服务器的健康情况和性能。例如，一个人可以查看 DB_CONNECTIONS 和 LONGEST_DB_WAIT 值，来查找 -w 选项或同步脚本中存在的潜在问题。这些值还可以提供跟踪系统范围内吞吐量测量（例如，每秒钟上载或下载的行数以及每秒的成功同步数）的简便方法。

建议的时间段为 60 秒。

如果时间段过短，则日志将会增长得非常快。

输出的每一行的前缀均为 **PERIODIC:**，有助于搜索和过滤值。

打印的值可包括以下信息：

- **CMD_PROCESSOR_STAGE_LEN** 同步工作的队列长度。
- **CPU_USAGE** MobiLink 服务器所使用的 CPU 的时间（以微秒为单位）。
- **DB_CONNECTIONS** 正在使用的数据库连接的数量。
- **FREE_DISK_SPACE** 临时磁盘上的可用磁盘空间（以字节为单位）。
- **HEARTBEAT_STAGE_LEN** 定期、非同步工作的队列长度。
- **LONGEST_DB_WAIT** 活动同步等待数据库的最长时间。
- **LONGEST_SYNC** 最早同步的有效期限（以微秒为单位）。
- **MEMORY_USED** 正在使用的 RAM 字节数（仅限 Windows）。
- **ML_NUM_CONNECTED_CLIENTS** 已连接的同步客户端的数目。
- **NUM_COMMITS** 提交的总数。
- **NUM_CONNECTED_FILE_XFERS** 当前已连接的 mlfiletransfer 的数目。
- **NUM_CONNECTED_LISTENERS** 当前已连接的监听器的数目。
- **NUM_CONNECTED_MONITORS** 当前已连接的监控器的数目。
- **NUM_CONNECTED_PINGS** 当前已连接的强制回应客户端的数目。
- **NUM_CONNECTED_SYNCS** 当前已连接的数据同步的数目。
- **NUM_ERRORS** 错误的总数。
- **NUM_FAILED_SYNCS** 失败同步的总数。
- **NUM_IN_APPLY_UPLOAD** 当前处于应用上载阶段的同步数量。

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

- **NUM_IN_AUTH_USER** 当前处于验证用户阶段的同步数量。
- **NUM_IN_BEGIN_SYNC** 当前处于开始同步阶段的同步数量。
- **NUM_IN_CONNECT** 当前处于连接阶段的同步数量。
- **NUM_IN_CONNECT_FOR_ACK** 当前处于为下载肯定应答连接阶段的同步数量。
- **NUM_IN_END_SYNC** 当前处于结束同步阶段的同步数量。
- **NUM_IN_FETCH_DNLD** 当前处于读取下载阶段的同步数量。
- **NUM_IN_GET_DB_WORKER_FOR_ACK** 当前处于获取肯定应答的 DB 工作阶段的同步数量。
- **NUM_IN_NON_BLOCKING_ACK** 当前处于非阻塞下载肯定应答阶段的同步数量。
- **NUM_IN_PREP_FOR_DNLD** 当前处于准备下载阶段的同步数量。
- **NUM_IN_RECVING_UPLOAD** 当前处于接收上载阶段的同步数量。
- **NUM_IN_SEND_DNLD** 当前处于发送下载阶段的同步数量。
- **NUM_IN_SYNC_REQUEST** 当前处于同步请求阶段的同步数量。
- **NUM_IN_WAIT_FOR_DNLD_ACK** 当前处于等待下载肯定应答阶段的同步数量。
- **NUM_ROLLBACKS** 回退的总数。
- **NUM_ROWS_DOWNLOADED** 发送至远程数据库的总行数。
- **NUM_ROWS_UPLOADED** 从远程数据库接收的总行数。
- **NUM_SUCCESS_SYNCS** 成功同步的总数。
- **NUM_UNSUBMITTED_ERROR_RPTS** 未提交的错误报告的数目。
- **NUM_UPLOAD_CONNS_IN_USE** 当前正在使用的上载连接的数量。
- **NUM_WAITING_CONS** 当前正等待统一数据库的同步数量。
- **NUM_WARNINGS** 警告的总数。
- **PAGES_IN_STREAMSTACK** 网络流所持有的页数。
- **PAGES_LOCKED** 已装载到内存中的高速缓存页数。
- **PAGES_LOCKED_MAX** 高速缓存中的页数。这将使用 -cm 进行设置。请参见“-cm 选项”一节第 51 页。
- **PAGES_SWAPPED_IN** 曾经从磁盘读取的总页数。
- **PAGES_SWAPPED_OUT** 曾经与磁盘交换的总页数。
- **PAGES_USED** 所使用的高速缓存页数。它包括与磁盘交换的页数，因此可能会大于高速缓存大小。
- **RAW_TCP_STAGE_LEN** 网络工作队列的长度。
- **SERVER_IS_PRIMARY** 指出此服务器是主服务器还是次服务器。如果该服务器为主服务器则显示 1，否则显示 0。

- **STREAM_STAGE_LEN** 高级网络处理队列的长度。
- **TCP_BYTES_READ** 曾经读取的字节总数。
- **TCP_BYTES_WRITTEN** 曾经写入的字节总数。
- **TCP_CONNECTIONS** 当前已打开的 TCP 连接的数量。
- **TCP_CONNECTIONS_CLOSED** 曾经关闭的连接总数。
- **TCP_CONNECTIONS_OPENED** 曾经打开的连接总数。
- **TCP_CONNECTIONS_REJECTED** 曾经拒绝的连接总数。

示例

以下是一个显示定期监控值的示例输出。

```
I. 2008-10-14 10:34:43. <Main> PERIODIC: TCP_CONNECTIONS: 0
I. 2008-10-14 10:34:43. <Main> PERIODIC: PAGES_USED: 0
I. 2008-10-14 10:34:43. <Main> PERIODIC: PAGES_LOCKED: 0
I. 2008-10-14 10:34:43. <Main> PERIODIC: PAGES_LOCKED_MAX: 12692
I. 2008-10-14 10:34:43. <Main> PERIODIC: TCP_CONNECTIONS_OPENED: 0
I. 2008-10-14 10:34:43. <Main> PERIODIC: TCP_CONNECTIONS_CLOSED: 0
I. 2008-10-14 10:34:43. <Main> PERIODIC: TCP_CONNECTIONS_REJECTED: 0
I. 2008-10-14 10:34:43. <Main> PERIODIC: TCP_BYTES_READ: 0
I. 2008-10-14 10:34:43. <Main> PERIODIC: TCP_BYTES_WRITTEN: 0
I. 2008-10-14 10:34:43. <Main> PERIODIC: ML_NUM_CONNECTED_CLIENTS: 0
I. 2008-10-14 10:34:43. <Main> PERIODIC: PAGES_SWAPPED_OUT: 0
I. 2008-10-14 10:34:43. <Main> PERIODIC: PAGES_SWAPPED_IN: 0
I. 2008-10-14 10:34:43. <Main> PERIODIC: PAGES_IN_STREAMSTACK: 0
I. 2008-10-14 10:34:43. <Main> PERIODIC: CPU_USAGE: 468750
I. 2008-10-14 10:34:43. <Main> PERIODIC: NUM_COMMITS: 0
I. 2008-10-14 10:34:43. <Main> PERIODIC: NUM_ROLLBACKS: 0
I. 2008-10-14 10:34:43. <Main> PERIODIC: NUM_SUCCESS_SYNCS: 0
I. 2008-10-14 10:34:43. <Main> PERIODIC: NUM_FAILED_SYNCS: 0
I. 2008-10-14 10:34:43. <Main> PERIODIC: NUM_ERRORS: 0
I. 2008-10-14 10:34:43. <Main> PERIODIC: NUM_WARNINGS: 1
I. 2008-10-14 10:34:43. <Main> PERIODIC: DB_CONNECTIONS: 0
I. 2008-10-14 10:34:43. <Main> PERIODIC: RAW_TCP_STAGE_LEN: 0
I. 2008-10-14 10:34:43. <Main> PERIODIC: STREAM_STAGE_LEN: 0
I. 2008-10-14 10:34:43. <Main> PERIODIC: HEARTBEAT_STAGE_LEN: 0
I. 2008-10-14 10:34:43. <Main> PERIODIC: CMD_PROCESSOR_STAGE_LEN: 0
I. 2008-10-14 10:34:43. <Main> PERIODIC: NUM_ROWS_DOWNLOADED: 0
I. 2008-10-14 10:34:43. <Main> PERIODIC: NUM_ROWS_UPLOADED: 0
I. 2008-10-14 10:34:43. <Main> PERIODIC: FREE_DISK_SPACE: 162552295424
I. 2008-10-14 10:34:43. <Main> PERIODIC: LONGEST_DB_WAIT: 0
I. 2008-10-14 10:34:43. <Main> PERIODIC: LONGEST_SYNC: 0
I. 2008-10-14 10:34:43. <Main> PERIODIC: NUM_UNSUBMITTED_ERROR_RPTS: 247
I. 2008-10-14 10:34:43. <Main> PERIODIC: MEMORY_USED: 94375936
I. 2008-10-14 10:34:43. <Main> PERIODIC: SERVER_IS_PRIMARY: 1
I. 2008-10-14 10:34:43. <Main> PERIODIC: NUM_CONNECTED_SYNCS: 0
I. 2008-10-14 10:34:43. <Main> PERIODIC: NUM_CONNECTED_PINGS: 0
I. 2008-10-14 10:34:43. <Main> PERIODIC: NUM_CONNECTED_FILE_XFERS: 0
I. 2008-10-14 10:34:43. <Main> PERIODIC: NUM_CONNECTED_MONITORS: 0
I. 2008-10-14 10:34:43. <Main> PERIODIC: NUM_CONNECTED_LISTENERS: 0
I. 2008-10-14 10:34:43. <Main> PERIODIC: NUM_WAITING_CONS: 0
I. 2008-10-14 10:34:43. <Main> PERIODIC: NUM_IN_SYNC_REQUEST: 0
I. 2008-10-14 10:34:43. <Main> PERIODIC: NUM_IN_RECVING_UPLOAD: 0
I. 2008-10-14 10:34:43. <Main> PERIODIC: NUM_IN_CONNECT: 0
I. 2008-10-14 10:34:43. <Main> PERIODIC: NUM_IN_AUTH_USER: 0
I. 2008-10-14 10:34:43. <Main> PERIODIC: NUM_IN_BEGIN_SYNC: 0
I. 2008-10-14 10:34:43. <Main> PERIODIC: NUM_IN_APPLY_UPLOAD: 0
```


Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

```
I. 2008-10-14 10:34:43. <Main> PERIODIC: NUM_IN_PREP_FOR_DNLD: 0
I. 2008-10-14 10:34:43. <Main> PERIODIC: NUM_IN_FETCH_DNLD: 0
I. 2008-10-14 10:34:43. <Main> PERIODIC: NUM_IN_END_SYNC: 0
I. 2008-10-14 10:34:43. <Main> PERIODIC: NUM_IN_SEND_DNLD: 0
I. 2008-10-14 10:34:43. <Main> PERIODIC: NUM_IN_WAIT_FOR_DNLD_ACK: 0
I. 2008-10-14 10:34:43. <Main> PERIODIC: NUM_IN_GET_DB_WORKER_FOR_ACK: 0
I. 2008-10-14 10:34:43. <Main> PERIODIC: NUM_IN_CONNECT_FOR_ACK: 0
I. 2008-10-14 10:34:43. <Main> PERIODIC: NUM_IN_NON_BLOCKING_ACK: 0
I. 2008-10-14 10:34:43. <Main> PERIODIC: NUM_UPLOAD_CONNS_IN_USE: 0
```

-q 选项

指示 MobiLink 在启动时以最小化窗口运行。

语法

mlsrv11 -c "*connection-string***" -q** ...

注释

最小化 MobiLink 服务器窗口。

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

-r 选项

设置死锁重试的最大次数。

语法

mlsrv11 -c "*connection-string***" -r** *retries* ...

注释

缺省情况下，MobiLink 服务器最多重试 10 次被死锁的上载。由于不能保证克服死锁，所以如果死锁不能解除，同步将失败。该选项允许设置任意重试次数。指定 **-r 0** 将指示服务器停止重试被死锁的事务。该设置的最大限制为 2 的 32 次方减 1。

-rd 选项

设置死锁重试之间的最大延迟时间。

语法

mlsrv11 -c "*connection-string***" -rd** *delay* ...

注释

在上载事务出现死锁时，MobiLink 服务器在重试事务之前将等待一个随机的时间段。延迟的随机特性提高了重试成功的可能性。该选项将以秒为单位指定延迟时间的最大值。延迟时间设置为 0（零）将即刻执行重试，但推荐使用较大值，因为这可以提高重试的成功率。缺省（也为最大）延迟值为 **30**。

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

-s 选项

设置可同时上载的最大行数。

语法

mlsrv11 -c "*connection-string***" -s** *count* ...

注释

将可同时插入、更新或删除的行数的最大值设置为 *count*。

MobiLink 服务器通过 ODBC 驱动程序将上载行发送到统一数据库。此选项控制在各批次中发送到数据库服务器的行数。增加此值可加快上载流的处理速度并减少网络时间。但此值设置的越高，MobiLink 服务器可能需要越多的资源来应用上载流。

每次上载的行数可在日志文件的 **rowset size** 项中查看。

缺省值为 10。

-sl dnet 选项

设置 .NET 公共语言运行库 (CLR) 选项并强制在启动时装载 CLR。

语法

mlsrv11 -c "*connection-string***" -sl dnet** *options ...*

注释

设置直接传递给 .NET CLR 的选项。选项为：

选项	说明
-D*name=value*	设置环境变量。例如， `-Dsynchtype=far -Dextra_rows=yes` 有关详细信息，请参见 .NET framework 类系统环境。
-MLAutoLoadPath=*path*	设置基程序集的位置。仅适用于专用程序集。若要让 MobiLink 知道程序集的位置，请使用此选项或 -MLDomConfigFile，但不能两者都用。当您使用 -MLAutoLoadPath 时，您不能在事件脚本中指定域。缺省值为当前目录。
-MLDomConfigFile=*file*	设置基程序集的位置。适用于以下情况：您有共享程序集，或者不想装载目录中的所有程序集，或者由于某种其它原因而无法使用 MLAutoLoadPath。若要让 MobiLink 知道程序集的位置，请使用 -MLDomConfigFile 或 -MLAutoLoadPath，但不能两者都用。
-MLStartClasses= *classnames*	在服务器启动时，按列出的顺序装载并实例化用户定义的启动类。
-clrConGC	在 CLR 中启用并发垃圾回收。
-clrFlavor=(wks | svr)	要装载的 .NET CLR 的风格。服务器的风格是 **svr**，工作站的风格是 **wks**。缺省情况下装载 **wks**。
-clrVersion=*version*	要装载的 .NET CLR 版本。此版本前面必须加上前缀 **v**。例如，**v1.0.3705** 装载目录 *\Microsoft.NET\Framework\v1.0.3705*。

要显示此选项列表，请运行以下命令：

```
mlsrv11 -sl dnet (?)
```

另请参见

- “使用 .NET 编写同步脚本”第 551 页

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

-sl java 选项

设置 Java 虚拟机选项，并强制在启动时装载虚拟机。

语法

mlsrv11 -c "*connection-string***" -sl java (** *options* **)** ...

注释

设置 -jrepath 及其它要直接传递给 Java 虚拟机的选项。选项为：

选项	说明
-hotspot | **-server** | **-classic**	取代 Java VM 使用的缺省选择值。
-cp *location*;...	指定在其中搜索类的一组目录或 JAR 文件。也可以使用 -classpath 代替 -cp。
-D*name*=*value*	设置系统属性。例如， `-Dsynchtype=far -Dextra_rows=yes`
-DMLStartClasses=*classname*, ...	在服务器启动时，按列出的顺序装载并实例化用户定义的启动类。
-jrepath *path*	替换缺省 JRE 路径，即 *install-dir*\Sun\jre160_*chip* 目录（其中芯片可以是任何支持的芯片，例如 x86）。
-verbose [**:class** |**:gc** | **:jni**]	启用详细输出。
-X *vm-option*	按照 *install-dir*\Sun\jre160_*chip*\bin\client\Xusage.txt 文件中所述的方式来设置 VM 特定的选项（其中芯片可以是任何支持的芯片，例如 x86）。

要显示可以使用的 Java 选项列表，请键入：

```
java
```

Unix 注释

选项必须用括号括起来。括号可以是圆括号（如上面的语法所示）或大括号 { }。

-jrepath 选项只在 Windows 上可用。在 Unix 上，如果想要装载特定的 JRE，应当设置 LD_LIBRARY_PATH（在 AIX 上为 LIBPATH，在 HP-UX 上为 SHLIB_PATH），使之包括含有该 JRE 的目录。该目录必须列在所有 SQL Anywhere 安装目录之前。

在 Unix 上，-cp 选项必须用冒号隔开。

另请参见

- “使用 Java 语言编写同步脚本”第 493 页

示例

例如，在 Windows 上，下面的部分 mlsrv11 命令行设置启用系统声明的 Java 虚拟机选项：

```
mlsrv11 -sl java (-cp ;\myclasses; -esa) ...
```

在 Windows 上，下面的部分 mlsrv11 命令行定义了 LDAP_SERVER 系统属性：

```
mlsrv11 -sl java ( -cp ;\myclasses; -DLDAP_SERVER=huron-ldap ) ...
```

下面的部分 mlsrv11 命令行用于 Unix：

```
mlsrv11 -sl java { -cp .:$CLASSPATH:/opt/myclasses:/opt/my.jar: }
```


Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

-sm 选项

设置可活动处理的最大同步数，该设置同时还限制网络连接的最大数目。

语法

mlsrv11 -c "*connection-string***" -sm** *number ...*

注释

MobiLink 服务器同时执行以下任务：

1. 从网络读取上载数据并将其解开。
2. 将上载应用到统一数据库。
3. 从统一数据库读取要下载的行。
4. 压缩下载数据并将其发送到远程数据库。

每个任务的同步数受以下限制：

- 执行任务 2 和 3 的同步数小于或等于 mlsrv11 -w 选项的设置。
- 执行任务 2 的同步数小于或等于 mlsrv11 -wu 选项的设置。
- 执行所有四项任务的同步数小于或等于 -sm 选项的设置。

较高的 -sm 值（尤其远高于 -w 值时）将允许 MobiLink 服务器执行比数据库任务（2 和 3）更多的网络任务（1 和 4）。这有助于确保在网络性能本来会成为瓶颈的情况下，数据库工作不必等待任务。从而可以提高吞吐量。但是，如果 -sm 值设置过高，MobiLink 服务器会分配多于直接可用内存的内存，从而激活操作系统的虚拟内存分页，而这又使得内存交换到磁盘上，最终显著降低吞吐量。

另请参见

- “-w 选项”一节第 100 页
- “-wu 选项”一节第 101 页

-ss 选项

需要单独授予许可的组件
-ss 选项是 MobiLink 高可用性选项的一个功能，它需要单独的许可证。请参见“单独授权的组件”一节 《SQL Anywhere 11 - 简介》。

使 MobiLink 服务器能够在服务器群中运行。

语法

mlsrv11 -c "*connection string***" -ss** ...

注释

缺省情况下，MobiLink 服务器并不在服务器群中运行。如果想要使当前 MobiLink 服务器能够在服务器群中运行，则需指定此选项。请注意，如果 MobiLink 服务器启动时指定了 -ss 选项，它会将自身添加到统一数据库的 ml_server 表中，同时停止其它启动时没有指定 -ss 选项的 MobiLink 服务器（如果有）与该统一数据库的连接。请参见“在服务器群中运行 MobiLink 服务器”一节第 37 页。

如果指定了此选项，服务器必须有使用 -zs 选项指定的名称。例如，服务器命令行可以为：

mlsrv11 -c "*connection-string***" -ss -zs** *server5*

另请参见

- “在服务器群中运行 MobiLink 服务器”一节第 37 页
- “-zs 选项”一节第 112 页
- “-lsc 选项”一节第 68 页

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

-tc 选项

为 SQL 脚本的执行设置长时间运行的语句阈值。

语法

mlsrv11 -c "*connection string***" -tc** *minutes* ...

注释

缺省情况下，MobiLink 服务器监视每个 SQL 脚本的执行时间，并在脚本的执行时间达到 10 分钟（缺省值）时发出警告消息。如果使用了 -tf 选项并且统一数据库运行在 Oracle 服务器上，MobiLink 服务器将使脚本失败。

缺省值可以重置为零或正整数，单位为分钟。当缺省值被设置为零时，将禁用 -tc 开关，此时 MobiLink 服务器不会监视任何脚本的执行。

当警告时间被设置为非零值时，MobiLink 服务器会以指数方式显示警告信息。当执行时间第一次超过指定时间时，会显示警告；当执行时间超过 2 x 给定时间时会再次显示警告，然后是 4 x 给定时间，以此类推。

警告消息中包含用于当前同步的连接 ID 和包括以下信息（如果可用）的超时警告上下文：远程 ID、ML 用户名、修改的用户名、事务、表名、行值和脚本版本。无论 MobiLink 服务器的详细程度设置如何，都会显示超时警告上下文。

当统一数据库在 Oracle 数据库服务器上运行并出现超时警告消息时，具有 DBA 权限的数据库用户可能需要检查统一数据库，以便确定产生问题的原因。可以在警告消息中找到同步所使用的连接 SID 和 SERIAL#。如果同步连接已停止，MobiLink 服务器会终止当前同步。

另请参见

- “-tf 选项”一节第 92 页

Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

-tf 选项

如果执行时间超过了 -tc 选项所指定的时间，使用此选项使 MobiLink 服务器造成 SQL 脚本执行失败。当统一数据库在 Oracle 服务器上运行时此选项不可用。

语法

mlsrv11 -c "*connection string***" -tf** ...

注释

如果 SQL 脚本执行失败，MobiLink 服务器会跳过行（如果脚本是上载脚本且 handle_error 脚本返回 1000），然后继续同步或中止同步。

如果指定了此选项并且针对 Oracle 服务器执行此选项，则 MobiLink 服务器会显示警告消息。

如果指定了 **-tc 0** 则忽略此选项。

Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

-tx 选项

使用事务性上载时，此选项会将事务组进行批处理，然后将它们一起提交。

语法

mlsrv11 -c "*connection-string***" -tx** *count* ...

注释

当进行事务性上载时可使用此选项来提高性能。

count 可以是任何非负的值。缺省值为 1，表示单独提交每个事务。而使用 0 值表示在完成上载所有事务后再执行一次提交。

另请参见

- “-tu 选项”一节 《MobiLink - 客户端管理》

-ud 选项

指示 MobiLink 作为守护程序运行。

语法

mlsrv11 -c "*connection-string***" -ud** ...

注释

仅适用于 Unix 平台。

另请参见

- “在当前会话外运行 MobiLink 服务器”一节第 33 页

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

-ui 选项

适用于具有 X 窗口服务器支持的 Linux，在未提供可用显示的情况下以 shell 模式启动 MobiLink 服务器。

语法

mlsrv11 -c "*connection-string***" -ui** ...

注释

使用此选项时，mlsrv11 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，服务器会尝试查找一个可用显示。如果没有找到，例如因为 X 窗口服务器没有运行，那么 MobiLink 服务器会以 shell 模式启动。

-ux 选项

适用于 Linux，用于打开显示消息的 MobiLink 服务器消息窗口。

语法

mlsrv11 -c "*connection-string* **-ux** ...

注释

当指定 -ux 时，MobiLink 服务器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，因为未设置 DISPLAY 环境变量或 X 窗口服务器未运行），MobiLink 服务器将无法启动。

若要以安静模式运行 MobiLink 服务器消息窗口，请使用 -q 选项。

在 Windows 上，MobiLink 服务器消息窗口会自动显示。

另请参见

- “-q 选项”一节第 82 页

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

-v 选项

允许您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。

语法

mlsrv11 -c "*connection-string***" -v[** *levels* **]** ...

注释

当使用 dbmlsync 事务级上载时，此选项特别有用。

该选项控制写入消息日志文件的消息类型。

如果仅指定 -v 选项，则 MobiLink 服务器记录有关每次同步的最小信息量。

高详细程度反而会影响性能，应该仅在开发期间使用。

levels 的值如下所示。一次可以使用一个或多个选项，例如 -vnrsu。

- **+** 打开所有增加详细程度的日志选项。
- **c** 在每个同步脚本被调用时显示其内容。该级别隐含 s。
- **e** 显示系统事件脚本。这些系统事件脚本用于维护 MobiLink 系统表及控制上载的 SQL 脚本。
- **f** 显示第一次读取的错误。这将记录以下情况下引发的错误：负载平衡设备通过建立不发送任何数据的连接检查服务器活动性，并会导致同步失败。

  对于 TCP/IP 连接，使用 TCP/IP 选项 **ignore** 可能会更好。有关详细信息，请参见“-x 选项”一节第 102 页。
- **h** 显示同步过程中上载的远程模式。
- **i** 显示上载的每行的列值。请使用此选项而不是 -vr，-vr 选项将显示上载和下载的每行的列值来降低正在记录的数据量。通过 -vq 指定 -vi 与指定 -vr 相同。
- **m** 每当同步完成后将每个同步的持续时间以及每个同步阶段的持续时间打印到日志中。同步阶段如下所示。它们与显示在 MobiLink 监控器中的阶段相同。所有时间均以微秒 (ms) 为单位显示。
  - **同步请求** 从建立远程数据库与 MobiLink 服务器之间的网络连接直至接收到上载流的第一批字节为止所花费的时间。当同步数大于由 -sm 指定的最大活动同步数时，除非您已将 -sm 设为一个比 -nc 更小的值（在此情况下，此时间包含同步暂停的时间），否则该时间是没有意义的。
  - **接收上载** 从 MobiLink 服务器接收到上载流的第一批字节开始直至完全接收到远程数据库的上载流为止所花费的时间。上载流包含表定义和正在上载的远程数据库行，所以即使对于仅下载同步，该时间也是有意义的。该时间取决于上载流的大小以及网络传输的带宽。
  - **获取 DB 工作** 获得空闲数据库工作线程所需的时间。
  - **连接** 在需要新数据库连接时数据库工作线程建立数据库连接所需的时间。例如，出现错误或脚本版本变更的情况下会需要建立新连接。
  - **验证用户** 验证用户所需的时间。

- **开始同步** begin_synchronization 事件（即使已定义）所需的时间，加上为每个预订读取 last_upload_time 的时间（使用共享的 MobiLink 管理连接）。
- **应用上载** 上载数据应用到统一数据库所需的时间。
- **准备下载** prepare_for_download 事件所需的时间。
- **读取下载** 为创建下载流而读取从统一数据库下载的行所需的时间。
- **结束同步** end_synchronization 事件所需的时间，在该时间后将释放数据库工作线程。如果使用阻塞下载确认，则此阶段出现在 [**等待下载肯定应答**] 阶段之后。否则，它会出现在下载流被送到远程数据库之前。
- **发送下载** 将下载流发送到远程数据库所需的时间。该时间取决于下载流的大小以及网络传输的带宽。对于仅上载同步，下载流只是上载确认。
- **等待下载肯定应答** 等待将下载应用到远程数据库以及远程数据库发送下载确认所花费的时间。此阶段只有在远程数据库启用了下载确认的情况下才会出现。
- **获取下载肯定应答的 DB 工作** 收到下载确认后，等待空闲数据库工作线程所花费的时间。此阶段只有在远程数据库启用了下载确认并且 MobiLink 服务器正在使用非阻塞下载确认的情况下才会出现。
- **为下载肯定应答连接** 在需要新数据库连接时数据库工作线程建立数据库连接所需的时间。此阶段只有在远程数据库启用了下载确认并且 MobiLink 服务器正在使用非阻塞下载确认的情况下才会出现。
- **非阻塞下载肯定应答** publication_nonblocking_download_ack 和 nonblocking_download_ack 连接事件所需的时间。此阶段只有在远程数据库启用了下载确认并且 MobiLink 服务器正在使用非阻塞下载确认的情况下才会出现。

每个值都以 "PHASE:" 作为前缀来帮助搜索和打印值。

以下示例为显示各种同步阶段的持续时间的示例输出：

```
I. 2008-06-05 14:48:36. <1> PHASE: start_time: 2008-06-05 14:48:36.048
I. 2008-06-05 14:48:36. <1> PHASE: duration: 175
I. 2008-06-05 14:48:36. <1> PHASE: sync_request: 0
I. 2008-06-05 14:48:36. <1> PHASE: receive_upload: 19
I. 2008-06-05 14:48:36. <1> PHASE: get_db_worker: 0
I. 2008-06-05 14:48:36. <1> PHASE: connect: 18
I. 2008-06-05 14:48:36. <1> PHASE: authenticate_user: 51
I. 2008-06-05 14:48:36. <1> PHASE: begin_sync: 69
I. 2008-06-05 14:48:36. <1> PHASE: apply_upload: 0
I. 2008-06-05 14:48:36. <1> PHASE: prepare_for_download: 1
I. 2008-06-05 14:48:36. <1> PHASE: fetch_download: 4
I. 2008-06-05 14:48:36. <1> PHASE: wait_for_download_ack: 0
I. 2008-06-05 14:48:36. <1> PHASE: end_sync: 0
I. 2008-06-05 14:48:36. <1> PHASE: send_download: 10
I. 2008-06-05 14:48:36. <1> PHASE: get_db_worker_for_download_ack: 0
I. 2008-06-05 14:48:36. <1> PHASE: connect_for_download_ack: 0
I. 2008-06-05 14:48:36. <1> PHASE: nonblocking_download_ack: 0
```

- **n** 显示汇总行计数。
- **o** 显示 SQL 直通活动。
- **p** 显示进程偏移。

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

- **q** 显示下载的每行的列值。请使用此选项而不是 -vr，-vr 选项将显示上载和下载的每行的列值来降低正在记录的数据量。通过 -vq 指定 -vi 与指定 -vr 相同。
- **r** 显示上载或下载的每行的列值。要仅记录上载的每行的列值，请使用 -vi。要仅记录下载的每行的列值，请使用 -vq。
- **s** 在每个同步脚本被调用时显示其名称。
- **t** 显示经过转换的 SQL，这些 SQL 由使用 ODBC 规范格式编写的脚本生成。该级别隐含 c。以下示例显示针对 SQL Anywhere 的语句的自动转换。

```
I. 02/11 11:02:14. [102]: begin_upload synch2
{ call SynchLogLine( ?, ?, 'begin_upload' ) }
I. 02/11 11:02:14. [102]: Translated SQL:
call SynchLogLine( ?, ?, 'begin_upload' )
```

  以下示例显示针对 Microsoft SQL Server 相同语句的转换。

```
I. 02/11 11:03:21. [102]: begin_upload synch2
{ call SynchLogLine( ?, ?, 'begin_upload' ) }
I. 02/11 11:03:21. [102]: Translated SQL:
EXEC SynchLogLine ?, ?, 'begin_upload'
```

- **u** 显示未定义的表脚本。这也许能帮助新用户了解同步过程。

另请参见

- “MobiLink 统计信息属性”一节第 181 页

-w 选项

设置数据库工作线程的数目。

语法

mlsrv11 -c "*connection-string***" -w** *count* ...

注释

每个数据库工作线程每次只接受一个同步请求。

每个数据库工作线程都使用一个统一数据库连接。MobiLink 服务器额外打开另一个连接用于管理。因此，从 MobiLink 服务器到统一数据库的最小连接数是 *count* + 1。

数据库工作线程数对 MobiLink 同步吞吐量有很大影响，您应当进行测试来确定适合您的特定同步设置的最佳数目。数据库工作线程数决定在统一数据库中有多少同步操作可以同时处于活动状态；其余的同步操作将排队等候数据库工作线程变得可用。添加数据库工作线程将增加吞吐量，但同时也会增加活动同步之间发生争用的可能性。达到某个程度后，添加数据库工作线程会降低吞吐量，因为争用的增加超过了重叠的同步所带来的好处。

为此选项设置的值同时也是 -wu 选项（用于限制可同时上载到统一数据库的线程数）的缺省设置。如果用于下载的数据库工作线程的最佳数目大于用于上载的最佳数目，这会很有用。为获得最佳吞吐量，应使用较大的数据库工作线程数（通过 -w），但允许较少的工作线程同时执行上载（通过 -wu）。一般来说，用于 -wu 的最佳数目与统一数据库有关，而与远程数据库的处理或网络速度关系不大。因此，当您使用 -w 来增大线程数时，您可能需要使用 -wu 来限制可同时上载的线程数。有关详细信息，请参见“-wu 选项”一节第 101 页。

缺省数据库工作线程数为 **5**。

另请参见

- “-wu 选项”一节第 101 页
- “-sm 选项”一节第 89 页
- “-cn 选项”一节第 52 页

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

-wu 选项

设置可以同时将上载应用到统一数据库的数据库工作线程的最大数目。

语法

mlsrv11 -c "*connection-string***" -wu** *count* ...

注释

使用 -wu 选项来限制可以将上载同时应用于统一数据库的数据库工作线程的数目。达到限制后，准备将其上载应用于统一数据库的数据库工作线程必须等到另一工作线程完成其上载后才能应用其上载。

统一数据库中发生争用的最常见原因是同时执行上载的数据库工作线程太多。下载引起的争用问题就少得多，所以只通过 mlsrv11 -w 选项限制下载。因此，-w 设置必须大于或等于 -wu 设置。

缺省情况下，所有的数据库工作线程可同时执行上载。使用的数据库工作线程数是通过 -w 选项设置的。缺省值为 **5**。

示例

在一个使用 LAN 并在多台 PC 上运行远程数据库的实验性安装中，您会发现最佳数据库工作线程数对于仅上载同步和仅下载同步大约都是 10 个，这相当于统一数据库上 100% 的 CPU 使用率。如果使用较少的数据库工作线程，则会发现吞吐量减少，而且统一数据库的 CPU 使用率降低。如果使用更多数据库工作线程，吞吐量并不增大，因为在使用 10 个工作线程时，统一数据库的处理速度已达到最大限度。

另请参见

- “-w 选项”一节第 100 页
- “-sm 选项”一节第 89 页

Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

-x 选项

为 MobiLink 客户端设置网络协议和协议选项。MobiLink 服务器使用该协议和这些参数来监听是否有同步请求。

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

mlsrv11 -c "*connection-string***" -x** *protocol*[*protocol-options*] ...

protocol : **tcpip** | **tls** | **http** | **https**

protocol-options : **(** *option*=*value*; ... **)**

缺省

缺省协议是 TCPIP，端口为 2439。

参数

协议的允许值如下所示：

- **tcpip** 通过 TCP/IP 接受连接。
- **tls** 通过使用传送层安全性的 TCP/IP 接受连接。
- **http** 通过标准 Web 协议接受连接。
- **https** 使用处理安全事务的 HTTP 的变异版本接受连接。使用 RSA 或 ECC 加密，HTTPS 协议通过 SSL/TLS 实现 HTTP。

您还可以指定以下网络协议选项，形式为 *option*=*value*。必须用分号分隔多个选项。

- **TCP/IP 选项** 如果指定了 tcpip 协议，则还可以有选择地指定以下协议选项（这些选项区分大小写）：

TCP/IP 协议选项	说明
host=*hostname*	MobiLink 服务器监听的主机名或 IP 值。缺省值为 localhost。

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

TCP/IP 协议选项	说明
ignore=*hostname*	进行连接时 MobiLink 服务器所忽略的主机名或 IP 地址。此选项使您可以忽略来自最低可能级别的负载平衡器的请求，从而防止 MobiLink 服务器日志和 MobiLink 监控器输出文件中出现过多的输出。可以指定多个需要忽略的主机，例如 `-x tcpip(ignore=lb1;ignore=123.45.67.89)`。如果在命令行上指定多个 -x 实例，将在所有实例上忽略相应的主机；例如，如果指定 -x tcpip(ignore=1.1.1.1) -x http，则在 TCP/IP 和 HTTP 流上均忽略 1.1.1.1 的连接。但这并不影响通过 -xo 选项进行的连接。
port=*portnumber*	MobiLink 服务器监听的套接字端口号。缺省端口为 2439，这是用于 MobiLink 服务器的 IANA 注册端口号。

- **用于具有传送层安全性的 TCP/IP 的选项** 如果指定 tls 协议（该协议是具有传送层安全的 TCP/IP 协议），则可选择指定以下协议选项（这些选项区分大小写）：

TLS 协议选项	说明
fips={**yes**|**no**}	如果通过 tls_type=rsa 指定了 TLS 协议，可以指定 fips=yes 以便使用 TCP/IP 协议和 FIPS 认可的加密算法接受连接。FIPS 连接使用单独的 FIPS 140-2 认证软件。使用无 FIPS 的 RSA 加密的服务器与使用有 FIPS 的 RSA 加密的客户端兼容，使用有 FIPS 的 RSA 加密的服务器与使用无 FIPS 的 RSA 加密的客户端兼容。
host=*hostname*	MobiLink 服务器监听的主机名或 IP 地址。缺省值为 localhost。
ignore=*hostname*	进行连接时 MobiLink 服务器所忽略的主机名或 IP 地址。此选项使您可以忽略来自最低可能级别的负载平衡器的请求，从而防止 MobiLink 服务器日志和 MobiLink 监控器输出文件中出现过多的输出。可以指定多个需要忽略的主机，例如 `-x tcpip(ignore=lb1;ignore=123.45.67.89)`。
port=*portnumber*	MobiLink 服务器监听的套接字端口号。缺省端口为 2439，这是用于 MobiLink 服务器的 IANA 注册端口号。

TLS 协议选项	说明
tls_type={**rsa**|**ecc**}	如果将 TCP/IP 协议指定为 tls，则可以指定椭圆曲线加密算法 (ecc) 或 RSA 加密 (rsa)。为了向后兼容，还可以将 ecc 指定为 certicom。缺省的 tls_type 为 rsa。 当使用 TLS 时，必须指定标识和标识口令： ○ **identity=*identity-file*** 指定要用于服务器验证的标识文件的路径和文件名。 ○ **identity_password=*password*** 为标识指定口令 请参见“启动支持传送层安全的 MobiLink 服务器”一节 《SQL Anywhere 服务器 - 数据库管理》。
e2ee_type={**rsa**|**ecc**}	交换会话密钥所使用的密钥类型。必须是 rsa 或 ecc，而且必须与专用密钥文件（参见下一个选项）中的密钥类型相匹配。缺省的 e2ee_type 为 rsa。
e2ee_private_key=*file*	包含 rsa 或 ecc 专用密钥的 PEM 编码文件。若要使端对端加密生效，则需要使用此选项。 可使用 createkey 实用程序创建 PEM 编码文件。请参见“密钥对生成器实用程序 (createkey)”一节 《SQL Anywhere 服务器 - 数据库管理》。
e2ee_private_key_password=*password*	专用密钥文件的口令。若要使端对端加密生效，则需要使用此选项。

- **HTTP 选项** 如果指定了 http 协议，则还可以有选择地指定以下协议选项（这些选项区分大小写）：

HTTP 选项	说明
buffer_size=*数字*	自 MobiLink 服务器发送的 HTTP 消息的最大主体大小，单位为字节。更改此选项会减小或增大为发送 HTTP 消息而分配的内存大小。缺省值为 65535 个字节。
host=*hostname*	MobiLink 服务器监听的主机名或 IP 地址。缺省值为 localhost。
port=*portnumber*	MobiLink 服务器监听的套接字端口号。端口号必须与 MobiLink 服务器所监控端口相匹配。缺省端口是 80。
version=*http-version*	MobiLink 服务器自动检测客户端使用的 HTTP 版本。该参数是一个字符串，它用于在服务器无法检测到客户端使用的方法时指定缺省的 HTTP 版本。可以选择 1.0 或 1.1。缺省值是 1.1。

- **HTTPS 选项** HTTPS 协议使用 RSA 或 ECC 数字证书以实现传送层安全性。如果指定 FIPS 加密，则协议将使用单独的、与 https 兼容的 FIPS 140-2 认证软件。

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

有关详细信息，请参见“启动支持传送层安全的 MobiLink 服务器”一节 《SQL Anywhere 服务器 - 数据库管理》。

如果指定了 **https** 协议，则还可以有选择地指定以下协议选项（这些选项区分大小写）：

HTTPS 选项	说明
buffer_size=*数字*	自 MobiLink 服务器发送的 HTTPS 消息的最大主体大小，单位为字节。更改此选项会减小或增大为发送 HTTPS 消息而分配的内存大小。缺省值为 65535 个字节。
identity=*server-identity*	要用于服务器验证的标识文件的路径和文件名。对于 HTTPS，必须使用 RSA 证书。
identity_password=*password*	用于为标识文件指定口令的可选参数。 请参见“传送层安全” 《SQL Anywhere 服务器 - 数据库管理》。
fips={yes|no}	可以指定 fips=yes 以便使用 HTTPS 协议和 FIPS 认可的加密算法接受连接。FIPS 连接使用单独的 FIPS 140-2 认证软件。使用无 FIPS 的 RSA 加密的服务器与使用有 FIPS 的 RSA 加密的客户端兼容，使用有 FIPS 的 RSA 加密的服务器与使用无 FIPS 的 RSA 加密的客户端兼容。
host=*hostname*	MobiLink 服务器监听的主机名或 IP 地址。缺省值为 localhost。
port=*portnumber*	MobiLink 服务器监听的套接字端口号。端口号必须与 MobiLink 服务器所监控端口相匹配。缺省端口是 443。
tls_type={rsa|ecc}	如果将 TCP/IP 协议指定为 tls，则可以指定椭圆曲线加密算法 (ecc) 或 RSA 加密 (rsa)。为了向后兼容，还可以将 ecc 指定为 certicom。缺省的 tls_type 为 rsa。 使用传送层安全性时，必须指定标识和标识口令： ○ **identity=*identity-file*** 指定要用于服务器验证的标识文件的路径和文件名。 ○ **identity_password=password** 指定标识文件的口令。 请参见“启动支持传送层安全的 MobiLink 服务器”一节 《SQL Anywhere 服务器 - 数据库管理》。

HTTPS 选项	说明
version=*http-version*	MobiLink 服务器自动检测客户端使用的 HTTP 版本。该参数是一个字符串，它用于在服务器无法检测到客户端使用的方法时指定缺省的 HTTP 版本。可以选择 1.0 或 1.1。缺省值是 1.1。
e2ee_type={**rsa**|**ecc**}	交换会话密钥所使用的密钥类型。必须是 rsa 或 ecc，而且必须与专用密钥文件（参见下一个选项）中的密钥类型相匹配。缺省的 e2ee_type 为 rsa。
e2ee_private_key=*file*	包含 rsa 或 ecc 专用密钥的 PEM 编码文件。若要使端对端加密生效，则需要使用此选项。 可使用 createkey 实用程序创建 PEM 编码文件。请参见“密钥对生成器实用程序 (createkey)”一节《SQL Anywhere 服务器 - 数据库管理》。
e2ee_private_key_password=*password*	专用密钥文件的口令。若要使端对端加密生效，则需要使用此选项。

示例

以下命令行将端口设置为 12345：

```
mlsrv11 -c "dsn=SQL Anywhere 11 CustDB;uid=ml_server;pwd=sql" -x
tcpip(port=12345)
```

下面的示例指定了安全类型 (RSA)、服务器标识文件和保护服务器专用密钥的标识口令：

```
mlsrv11 -c "dsn=my_cons"
 -x tls(tls_type=rsa;identity=c:\test\serv_rsa1.crt;identity_password=pwd)
```

以下示例与前面的示例相似，不同之处在于本例的标识文件名中包含空格：

```
mlsrv11 -c "dsn=my_cons"
 -x "tls(tls_type=rsa;identity=c:\Program Files\test
\serv_rsa1.crt;identity_password=pwd)"
```

以下示例显示如何通过 HTTPS 协议使用端对端加密：

```
mlsrv11 -c "dsn=my_cons" -x https(tls_type=rsa;identity=my_identity.crt;
identity_password=my_id_pwd;e2ee_type=rsa;e2ee_private_key=my_pk.pem;
e2ee_private_key_password=my_pk_pwd)
```


Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

-xo 选项

为版本 8 和 9 MobiLink 客户端设置网络协议和协议选项。

语法

mlsrv11 -c "*connection-string***"**
-xo *protocol*[*protocol-options*] ...

protocol-options : **(** *keyword***=***value*; ... **)**

注释

指定与客户端应用程序进行通信所使用的通信协议。缺省协议是 TCPIP，端口为 2439。

协议的允许值如下所示：

- **tcpip** 使用 TCP/IP 接受来自应用程序的连接。
- **http** 通过标准 Web 协议接受连接。客户端应用程序可以挑选自己的 HTTP 版本，MobiLink 服务器将基于每个连接进行相应的调整。
- **https** 使用处理安全事务的 HTTP 的变异版本接受连接。这个 HTTPS 协议通过 SSL/TLS（使用 RSA 加密）实现 HTTP，并且与任何其它 HTTPS 服务器都兼容。
- **https_fips** 通过使用 HTTPS 协议和经 FIPS 认可的加密算法接受连接。HTTPS_FIPS 使用单独的 FIPS 140-2 认证软件。使用 rsa_tls 的服务器与使用 rsa_tls_fips 的客户端兼容，并且使用 rsa_tls_fips 的服务器与使用 rsa_tls 的客户端兼容。

您还可以指定网络协议选项，形式为 *option=value*。必须用分号分隔多个选项。可以指定的选项取决于所选择的协议。

- **TCP/IP 选项** 如果指定 tcpip 协议，则还可以选择指定以下协议选项：
  - **backlog=number-of-connections** 在 MobiLink 服务器拒绝新的同步请求（从而在客户端导致同步失败）之前所允许的最大远程连接数。通过指定积压请求数限制，可以避免客户端在服务器忙时等待进行同步。如果未指定积压请求数限制，MobiLink 服务器将接受尽可能多的连接，从而可能达到或超过操作系统的网络连接限制。这可能导致速度减慢或性能不稳定。

    客户端尝试与 MobiLink 服务器同步时，如果服务器已接受了其最大的远程连接数，则客户端会收到错误代码 -85 (SQLE_COMMUNICATIONS_ERROR)。客户端应用程序应该处理该错误并在几分钟后再次尝试连接。

    有关 SQLE_COMMUNICATIONS_ERROR 的详细信息，请参见“通信错误”一节 《错误消息》。

    如果在可同时进行上万个同步的环境中使用 MobiLink，可使用该 backlog 选项来指定一个低于操作系统限制的最大远程连接数。请参见“操作系统限制计划”一节第 161 页。
  - **host=hostname** MobiLink 服务器监听的主机名或 IP 地址。缺省值为 localhost。
  - **ignore=hostname** 在进行连接时被 MobiLink 服务器忽略的主机名或 IP 值。此选项使您可以忽略来自最低可能级别的负载平衡器的请求，从而防止 MobiLink 服务器日志和

MobiLink 监控器输出文件中出现过多的输出。可以指定多个需要忽略的主机，例如 -x tcpip(ignore=lb1;ignore=123.45.67.89)。

- **liveness_timeout=n** 在与某客户端进行最后一次通信之后，到 MobiLink 终止同步之前所经过的时间量（以秒为单位）。值为 0 表示没有超时。缺省值是 120 秒。
- **port=portnumber** MobiLink 服务器监听的套接字端口号。缺省端口为 2439，这是用于 MobiLink 服务器的 IANA 注册端口号。

> **注意**
> mlsrv11 -x 和 -xo 选项使用同一个缺省端口，而且即使没有指定 -x 选项也会启动。因此，若不在 -x 选项中更改端口，就必须更改 -xo 的端口。

- **security=cipher(keyword=value;...)** 通过此连接的所有通信都会使用传送层安全性进行加密和验证。编码器可以是以下各项之一：

编码器	说明
rsa_tls	RSA 加密。
rsa_tls_fips	经 FIPS 认可的 RSA 加密。rsa_tls_fips 使用单独的 FIPS 140-2 认证软件，但是它与使用 https 的客户端（以及 SQL Anywhere 9.0.2 版或更高版本）兼容。
ecc_tls	椭圆曲线加密算法。为了向后兼容，还可以将 ecc_tls 指定为 certicom_tls。

安全参数为 certificate（用于服务器验证的标识文件的路径和文件名）和 certificate_password。必须使用与您选择的编码器套件匹配的证书。

有关详细信息，请参见“启动支持传送层安全的 MobiLink 服务器”一节 《SQL Anywhere 服务器 - 数据库管理》。

> **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》。

- **HTTP 选项** 如果指定 http 协议，则还可以选择指定以下协议选项：
  - **backlog=number-of-connections** 在 MobiLink 服务器拒绝新的同步请求（从而在客户端导致同步失败）之前所允许的最大远程连接数。通过指定积压请求数限制，可以避免客户端在服务器忙时等待进行同步。如果未指定积压请求数限制，MobiLink 服务器将接受尽可能多的连接，从而可能达到或超过操作系统的网络连接限制。这可能导致速度减慢或性能不稳定。

    客户端尝试与 MobiLink 服务器同步时，如果服务器已接受了其最大的远程连接数，则客户端会收到错误代码 -85 (SQLE_COMMUNICATIONS_ERROR)。客户端应用程序应该处理该错误并在几分钟后再次尝试连接。

    有关 SQLE_COMMUNICATIONS_ERROR 的详细信息，请参见“通信错误”一节 《错误消息》。

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

如果在可同时进行上万个同步的环境中使用 MobiLink，可使用该 backlog 选项来指定一个低于操作系统限制的最大远程连接数。有关详细信息，请参见“操作系统限制计划”一节 第 161 页。

- **buffer_size=number** 自 MobiLink 服务器发送的 HTTP 消息的最大主体大小，单位为字节。更改此选项会减小或增大为发送 HTTP 消息而分配的内存大小。缺省值为 65535 个字节。
- **contd_timeout=seconds** MobiLink 服务器在放弃同步之前等待接收未完成的同步中下一部分的秒数。您可以调整该选项，以便在等待时间较长足以表明客户端不会继续连接时释放 MobiLink 服务器的资源。缺省值为 30 秒。
- **host=hostname** MobiLink 服务器监听的主机名或 IP 地址。缺省值为 localhost。
- **port=portnumber** MobiLink 服务器监听的套接字端口号。端口号必须与 MobiLink 服务器正在监控的端口相匹配。缺省端口是 80。

> **注意**
> mlsrv11 -x 和 -xo 选项使用同一个缺省端口，并且即使未指定 -x 选项也会启动。因此，若不在 -x 选项中更改端口，就必须更改 -xo 的端口。

- **session_key={ cookie | header }** 创建一个 JSESSIONID 的替代方法，用于跟踪连接。如果您的网络已使用了 JSESSIONID，此选项可能是必需的。
- **unknown_timeout=seconds** 在 MobiLink 服务器放弃同步之前在新连接上等待接收 HTTP 标头的秒数。您可以调整该选项，以便在等待时间较长足以表明客户端不再继续连接时释放 MobiLink 服务器资源。缺省值为 30 秒。
- **version=http 版本** MobiLink 服务器自动检测客户端使用的 HTTP 版本。该参数是一个字符串，它用于在服务器无法检测到客户端使用的方法时指定缺省的 HTTP 版本。可以选择 1.0 或 1.1。缺省值是 1.1。

- **HTTPS 或 HTTPS_FIPS 选项** https 协议使用 RSA 数字证书来实现传送层安全性。https_fips 协议使用单独的 FIPS 140-2 认证软件，但该软件与 https 兼容。

  有关详细信息，请参见“启动支持传送层安全的 MobiLink 服务器”一节 《SQL Anywhere 服务器 - 数据库管理》。

  > **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》。

  如果指定 https 协议，则可选择指定以下协议选项：

  - **backlog=number-of-connections** 在 MobiLink 拒绝新的同步请求（从而在客户端导致同步失败）之前所允许的最大远程连接数。通过指定积压请求数限制，可以避免客户端在服务器忙时等待进行同步。如果不指定积压请求数限制，则无论存在多少积压请求，客户端都会尝试进行同步。

- **buffer_size=number** 自 MobiLink 服务器发送的 HTTPS 消息的最大主体大小，单位为字节。更改此选项会减小或增大为发送 HTTPS 消息而分配的内存大小。缺省值为 65535 个字节。
- **contd_timeout=seconds** MobiLink 服务器在放弃同步之前等待接收未完成的同步中下一部分的秒数。您可以调整该选项，以便在等待时间较长足以表明客户端不会继续连接时释放 MobiLink 服务器的资源。缺省值为 30 秒。
- **host=hostname** MobiLink 服务器监听的主机名或 IP 地址。缺省值为 localhost。
- **port=portnumber** MobiLink 服务器监听的套接字端口号。端口号必须与 MobiLink 服务器所监控的端口相匹配。缺省端口是 443。

  注意
  mlsrv11 -x 和 -xo 选项使用同一个缺省端口并启动 -x（即使未指定它），因此，若不在 -x 选项中更改端口，就必须更改 -xo 的端口。

- **certificate** 用于服务器验证的证书文件的路径和文件名。它必须是 RSA 证书。
- **certificate_password** 一个用于为证书文件指定口令的可选参数。

  有关安全性的详细信息，请参见“传送层安全”《SQL Anywhere 服务器 - 数据库管理》。
- **session_key={ cookie | header }** 创建一个 JSESSIONID 的替代方法，用于跟踪连接。如果您的网络已使用了 JSESSIONID，此选项可能是必需的。
- **unknown_timeout=seconds** 在 MobiLink 服务器放弃同步之前在新连接上等待接收 HTTP 标头的秒数。您可以调整该选项，以便在等待时间较长足以表明客户端不再继续连接时释放 MobiLink 服务器资源。缺省值为 30 秒。
- **version=http 版本** MobiLink 服务器自动检测客户端使用的 HTTP 版本。该参数是一个字符串，它用于在服务器无法检测到客户端使用的方法时指定缺省的 HTTP 版本。可以选择 1.0 或 1.1。缺省值是 1.1。

示例

下面的命令行将积压请求数设置为 10 个连接。

```
mlsrv11 -c "dsn=SQL Anywhere 11 CustDB;uid=ml_server;pwd=sql" -xo
http(backlog=10)
```

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

-zp 选项

调整以确定使用哪些时间戳值进行冲突检测。

语法

mlsrv11 -c "*connection-string***" -zp**

注释

如果统一数据库和远程数据库之间发生时间戳冲突，利用此选项可将精度比最低精度高的时间戳值用于检测冲突。当统一数据库中的时间戳比远程数据库中的时间戳更为精确时可使用此选项，因为在远程数据库中更新时间戳会导致在下一次同步时发生冲突。此选项允许 MobiLink 忽略这些冲突。如果在统一数据库与远程数据库之间精确度不匹配并且没有使用 -zp，在日志中将为每个同步及每个模式相关的表写入相应警告，以建议使用 -zp 选项。另外还为每个同步添加一个警告，通知用户在条件允许的情况下调整远程数据库上时间戳的精度。

-zs 选项

在服务器群中为 mlstop 和共享服务器状态指定 MobiLink 服务器名。

语法

mlsrv11 -c "*connection-string***" -zs** *name*

注释

指定的名称可以包含 ASCII 字母和数字，但不能包含任何其它字符。

当 mlstop 用于关闭通过 -zs 选项启动的 MobiLink 服务器时，必须在 mlstop 命令行上指定该服务器名。例如，`mlstop myMLserver`。只能从安装了 MobiLink 服务器的计算机启动关闭。

当 MobiLink 在服务器群中运行时，必须指定此名以便唯一标识服务器。

另请参见

- “MobiLink 停止实用程序 (mlstop)”一节第 649 页
- “在服务器群中运行 MobiLink 服务器”一节第 37 页
- “-lsc 选项”一节第 68 页
- “-ss 选项”一节第 90 页
- “MobiLink 服务器群中的通告程序”一节 《MobiLink - 服务器启动的同步》

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

-zt 选项

指定用来运行 MobiLink 服务器的处理器的最大数量。

语法

mlsrv11 -c "*connection-string***" -zt** *number*

注释

此选项对于某些 ODBC 驱动程序来说是必需的。此选项还使您可以对处理器资源进行精细控制。

此选项只能在 Windows 操作系统上使用。缺省值是计算机上的处理器数量。

-zu 选项

在未定义 authenticate_user 脚本时，控制用户的自动添加。

语法

mlsrv11 -c "*connection-string***" -zu**{ **+** | **-** } ...

注释

如果指定为 -zu+，则在首次同步时向 ml_user 表中添加未识别的 MobiLink 用户名。如果该参数被指定为 -zu- 或未指定，将导致未识别的用户名无法进行同步。

此选项在开发过程中对注册用户非常有用。对于已部署的应用程序，建议不使用此选项。

另请参见

- “新用户的同步”一节 《MobiLink - 客户端管理》
- “MobiLink 用户” 《MobiLink - 客户端管理》
- “MobiLink 用户验证实用程序 (mluser)”一节第 650 页
- “authenticate_user 连接事件”一节第 335 页

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

-zus 选项

即使没有为表上载任何行，也会使 MobiLink 服务器调用该表的上载脚本。

语法

mlsrv11 -c "*connection-string***" -zus** ...

注释

缺省情况下，如果没有为表上载任何行，MobiLink 服务器不调用该表的上载脚本，即使已定义了这些脚本也如此。此选项会覆盖缺省行为并使 MobiLink 服务器调用表的上载脚本，而不管是否上载了行。

-zw 选项

控制要显示的警告消息级别。

语法

mlsrv11 -c "*connection-string***" -zw** *levels*

注释

MobiLink 的警告消息有五个级别：

级别	说明
0	取消所有警告消息
1	服务器和高 ODBC 级：启动 MobiLink 服务器时的警告消息
2	同步和用户级：启动同步时的警告消息
3	模式级：MobiLink 服务器处理客户端模式时的警告消息
4	脚本和低 ODBC 级：MobiLink 服务器读取、准备或执行脚本时的警告消息
5	表或行级：MobiLink 服务器在上载或下载期间执行表操作时的警告消息

若要指定需报告出的警告消息级别，可以用逗号将不同级别隔开，或者中间用两个点表示一系列级别。例如，**-zw 1..3,5** 与 **-zw 1,2,3,5** 相同。

报告消息对性能稍有影响。以较大的数字表示的级别往往会生成较多消息。

如果 -zw 在同一命令行中使用了多次，MobiLink 仅识别最后一个实例。如果 -zw、-zwd 和 -zwe 的设置冲突，MobiLink 优先使用 -zwe，然后是 -zwd，最后是 -zw。

缺省设置是 **1,2,3,4,5**，表示显示所有级别的警告消息。

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

-zwd 选项

禁用特定的警告代码。

语法

mlsrv11 -c "*connection-string***" -zwd** *code*, ...

注释

您可禁用特定的警告代码，使这些代码不被报告出，但同一级别的其它代码仍会报告出。

有关警告消息代码的完整列表，请参见“MobiLink 服务器警告消息”《错误消息》。

如果在同一命令行中多次使用 -zwd，MobiLink 将合并使用这些设置。如果 -zw、-zwd 和 -zwe 的设置冲突，MobiLink 优先使用 -zwe，然后是 -zwd，最后是 -zw。

-zwe 选项

启用特定的警告代码。

语法

mlsrv11 -c "*connection-string***" -zwe** *code*, ...

注释

可以启用特定的警告代码，使您在使用 -zw 禁用同一级别的其它代码的情况下仍能报告这些代码。

有关警告消息代码的完整列表，请参见“MobiLink 服务器警告消息” 《错误消息》。

如果在同一命令行上多次使用 -zwe，MobiLink 将合并使用这些设置。如果 -zw、-zwd 和 -zwe 的设置冲突，MobiLink 优先使用 -zwe，然后是 -zwd，最后是 -zw。

Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

同步技术

目录

MobiLink 开发提示 120
基于时间戳的下载 121
快照同步 125
在远程数据库之间对行进行分区 127
仅上载同步和仅下载同步 130
维护唯一主键 131
冲突处理 137
强制冲突 145
数据输入 146
处理删除 147
处理失败的下载 149
下载确认 152
从存储过程调用中下载结果集 153
从自引用表上载数据 155
MobiLink 隔离级别 156

MobiLink 开发提示

向应用程序中添加同步功能的同时也会增加应用程序的复杂程度。以下提示或许对您有所帮助。

将同步添加到原型应用程序中时，将很难发现哪些组件是问题的根源所在。因此在开发原型时，您可以在应用程序中临时增加一些硬编码的 INSERT 语句以便为测试和演示提供数据。在原型可以正常工作后，启用同步并删除临时 INSERT 语句。

开始时先使用简单的同步技术。诸如仅需要一两个脚本的简单上载或下载操作。在这些脚本可以正常工作之后，再引入更高级的技术，如时间戳、主键池和冲突解决。

MobiLink 和主键

在同步系统中，主键是标识不同数据库（远程数据库和统一数据库）中相同行的唯一方式，也是检测冲突的唯一方式。因此，MobiLink 应用程序必须遵守以下规则：

- 每个要同步的表都必须有一个主键。
- 绝不要更新主键的值。
- 主键在所有已同步的数据库中都必须是唯一的。

  请参见“维护唯一主键”一节第 131 页。

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

基于时间戳的下载

时间戳方法是进行高效同步的最实用的通用技术。此方法将跟踪每个用户上次同步的时间并仅下载在该时刻后更改的行。

MobiLink 维持一个时间戳值，用来指示每个 MobiLink 用户上次下载数据的时间。此值被称为**上次下载时间**。

请参见“在脚本中使用上次下载时间”一节第 122 页。

♦ 实现表的基于时间戳的同步

1. 在统一数据库中，添加一列以保存行的最近修改时间。通常可以按照如下方式声明该列：

DBMS	最近修改的列
Adaptive Server Enterprise	`datetime`
IBM DB2 LUW	`timestamp`
IBM DB2 主机	`timestamp`
Microsoft SQL Server	`datetime`
MySQL	`timestamp`
Oracle	`date`
SQL Anywhere	`timestamp DEFAULT timestamp`

2. 在 download_cursor 和 download_delete_cursor 事件的脚本中，将第一个参数与时间戳列中的值进行比较。

示例

如下表声明及脚本将实现 Contact 示例中 Customer 表的基于时间戳的同步：

- 表定义：

```
CREATE TABLE "DBA"."Customer"(
   "cust_id"  integer NOT NULL DEFAULT GLOBAL AUTOINCREMENT,
   "name"  char(40) NOT NULL,
   "rep_id"  integer NOT NULL,
   "last_modified" timestamp NULL DEFAULT timestamp,
   "active"  bit NOT NULL,
   PRIMARY KEY ("cust_id") )
```

- download_delete_cursor 脚本：

```
SELECT cust_id
FROM Customer JOIN SalesRep
ON Customer.rep_id = SalesRep.rep_id
WHERE Customer.last_modified >= {ml s.last_table_download}
```

```
AND ( SalesRep.ml_username != {ml s.username}
      OR Customer.active = 0 )
```

- download_cursor 脚本：

```
SELECT cust_id, Customer.name, Customer.rep_id
FROM Customer KEY JOIN SalesRep
WHERE Customer.last_modified >= {ml s.last_table_download}
    AND SalesRep.ml_username = {ml s.username}
    AND Customer.active = 1
```

请参见“同步逻辑源代码”一节 《MobiLink - 入门》和“同步 Contact 示例中的联系人”一节 《MobiLink - 入门》。

在脚本中使用上次下载时间

上次下载时间戳将作为参数提供给许多 MobiLink 事件。在下载阶段前面的最后一次成功同步过程中，从统一数据库中获取的时间值即为上次下载时间戳。如果当前 MobiLink 用户从未进行过同步或从未成功进行过同步，此值将被设置为 1900-01-01。

请参见“如何生成和使用下载时间戳”一节第 122 页。

如果有多个发布并在不同的时间同步它们，则可以拥有两个不同的上次下载时间戳。因此，上次下载时间戳有两个脚本参数名：

- **last_table_download** 是表的上次下载时间戳。
- **last_download** 是上次同步所有表的时间。它是所有表最早的 last_table_download 值。

如果在 MobiLink 脚本中使用问号而非命名参数，则始终会使用正确的值。

小心
如果您正在使用 SQL Anywhere 统一数据库，而且保存上次修改信息的列的类型为 DEFAULT TIMESTAMP，则不应同步该列。如果您的远程数据库需要这样的列，应使用不同的列名。否则，缺省的时间戳值可能被上载值覆盖，将不会包含上次在统一数据库中修改该行的时间。

另请参见

- “脚本参数”一节第 299 页

示例

```
SELECT cust_id, Customer.name, Customer.rep_id
FROM Customer KEY JOIN SalesRep
WHERE Customer.last_modified >= {ml s.last_table_download}
    AND SalesRep.ml_username = {ml s.username}
    AND Customer.active = 1
```

如何生成和使用下载时间戳

MobiLink 按以下方式生成时间戳并将其用于基于时间戳的下载：

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

- 在提交上载后、调用 prepare_for_download 事件前，MobiLink 服务器从统一数据库读取当前时间并保存该值。此时间戳值表示当前下载的开始时间；下一次同步只需下载此时刻后更改的数据。
- MobiLink 服务器将发送此时间戳值作为下载的一部分，并且客户端会将其存储下来。
- 客户端下次同步时，将使用与上载一同发送的 **last_download_timestamp** 的时间戳值。
- MobiLink 服务器将把客户端刚刚上载的 last_download_timestamp 传递到 download_cursor 和 download_delete_cursor。然后游标可以选择包含晚于或等于上次 last_download_timestamp 时间戳的更改以确保仅下载新更改。

上次下载时间的存储位置

上次下载时间存储在远程数据库中。这个位置很合适，因为只有远程数据库知道是否已成功应用下载。

如果是 SQL Anywhere 远程数据库，会在每次预订时存储上次下载时间。请参见“SYSSYNC 系统视图”一节 《SQL Anywhere 服务器 - SQL 参考》。

如果是 UltraLite 远程数据库，会在每次发布时存储上次下载时间。请参见“syspublication 系统表”一节 《UltraLite - 数据库管理和参考》。

更改上次下载时间

在极少数情况下，您可能想要修改 last_download_timestamp。例如，如果意外删除了远程数据库上的所有数据，可通过定义 modify_last_download_timestamp 连接脚本来重置上次下载时间戳值，从而重新同步远程数据库。还有另一个事件（名为 modify_next_last_download_timestamp）可用来重置下次同步时间戳，而不是当前同步时间戳。请参见：

- “modify_last_download_timestamp 连接事件”一节第 430 页
- “modify_next_last_download_timestamp 连接事件”一节第 433 页

UltraLite 还提供更改远程数据库中的上次下载时间的功能。请参见：

- C/C++ 嵌入式 SQL： “ULResetLastDownloadTime 函数”一节 《UltraLite - C 及 C++ 编程》
- C++： “ResetLastDownloadTime 函数”一节 《UltraLite - C 及 C++ 编程》
- .NET 2.0： “ResetLastDownloadTime 方法”一节 《UltraLite - .NET 编程》

另请参见

- “在脚本中使用上次下载时间”一节第 122 页

处理 daylight savings time

如果在更改时间期间同步数据，夏令时会在分布式数据库系统中引发问题。事实上，您可能会丢失数据。仅当在秋季时间恢复并有一个小时的模糊时间时，才发生此问题。

要处理 daylight savings time，有三种可能的解决方案：

- 确保统一数据库服务器使用 UTC 时间。

- 在统一数据库服务器上关闭 daylight savings time。
- 当时间更改时关机一小时。

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

快照同步

基于时间戳的同步适用于大多数同步。不过，有时候您可能需要更新数据快照。

表的快照同步是指完全下载表中的所有相关行，即使以前已经下载过。这是最简单的同步方法，但会引起大量的不必要的数据集交换，从而导致性能下降。

您可以使用快照同步下载表中的所有行，或者将此方法与行分区方法结合使用。请参见“在远程数据库之间对行进行分区”一节第 127 页。

何时使用快照同步

快照同步最适合用于同时具有以下两种特征的表。

- **行数相对较少** 当表中的行数比较少时，下载所有行所产生的开销也会相对较小。
- **行信息经常更改** 当表中的大多数行频繁进行更改时，采用显式排除上次同步后未发生更改的行的方法就没有太大的必要了。

用于保存汇率列表的表非常适合采用这种方法，因为货币的币种相对较少，而大多数汇率的变动却非常频繁。根据公司业务的特点，包含价格、利率列表或最新新闻条目的表都适合采用这种方法。

♦ 实现基于快照的同步

1. 除非远程用户更新值，否则不要定义上载脚本。
2. 如果表中可能有已删除的行，请编写 download_delete_cursor 脚本删除远程表中的所有行，或至少删除不再需要的所有行。不要从统一数据库中删除行；而应将其标记为待删除。而且在从远程数据库中删除行时，必须清楚了解该行中的值。

   请参见“编写 download_delete_cursor 脚本”一节第 314 页。
3. 编写 download_cursor 脚本，选择要包括在远程表中的所有行。

删除行

最好不要删除统一数据库中的行，但可以将其标记为待删除。而且在从远程数据库中删除行时，必须清楚了解该行中的值。在 download_cursor 脚本中仅选择未标记的行，在 download_delete_cursor 脚本中仅选择已标记的行。

download_delete_cursor 脚本在 download_cursor 脚本之前执行。如果要将某行包含在下载中，则在删除列表中不必包含具有相同主键的行。在远程位置接收到某个下载行时，它将取代已存在的具有相同主键的行。

请参见“编写用于下载行的脚本”一节第 312 页。

其它可用的删除技术

您可以允许远程应用程序删除行，而不是使用 download_cursor 脚本从远程数据库中删除行。例如，您可以使用应用程序在同步完成后立即执行 SQL 语句以删除不需要的行。

应用程序删除的行通常在下一次同步时上载到 MobiLink 服务器，但您可以使用 STOP SYNCHRONIZATION DELETE 语句阻止此上载操作。例如，

```
STOP SYNCHRONIZATION DELETE;
DELETE FROM table-name
 WHERE expiry_date < CURRENT TIMESTAMP;
COMMIT;
START SYNCHRONIZATION DELETE;
```

请参见“编写 download_delete_cursor 脚本”一节第 314 页。

快照示例

使用快照同步对示例应用程序中的 ULProduct 表进行维护。此表中的行数较少，因此使用快照同步的开销也较小。

1. 没有上载脚本。这表明不能在远程数据库添加产品的业务决策。
2. 没有 download_delete_cursor 脚本，反映了未从列表中删除产品这一假设。
3. download_cursor 脚本选择当前每种产品的产品标识符、价格和名称。如果产品已预先存在，则在远程表中更新价格。如果为新产品，则在远程表中插入一行。

   ```
   SELECT prod_id, price, prod_name
   FROM ULProduct
   ```

有关对仅包含少数行的表进行快照同步的另一示例，请参见“同步 Contact 示例中的销售代表”一节 《MobiLink - 入门》 。

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

在远程数据库之间对行进行分区

每个 MobiLink 远程数据库都可以包含不同的统一数据库数据子集。这意味着您可以编写自己的同步脚本，这样就可在远程数据库间对数据进行**分区**。

分区可以是无交集的，也可以包含交集。例如，如果每个员工都拥有其自己的客户集，而没有共享的客户，则分区将是**无交集**的。如果存在共享客户，这些客户出现在多个远程数据库中，则分区中会包含**交集**。

分区是在表的 download_cursor 和 download_delete_cursor 脚本中实现的，这两个脚本定义要下载到远程数据库中的行。每个脚本都将 MobiLink 用户名作为参数。通过在 WHERE 子句中使用此参数定义脚本，每个用户都可获得相应的行。

无交集分区

分区由同步中涉及的每个表的 download_cursor 和 download_delete_cursor 脚本控制。这些脚本采用两个参数：上次下载时间戳和同步调用中提供的 MobiLink 用户名。

♦ 在远程数据库之间对表进行分区

1. 在表定义中应包括一个包含统一数据库中的同步用户名的列。您不必将此列下载到远程数据库。
2. 在 download_cursor 和 download_delete_cursor 脚本的 WHERE 子句中包括一个条件，要求此列与脚本参数相匹配。

   脚本参数在脚本中可使用问号或命名参数来表示。例如，下面的 download_cursor 脚本根据雇员 ID 对 Contact 表进行分区。

   ```
   SELECT id, contact_name
   FROM Contact
   WHERE last_modified >= {ml s.last_table_download}
   AND emp_id = {ml s.username}
   ```

   请参见“download_cursor 表事件”一节第 370 页和“download_delete_cursor 表事件”一节第 373 页。

示例

CustDB 示例应用程序中的主键池表用于为每个远程数据库提供各自的主键值集。此技术可用于避免出现重复的主键，在“使用主键池”一节第 134 页中有对此技术的详细论述。

使用此方法的一个必要条件是，必须以无交集方式在远程数据库之间对主键池表进行分区。

一个键池表是 ULCustomerIDPool，此表包含每个用户在添加客户时可以使用的主键值。此表包含三列：

- **pool_cust_id** 在 ULCustomer 表中使用的主键值。此列是唯一下载到远程数据库中的列。
- **pool_emp_id** 拥有此主键的员工。
- **last_modified** 此表是使用时间戳技术并基于 last_modified 列来进行维护的。

Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

有关时间戳同步的信息，请参见“基于时间戳的下载”一节第 121 页。

此表的 download_cursor 脚本如下所示。

```
SELECT pool_cust_id
FROM ULCustomerIDPool
WHERE last_modified >= {ml s.last_table_download}
 AND pool_emp_id = {ml s.username}
```

如果不使用变量或指定参数，则可以使用包含 **?** 占位符的连接或子选择。

请参见“同步 Contact 示例中的客户”一节 《MobiLink - 入门》和“同步 Contact 示例中的联系人”一节 《MobiLink - 入门》。

有交集分区

统一数据库中的一些表中可能包含属于多个远程数据库的行。每个远程数据库都拥有一个统一数据库中的行的子集，并且该子集与其它远程数据库之间存在交集。客户表便是这种情况。这时，该表与远程数据库之间存在多对多的关系，而且通常具有一个描述这种关系的表格。download_cursor 和 download_delete_cursor 事件的脚本需要将正在下载的表连接到关系表。

示例

CustDB 示例应用程序在 ULOrder 表中使用此技术。ULEmpCust 表保存 ULCustomer 和 ULEmployee 之间的多对多关系的信息。

每个远程数据库只从 ULOrder 表中接收那些其 emp_id 列与 MobiLink 用户名相匹配的行。

在 CustDB 应用程序中，用于 ULOrder 的 download_cursor 脚本的 SQL Anywhere 版本如下：

```
SELECT o.order_id, o.cust_id, o.prod_id,
   o.emp_id, o.disc, o.quant, o.notes, o.status
FROM ULOrder o , ULEmpCust ec
WHERE o.cust_id = ec.cust_id
 AND ec.emp_id = {ml s.username}
 AND ( o.last_modified >= {ml s.last_table_download}
  OR ec.last_modified >= {ml s.last_table_download})
 AND  ( o.status IS NULL
  OR o.status != 'Approved' )
 AND ( ec.action IS NULL )
```

此脚本相当复杂。它说明在远程数据库中定义了一个表的查询中可以包含统一数据库中的多个表。脚本将下载 ULOrder 中满足以下条件的所有行：

- ULOrder 表中的 cust_id 列与 ULEmpCust 表中的 cust_id 列相匹配
- ULEmpCust 表中的 emp_id 列与同步用户名相匹配
- 订单或员工-客户关系的上次修改时间晚于此用户的最近同步的时间
- 状态不是 **Approved**

ULEmpCust 上的操作列用于标记要删除的列。其目的与当前主题并不相关。

下面是 download_delete_cursor 脚本。

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

```
SELECT o.order_id
FROM ULOrder o, ULEmpCust ec
WHERE o.cust_id = ec.cust_id
  AND ec.emp_id = {ml s.username}
  AND ( o.last_modified >= {ml s.last_table_download} OR
        c.last_modified >= {ml s.last_table_download} )
  AND ( o.status IS NULL OR
        o.status != 'Approved' )
  AND ( ec.action IS NULL )
```

此脚本从远程数据库中删除所有已核准的行。

对子表进行分区

上一节中的示例说明了如何根据其它表的条件对表分区。请参见“有交集分区”一节第 128 页。

在远程数据库中，某些表可能包含无交集子集或有交集子集，但并不包含用于确定子集的列。这些子表通常包含一个外键（或一系列外键），而外键引用另一个表。被引用的表中包含用于确定正确子集的列。

在这种情况下，download_cursor 脚本和 download_delete_cursor 脚本需要连接被引用的表，并且使用 WHERE 子句将行限定为正确的子集。

有关示例内容，请参见“同步 Contact 示例中的联系人”一节 《MobiLink - 入门》。

仅上载同步和仅下载同步

缺省情况下，同步为双向：数据可以上载，也可以下载。但是，您可以选择仅执行上载或仅执行下载。

同步模型注释
此主题针对在数据库中创建 MobiLink 同步系统时建立仅上载和仅下载同步的方法提供有关信息。在 Sybase Central 中创建同步模型时，您也可以指定仅上载或仅下载。

SQL Anywhere 远程数据库

- **上载** 要执行仅上载同步，可使用 dbmlsync 选项 -uo 或扩展选项 UploadOnly。请参见：
  - “-uo 选项”一节 《MobiLink - 客户端管理》
  - “UploadOnly (uo) 扩展选项”一节 《MobiLink - 客户端管理》
- **下载** 要执行仅下载同步，可使用 dbmlsync 选项 -ds 或扩展选项 DownloadOnly。请参见：
  - “-ds 选项”一节 《MobiLink - 客户端管理》
  - “DownloadOnly (ds) 扩展选项”一节 《MobiLink - 客户端管理》

  SQL Anywhere 远程数据库也可以使用仅下载发布。此下载方法不同于仅下载同步。请参见“仅下载发布”一节 《MobiLink - 客户端管理》。

UltraLite 远程数据库

- **上载** 使用仅上载同步参数来执行仅上载同步。

  请参见“Upload Only 同步参数”一节 《UltraLite - 数据库管理和参考》。
- **下载** 使用仅下载同步参数来执行仅下载同步。

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

Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

维护唯一主键

每个要同步的表必须有一个主键，并且该主键在所有已同步的数据库中必须是唯一的。不应更新主键的值。

使用单列作为表的主键通常是非常便利的。例如，您应该为每个客户指派唯一的标识值。如果所有销售代表工作的环境都与数据库有直接的连接，则很容易指派这些数值。每当在客户表中插入新的客户时，将自动添加一个大于上一个主键值的新主键值。

在未连接的环境中，要在插入新行时为主键指派一个唯一值并不那么容易。当销售代表添加一个新客户时，需要在 Customer 表的一个远程副本中执行此操作。您必须防止那些对其它 Customer 表副本进行操作的销售代表使用相同的客户标识值。

本节介绍了以下几种方法来解决唯一主键生成方面的问题：

- “使用组合键”一节第 131 页
- “使用 UUID”一节第 131 页
- “使用全局自动增量”一节第 132 页
- “使用主键池”一节第 134 页

使用组合键

MobiLink 远程 ID 可唯一定义同步系统中的远程数据库。因此，创建唯一主键的一个简便方法就是创建一个组合主键，其中包含 MobiLink 远程 ID 作为其值的一部分。如果要维持唯一的 MobiLink 用户名，则应使用该用户名而不使用远程 ID。

请参见“远程 ID”一节 《MobiLink - 客户端管理》。

使用 UUID

通过使用 newid( ) 函数为主键创建通用唯一值，可以确保主键的唯一性。使用 uuidtostr( ) 函数可以将生成的 UUID 转换为字符串，使用 strtouuid( ) 函数可以将其转换回二进制形式。

UUID（也称为 GUID）在所有计算机中都是唯一的。不过，它们的值是完全随机的，因此不能用来确定添加值的时间或值的顺序。另外，UUID 值比其它方法（包括全局自动增量）所需的值大得多，并且在主键表和外键表中都需要更多的表空间。使用 UUID 的表索引的效率也更低。

另请参见

SQL Anywhere 数据库：

- “NEWID 缺省值”一节 《SQL Anywhere 服务器 - SQL 的用法》
- “NEWID 函数 [Miscellaneous]”一节 《SQL Anywhere 服务器 - SQL 参考》
- “UNIQUEIDENTIFIER 数据类型”一节 《SQL Anywhere 服务器 - SQL 参考》

Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

UltraLite 数据库：

- “UltraLite 中的主键唯一性”一节 《UltraLite - 数据库管理和参考》
- “NEWID 函数 [Miscellaneous]”一节 《UltraLite - 数据库管理和参考》

示例

下面的 SQL Anywhere CREATE TABLE 语句创建一个通用唯一的主键：

```
CREATE TABLE customer (
   cust_key UNIQUEIDENTIFIER NOT NULL
            DEFAULT NEWID( ),
   rep_key VARCHAR(5),
   PRIMARY KEY(cust_key))
```

使用全局自动增量

在 SQL Anywhere 和 UltraLite 数据库中，可以将缺省列值设置为 GLOBAL AUTOINCREMENT。您可以对要在其中保持唯一值的任何列使用此缺省值，但它对主键特别有用。

♦ 使用全局自动增量列

1. 将该列声明为全局自动增量列。

   如果您指定了缺省全局自动增量，则将对该列的值域进行分区。每个分区都包含相同数目的值。例如，如果将数据库中一个整数列的分区大小设置为 1000，则一个分区的范围是从 1001 到 2000，下一个分区从 2001 到 3000，依此类推。

   请参见“声明缺省全局自动增量”一节第 132 页。

2. 设置 global_database_id 值。

   SQL Anywhere 只从用数据库编号唯一标识的分区中提供数据库中的缺省值。例如，如果指定数据库的标识号为 10，分区大小为 1000，则将在 10001-11000 范围内选择该数据库的缺省值。标识号指派为 11 的另一个数据库副本，将在 11001-12000 范围内为同一列提供缺省值。

   请参见“设置全局数据库 ID”一节第 133 页。

声明缺省全局自动增量

您可以在数据库中设置缺省值，具体方法是：在 Sybase Central 中选择列属性，或在 CREATE TABLE 或 ALTER TABLE 语句中包含 DEFAULT GLOBAL AUTOINCREMENT 短语。

或者，也可以在紧随 AUTOINCREMENT 关键字之后的括号中指定分区大小。分区大小可以是任意正整数，但分区大小的选择一般需要保证任何一个分区内的编号资源不被用尽，或极少用尽。

对于 INT 或 UNSIGNED INT 类型的列，缺省分区大小是 $2^{16} = 65536$；对于其它类型的列，缺省分区大小是 $2^{32} = 4294967296$。由于这些缺省值可能不合适（尤其当列不是 INT 或 BIGINT 类型时），因此最好显式地指定分区大小。

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

例如，以下的 SQL 语句创建一个简单的表，其中包含两列：用于保存客户标识号的整数列和用于保存客户名称的字符串列。分区大小设置为 5000。

```
CREATE TABLE customer (
 id   INT          DEFAULT GLOBAL AUTOINCREMENT (5000),
 name VARCHAR(128) NOT NULL,
 PRIMARY KEY (id)
)
```

另请参见

- SQL Anywhere： “CREATE TABLE 语句”一节 《SQL Anywhere 服务器 - SQL 参考》
- UltraLite： “UltraLite CREATE TABLE 语句”一节 《UltraLite - 数据库管理和参考》

设置全局数据库 ID

在部署应用程序时，必须为每个数据库指派一个不同的标识号。您可以通过多种方法来创建和分发标识号。一种方法是将值放置在表中，并根据某些其它的唯一属性（例如远程 ID）将合适的行下载到每个数据库。

♦ 设置全局数据库标识号

- 在 SQL Anywhere 中，可通过设置公共选项 global_database_id 的值来设置数据库的全局 ID。该标识号必须为非负的整数。请参见“global_database_id 选项 [数据库]”一节 《SQL Anywhere 服务器 - 数据库管理》。

  在 UltraLite 中，可通过设置 global_id 选项来设置数据库的全局 ID。请参见“UltraLite global_database_id 选项”一节 《UltraLite - 数据库管理和参考》。

如何选择缺省值

全局数据库 ID 在 SQL Anywhere 中使用公共选项 global_database_id 来设置，而在 UltraLite 中使用 global_id 选项来设置。

每个数据库中的全局数据库 id 选项都必须设置为唯一的非负整数。某一特定数据库的缺省值范围是从 $pn + 1$ 到 $p(n + 1)$，其中 p 为分区大小，而 n 为全局数据库 ID 的值。例如，如果分区大小为 1000 而全局数据库 ID 设置为 3，则范围就是从 3001 到 4000。

SQL Anywhere 和 UltraLite 使用以下规则选择缺省值：

- 如果在当前分区中列不包含任何值，则第一个缺省值为 $pn + 1$，其中 p 为分区大小，而 n 为全局数据库 ID 的值。
- 如果列包含当前分区中的值，但都小于 $p(n + 1)$，则下一个缺省值为该范围内上一个最大值加 1。
- 缺省列值不受列中当前分区外的值的影响；也就是说，不受小于 $pn + 1$ 或大于 $p(n + 1)$ 的值的影响。如果通过 MobiLink 同步从另一个数据库中复制了这些值，则这些值就可能存在。

如果全局数据库 ID 设置为缺省值 2147483647，则将向该列插入一个空值。如果不允许空值，则插入行的尝试将导致错误。例如，如果列包含在表的主键中，便会发生这种情况。

因为全局数据库 ID 不能设置为负值，所以所选值始终为正数。最大标识号仅受列数据类型和分区大小的限制。

当分区内的可用值用完时，也会生成空缺省值。在此情况下，应向数据库指派一个新全局数据库 ID 值，以允许从其它分区选择缺省值。如果该列不允许使用空值，则插入空值的尝试将导致错误。要检测提供的未使用值是否太小并处理此情况，可创建一个 GlobalAutoincrement 类型的事件。

如果一个特定分区中的值已接近用完，您可以为该数据库指派一个新的全局数据库 ID。您可以使用任何简便的方式指派新的数据库 ID。但是，一种可行的方法是维护未使用的数据库 ID 值的池。该池的维护与主键池的维护方法相同。

可以设置一个事件处理程序，在分区快要用尽时自动通知数据库管理员（或执行其它操作）。有关 SQL Anywhere 数据库的信息，请参见“定义事件的触发条件”一节 《SQL Anywhere 服务器 - 数据库管理》。

另请参见

- “设置全局数据库 ID”一节第 133 页
- SQL Anywhere： “global_database_id 选项 [数据库]”一节 《SQL Anywhere 服务器 - 数据库管理》
- UltraLite： “UltraLite global_database_id 选项”一节 《UltraLite - 数据库管理和参考》

示例

在 SQL Anywhere 数据库中，以下语句将数据库标识号设置为 20。

```
SET OPTION PUBLIC.global_database_id = 20
```

如果特定列的分区大小为 5000，则在 100001-105000 范围内选择此数据库的缺省值。

使用主键池

解决此唯一主键问题的一个有效方法是为数据库的每个用户指派一个可在需要时使用的主键值池。例如，您可以为每个销售代表指派 100 个新标识值。这样，每个销售代表就可以为新客户任意指派自己池中的值。

♦ 实现主键池

1. 在统一数据库和每个远程数据库中添加一个新表来保存新的主键池。除了用于保存唯一值的列，这些表中还应包含一个用户名列，以标识哪些用户已被授予指派该值的权限。
2. 编写存储过程以确保为每个用户都指派了足够多的新标识值。为那些插入许多新条目或不经常执行同步的远程用户指派更多的新值。
3. 编写 download_cursor 脚本，以选择指派给每个用户的新值，并将它们下载到远程数据库中。
4. 修改将使用远程数据库的应用程序，以便在用户插入新行时应用程序使用池中的其中一个值。然后，应用程序必须从池中删除该值，以确保不会再次使用它。
5. 编写上载脚本。然后，MobiLink 服务器从统一值池中删除那些用户已从远程数据库中自己的值池中删除的行。

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

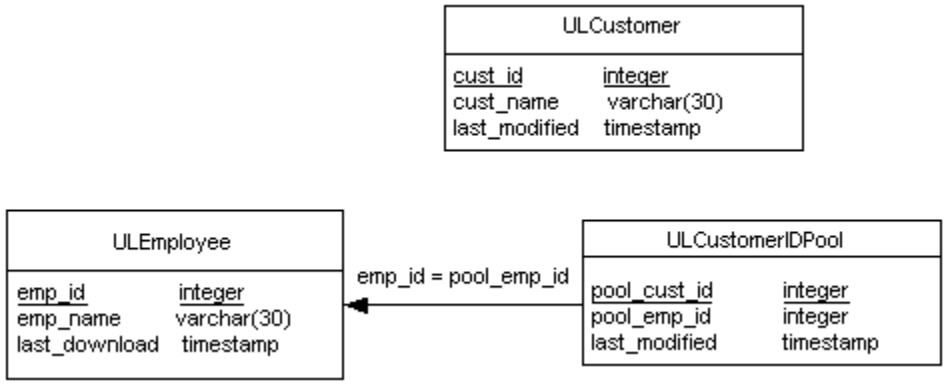
6. 编写 end_upload 脚本以调用用来维护值池的存储过程。这样做将向用户的池中添加更多的值以代替在上载过程中删除的值。

示例

该示例应用程序允许远程用户添加客户。应确保每个新行都有唯一的主键值，而且在输入数据时所有远程数据库的连接都必须是断开的。

ULCustomerIDPool 包含可由各远程数据库使用的主键值的列表。另外，ULCustomerIDPool_maintain 存储过程在值用完时将池加满。维护过程由表级 end_upload 脚本调用，而各个远程数据库上的池由 upload_insert 和 download_cursor 脚本维护。

1. 统一数据库中的 ULCustomerIDPool 表用于保存新客户标识号池。它与 ULCustomer 表之间不存在直接链接。



2. ULCustomerIDPool_maintain 过程将更新统一数据库中的 ULCustomerIDPool 表。以下示例代码适用于 SQL Anywhere 统一数据库。

```
CREATE PROCEDURE ULCustomerIDPool_maintain ( IN syncuser_id INTEGER )
BEGIN
    DECLARE pool_count INTEGER;

    -- Determine how may ids to add to the pool
    SELECT COUNT(*) INTO pool_count
    FROM ULCustomerIDPool
    WHERE pool_emp_id = syncuser_id;

    -- Top up the pool with new ids
    WHILE pool_count < 20 LOOP
      INSERT INTO ULCustomerIDPool ( pool_emp_id )
      VALUES ( syncuser_id );
      SET pool_count = pool_count + 1;
    END LOOP;
END
```

此过程对目前指派给当前用户的号码进行计数并插入新行，以确保此用户具有足够的客户标识号。

此过程由 ULCustomerIDPool 表的 end_upload 表脚本在上载结束时调用。该脚本如下所示：

```
CALL ULCustomerIDPool_maintain( {ml s.username} )
```

3. ULCustomerIDPool 表的 download_cursor 脚本将新的标识号下载到远程数据库中。

```
SELECT pool_cust_id
FROM ULCustomerIDPool
WHERE pool_emp_id = {ml s.username}
AND last_modified >= {ml s.last_table_download}
```

4. 要插入新客户，使用远程数据库的应用程序必须从池中选择一个未使用的标识号，并从池中删除此编号，然后使用此标识号插入新客户的信息。以下 UltraLite 应用程序的嵌入式 SQL 函数将从池中检索一个新客户号。

```
bool CDemoDB::GetNextCustomerID( void )
/************************************/
{
    short ind;

    EXEC SQL SELECT min( pool_cust_id )
    INTO :m_CustID:ind FROM ULCustomerIDPool;
    if( ind < 0 ) {
        return false;
    }
    EXEC SQL DELETE FROM ULCustomerIDPool
    WHERE pool_cust_id = :m_CustID;
    return true;
}
```

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

冲突处理

在向统一数据库中上载行时可能会发生冲突。如果两个用户修改不同远程数据库中相同的行，则当这两行中的第二个到达 MobiLink 服务器时将出现冲突。

缺省情况下，

- 如果在尝试插入一行时发现此行已经插入，则会导致错误。
- 如果在尝试删除一行时发现该行已经删除，则第二次尝试删除的操作将被忽略。

如果需要不同的行为，可通过定义一个或多个本节所述的上载事件来实现。

关于冲突

小心
切勿更新同步表中的主键。更新主键会导致主键的目标落空，因为主键是标识不同数据库（远程和统一数据库）中相同行的唯一方式并且是检测冲突的唯一方式。

冲突与错误是两个不同的概念。在冲突发生时，您应定义一个过程以计算正确值，或至少使用日志文件记录冲突。冲突处理是设计合理的应用程序中不可缺少的一部分。

在同步过程的下载阶段，远程数据库中不会出现冲突。如果下载的行中包含一个新的主键，则该行的值将插入到新的一行。如果该主键与一个现有行的主键相匹配，则会更新该行中的值。

示例

User1 开始时的库存为十件，卖出三件后将 Remote1 库存值更新为七件。User2 卖出 4 件并将 Remote2 库存更新为 6。Remote1 同步时，同步数据库将被更新为 7。Remote2 同步时，由于库存值不再是 10，因此会检测到冲突。若要以编程方式解决这一冲突，需要三个行值：

1. 统一数据库中的当前值。
2. Remote2 上载的新行值。
3. Remote2 在上一次同步期间获取的旧行值。

在这种情况下，业务逻辑可使用以下方法来计算新的库存值并解决冲突：

```
current consolidated - (old remote - new remote)
-> 7 - (10-6) = 3
```

有关如何处理冲突的其它示例，请参见：

- “同步 Contact 示例中的产品”一节 《MobiLink - 入门》

检测冲突

在 MobiLink 客户端向 MobiLink 服务器发送更新行时，其中不仅包含新的更新值（后映像），还包含上次下载的或在此行第一次上载前已存在的行值中获得的旧行值副本（前映像）。前映像与统一数据库中的当前值不匹配时，则会检测到冲突。

提供了多种脚本用于检测冲突。仅当使用以下脚本之一时，MobiLink 服务器才检测冲突：

- upload_fetch 或 upload_fetch_column_conflict 脚本。
- 在 WHERE 子句中包含所有非主键列的 upload_update 脚本。

使用 upload_fetch 脚本检测冲突

如果为表定义了 upload_fetch 或 upload_fetch_column_conflict 脚本，MobiLink 服务器会将 upload_fetch 脚本返回的行的更新值的前映像与相同主键的值进行比较。如果前映像中的值与当前统一数据库中的值不匹配，则 MobiLink 服务器会检测到冲突。

upload_fetch 脚本从统一数据库表中选择对应于正在更新的行的单行数据。典型的 upload_fetch 脚本的语法如下所示：

SELECT *pk1*, *pk2*, ...*col1*, *col2*, ...
FROM *table-name*
WHERE *pk1* **= {ml r.***pk1***} AND** *pk2* **= {ml r.***pk2***}** ...
  AND *col1* **= {ml r.***col1***} AND** *col2* **= {ml r.***col2***}** ...

请参见“upload_fetch 表事件”一节第 467 页。

upload_fetch_column_conflict 事件与 upload_fetch 相似，但它只检测两个用户更新同一列时的冲突。不同的用户可更新同一行，只要他们不更新同一列，就不会产生冲突。

请参见“upload_fetch_column_conflict 表事件”一节第 469 页。

您只能为远程数据库中的每个表编写一个 upload_fetch 脚本或一个 upload_fetch_column_conflict 脚本。

锁定统一数据库中的行

在 upload_fetch 脚本检测到冲突之后和冲突解决之前，统一数据库中的行可以发生更改。此问题会导致数据不正确，要避免此问题，可以实现带有行锁定的 upload_fetch 或 upload_fetch_column_conflict 脚本。

在 SQL Anywhere 统一数据库中，您可以使用 UPDLOCK 或 HOLDLOCK 关键字，但 UPDLOCK 更适合并发。例如：

```
SELECT column-names from table-name WITH (UPDLOCK)
  WHERE where-clause
```

对于 Oracle、DB2 LUW 和 DB2 主机，请使用 FOR UPDATE。例如：

```
SELECT column-names FROM table-name
  WHERE where-clause
    FOR UPDATE OF column_name1, column_name3, column_name6
```

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

注意
如上例所示，指定要更新的列名称可节省计算机资源并提高性能。

对于 Microsoft SQL Server，请使用 HOLDLOCK。例如，

```
SELECT column-names FROM table-name WITH (HOLDLOCK)
  WHERE where-clause
```

对于 Adaptive Server Enterprise，请使用 HOLDLOCK。例如，

```
SELECT column-names FROM table-name
  HOLDLOCK
  WHERE where-clause
```

示例

定义一个 upload_fetch 脚本。MobiLink 服务器使用此脚本在统一数据库中检索当前行并将该行与更新行的前映像进行比较。如果这两行包含相同的值，则没有冲突。如果两行的值不同，则检测到冲突，MobiLink 将调用 upload_old_row_insert 和 upload_new_row_insert 脚本，然后调用 resolve_conflict。

请参见“使用 resolve_conflict 脚本解决冲突”一节第 140 页。

使用 upload_update 脚本检测冲突

要使用 upload_update 脚本检测冲突，请在 WHERE 子句中包括所有列：

UPDATE *table-name*
SET *col1* **= {ml r.***col1***},** *col2* **= {ml r.** *col2* **}** ...
WHERE *pk1* **= {ml o.***pk1***} AND** *pk2* **= {ml o.***pk2***}** ...
  AND *col1* **= {ml o.***col1***} AND** *col2* **= {ml o.***col2***}** ...

在此语句中，col1、col2 等是非主键列，而 pk1、pk2 等是主键列。传递到第二组非主键列 (o.) 的值是更新行的前映像（或旧值）。WHERE 子句将比较从远程数据库上载的旧值与统一数据库中的当前值。如果两个值不匹配，将忽略更新，保留统一数据库中已有的值。

请参见“upload_update 表事件”一节第 488 页。

仅当使用 upload_fetch 或 upload_fetch_column_conflict 没有检测到冲突时才使用 upload_update 脚本检测冲突。

第 1 种情形

为以下事件定义脚本：upload_update、upload_old_row_insert、upload_new_row_insert 和 resolve_conflict。

定义以下 upload_update 脚本：

```
UPDATE product
SET name={ml r.name}, description={ml r.description}
WHERE id={ml r.id}
   AND name={ml o.name}
   AND description={ml o.description}
```

MobiLink 执行更新，然后查看修改了多少行。如果没有行被修改，则 MobiLink 检测到冲突：统一数据库中没有行与前映像行匹配。MobiLink 将调用 upload_old_row_insert 和 upload_new_row_insert 脚本，然后调用 resolve_conflict。

请参见“使用 resolve_conflict 脚本解决冲突”一节第 140 页。

第 2 种情形

不为 upload_old_row_insert、upload_new_row_insert 和 resolve_conflict 定义脚本。而是创建一个存储过程来处理冲突检测和冲突解决，并在 upload_update 脚本中调用它。

请参见“使用 upload_update 脚本解决冲突”一节第 142 页。

解决冲突

解决冲突时有几种选择：

- 出现冲突时，使用临时表（或永久表）和 resolve_conflict 脚本加以解决。

  请参见“使用 resolve_conflict 脚本解决冲突”一节第 140 页。
- 出现冲突时，使用 upload_update 脚本加以解决。

  请参见“使用 upload_update 脚本解决冲突”一节第 142 页。
- 使用表的 end_upload 脚本立即解决所有冲突。

  请参见“end_upload 表事件”一节第 403 页。

使用 resolve_conflict 脚本解决冲突

当 MobiLink 服务器使用 upload_fetch 脚本检测到冲突后，将发生以下事件。

- MobiLink 服务器按照 upload_old_row_insert 脚本的定义插入从远程数据库上载的旧行值。通常将旧值插入到临时表中。

  请参见“upload_old_row_insert 表事件”一节第 476 页。
- MobiLink 服务器按照 upload_new_row_insert 脚本的定义插入从远程数据库上载的新行值。通常将新值插入到临时表中。

  请参见“upload_new_row_insert 表事件”一节第 473 页。
- MobiLink 服务器执行 resolve_conflict 脚本。在此脚本中，您可以调用存储过程，或定义一系列步骤以便使用新行值和旧行值来解决冲突。

有关详细信息，请参见“resolve_conflict 表事件”一节第 450 页。

示例

在下面的示例中，您将为 6 个事件创建脚本，然后创建一个存储过程。

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

- 在 begin_synchronization 脚本中，创建两个名为 contact_new 和 contact_old 的临时表。（也可在 begin_connection 脚本中执行此操作。）
- upload_fetch 脚本检测到冲突。
- 当存在冲突时，upload_old_row_insert 和 upload_new_row_insert 脚本将使用从远程数据库中上载的新旧数据来填充这两个临时表。
- resolve_conflict 脚本调用存储过程 MLResolveContactConflict 来解决此冲突。

事件	脚本

begin_synchronization	`CREATE TABLE #contact_new(`
	`  id  INTEGER,`
	`  location CHAR(36),`
	`  contact_date DATE);`
	`CREATE TABLE #contact_old(`
	`  id  INTEGER,`
	`  location CHAR(36),`
	`  contact_date DATE)`

upload_fetch	`SELECT id, location, contact_date`
	`  FROM contact`
	`  WHERE id = {ml r.id}`

upload_old_row_insert	`INSERT INTO #contact_new( id, location, contact_date )` `  VALUES ( {ml r.id}, {ml r.location}, {ml r.contact_date} )`
upload_new_row_insert	`INSERT INTO #contact_old( id, location, contact_date )` `  VALUES ( {ml r.id}, {ml r.location}, {ml r.contact_date} )`
resolve_conflict	`CALL MLResolveContactConflict( )`
end_synchronization	`DROP TABLE #contact_new;` `DROP TABLE #contact_old`

存储过程 MLResolveContactConflict 如下：

```
CREATE PROCEDURE MLResolveContactConflict( )
BEGIN
  --update the consolidated database only if the new contact date
  --is later than the existing contact date
  UPDATE contact c
    SET c.contact_date = cn.contact_date
      FROM #contact_new cn
      WHERE c.id = cn.id
        AND cn.contact_date > c.contact_date;
  --cleanup
  DELETE FROM #contact_new;
  DELETE FROM #contact_old;
END
```

使用 upload_update 脚本解决冲突

解决冲突时，您可以调用 upload_update 脚本中的存储过程而不是使用 resolve_conflict 脚本。使用此技术时，必须以编程方式检测并解决冲突。

存储过程必须使用带有 WHERE 子句的 upload_update 脚本格式，此 WHERE 子句包括除使用前映像值（即旧值）的列以外的所有列。

upload_update 脚本可能如下所示：

```
{CALL UpdateProduct(
   {ml o.id}, {ml o.name}, {ml o.desc}, {ml r.name}, {ml r.desc}
)
}
```

UpdateProduct 存储过程可能是：

```
CREATE PROCEDURE UpdateProduct(
  @id INTEGER,
  @preName VARCHAR(20),
  @preDesc VARCHAR(200),
  @postName VARCHAR(20),
  @postDesc VARCHAR(200) )
BEGIN
    UPDATE product
     SET name = @postName, description = @postDesc
     WHERE id = @id
       AND name = @preName
       AND description = @preDesc
    IF @@rowcount=0 THEN
        // A conflict occurred: handle resolution here.
    END IF
END
```

与使用 resolve_conflict 脚本解决冲突相比，此方法更易于维护，因为只有一个脚本需要维护，所有逻辑都包含在一个存储过程中。但是，如果表列可以为空，或其中包含 BLOB 或 CLOB，存储过程的代码可能会非常复杂。此外，对于一些受 MobiLink 统一数据库支持的 RDBMS，它们可以传递到存储过程的值有大小限制。

请参见：

- “使用 upload_update 脚本检测冲突”一节第 139 页
- “upload_update 表事件”一节第 488 页
- “使用 resolve_conflict 脚本解决冲突”一节第 140 页

示例

下面的存储过程 sp_update_my_customer 包含用于进行冲突检测和解决的逻辑。它接受旧列值和新列值。此示例使用 SQL Anywhere 功能。可按照以下方法实现此脚本。

```
{CALL sp_update_my_customer(
   {ml o.cust_1st_pk},
   {ml o.cust_2nd_pk},
   {ml o.first_name},
   {ml o.last_name},
   {ml o.nullable_col},
   {ml o.last_modified},
   {ml r.first_name},
```

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

```
   {ml r.last_name},
   {ml r.nullable_col},
   {ml r.last_modified}
)}
CREATE PROCEDURE sp_update_my_customer(
     @cust_1st_pk        INTEGER,
     @cust_2nd_pk        INTEGER,
     @old_first_name     VARCHAR(100),
     @old_last_name      VARCHAR(100),
     @old_nullable_col   VARCHAR(20),
     @old_last_modified  DATETIME,
     @new_first_name     VARCHAR(100),
     @new_last_name      VARCHAR(100),
     @new_nullable_col   VARCHAR(20),
     @new_last_modified  DATETIME
)
BEGIN
DECLARE @current_last_modified DATETIME;
// Detect a conflict by checking the number of rows that are
// affected by the following update. The WHERE clause compares
// old values uploaded from the remote to current values in
// the consolidated database. If the values match, there is
// no conflict. The COALESCE function returns the first non-
// NULL expression from a list, and is used in this case to
// compare values for a nullable column.


UPDATE my_customer
SET first_name                  = @new_first_name,
 last_name                      = @new_last_name,
 nullable_col                   = @new_nullable_col,
 last_modified                  = @new_last_modified

WHERE cust_1st_pk               = @cust_1st_pk
 AND cust_2nd_pk                = @cust_2nd_pk
 AND first_name                 = @old_first_name
 AND last_name                  = @old_last_name
 AND COALESCE(nullable_col, '') = COALESCE(@old_nullable_col, '')
 AND last_modified              = @old_last_modified;
...
// Use the @@rowcount global variable to determine
// the number of rows affected by the update. If @@rowcount=0,
// a conflict has occurred. In this example, the database with
// the most recent update wins the conflict. If the consolidated
// database wins the conflict, it retains its current values
// and no action is taken.

IF( @@rowcount = 0 ) THEN
// A conflict has been detected. To resolve it, use business
// logic to determine which values to use, and update the
// consolidated database with the final values.

  SELECT last_modified INTO @current_last_modified
     FROM my_customer WITH( HOLDLOCK )
     WHERE cust_1st_pk=@cust_1st_pk
        AND cust_2nd_pk=@cust_2nd_pk;

     IF( @new_last_modified > @current_last_modified ) THEN
     // The remote has won the conflict: use the values it
     // uploaded.

       UPDATE my_customer
       SET first_name    = @new_first_name,
         last_name     = @new_last_name,
```

```
            nullable_col  = @new_nullable_col,
            last_modified = @new_last_modified
          WHERE cust_1st_pk  = @cust_1st_pk
             AND cust_2nd_pk   = @cust_2nd_pk;

      END IF;
  END IF;
  END;
```

请参见：

- “COALESCE 函数 [Miscellaneous]”一节 《SQL Anywhere 服务器 - SQL 参考》
- “全局变量”一节 《SQL Anywhere 服务器 - SQL 参考》中的 @@rowcount

Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

强制冲突

强制冲突解决是一种特殊的技术，它强制将每个上载的行都作为冲突对待。

如果 upload_insert、upload_update 和 upload_delete 脚本均未定义，MobiLink 服务器将使用强制冲突解决。在这种操作模式下，MobiLink 服务器尝试使用由 upload_old_row_insert 和 upload_new_row_insert 脚本定义的语句来插入此表中的所有上载行。实质上，所有上载的行都将作为冲突来对待。您可以编写存储过程或脚本以任何您期望的方式处理上载的值。

如果脚本 upload_insert、upload_update 或 upload_delete 中有任何一个不存在，则将免去正常的冲突解决过程。此技术有两个主要用途。

- **任意冲突检测和解决** 自动机制在更新行过程中并且只有在出现旧值与统一数据库中的当前值不匹配的情况下才检测错误。

  您可以使用 upload_old_row_insert 和 upload_new_row_insert 脚本捕获原始上载数据，然后在适当的时候处理行。

- **性能** 如果没有定义 upload_insert、upload_update 和 upload_delete，则 MobiLink 服务器就可以免去正常的冲突检测任务（该任务涉及逐行查询统一数据库）。相反，它只需要使用由 upload_old_row_insert 和 upload_new_row_insert 脚本定义的语句插入原始上载信息。由于 MobiLink 服务器不通过网络读取行，因此性能得到了改善。

另请参见

- “强制冲突统计信息”一节第 183 页

数据输入

有些数据库中包含仅用于数据输入的表。处理这些表的一种方法是在每次同步过程中上载所有插入行，并在下载中将它们从远程数据库中删除。同步之后，远程数据库表再次为空，可以输入下一批数据。

要实现此模型，可以将行上载到临时表中，然后使用 end_upload 表脚本将它们插入到基表中。成功同步后，可以在 download_delete_cursor 中使用临时表从远程数据库中删除行。

或者，您可以允许客户端应用程序删除行，并使用 STOP SYNCHRONIZATION DELETE 语句在下一次同步过程中停止正在上载的删除操作。

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

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

处理删除

从统一数据库中删除行时，需要对该行进行记录，以便可以从任何包含该行的远程数据库中删除该行。可以使用逻辑删除或影子表两种方法来执行此操作。

- **逻辑删除** 使用此方法时，行并未删除。在状态列中将不再需要的数据标记为非活动。可在 download_cursor 和 download_delete_cursor 的 WHERE 子句中引用该行的状态。

  CustDB 示例应用程序在 ULEmpCust 表中也应用了此技术，该示例在操作列中使用 D 表示 [删除]。脚本使用该值从远程数据库中删除记录，并在同步完成时从统一数据库中删除该记录。CustDB 在 ULOrder 表中也使用了此技术，Contact 示例则在 Customer、Contact 和 Product 表中使用了此技术。

  MobiLink 同步模型支持逻辑删除，它会假定逻辑删除列仅存在于统一数据库中而不在远程数据库中。在将统一模式复制到新的远程模式时，将忽略所有与模型的同步设置中的逻辑删除列相匹配的列。对于新模型，将删除缺省列名称。

  将逻辑删除列名称添加至远程模式：

  1. 在 [**下载删除**] 页面的 [**创建同步模型向导**] 中，选择 [**使用逻辑删除**]。
  2. 重命名逻辑删除列，以使其不与统一数据库中的任何列名称相匹配。
  3. 完成向导后，更新远程模式并保留缺省表选择。逻辑删除列名称将出现在模式更改列表中，并将被添加到远程模式中。

  注意
  您需要设置列映射，将远程数据库的逻辑删除列映射到统一数据库的逻辑删除列。

- **影子表** 使用此方法时，将创建一个影子表来存储被删除的行的主键值。删除一行后，触发器将填充影子表。download_delete_cursor 可以使用影子表从远程数据库中删除行。影子表只需要包含真实表中的主键列。

请参见“编写 download_delete_cursor 脚本”一节第 314 页。

暂停删除同步

通常，SQL Anywhere 将自动记录对带有同步预订的发布中的表或列所做的所有更改。这些更改将在下次同步时上载到统一数据库。

但您可能希望从同步数据中删除行而不上载这些更改。可使用 STOP SYNCHRONIZATION DELETE 来完成此任务。此功能可用于异常纠正，但应谨慎使用，因为它将禁止部分自动同步功能。对使用 download_delete_cursor 脚本删除必要的行，此技术是一种实用的替代方法。

执行 STOP SYNCHRONIZATION DELETE 语句之后，不会同步在此连接上执行的任何删除操作。效应一直持续到执行 START SYNCHRONIZATION DELETE 语句为止。此效应并不嵌套，即首次执行 STOP SYNCHRONIZATION DELETE 后，再次执行时不会产生附加效应。

♦ **暂时禁止连接中发出的删除上载**

1. 执行下列语句停止对删除的自动记录。

   ```
   STOP SYNCHRONIZATION DELETE
   ```

2. 根据需要使用 DELETE 语句从同步数据中删除行。提交这些更改。
3. 使用下列语句重新启动对删除的记录。

   ```
   START SYNCHRONIZATION DELETE
   ```

删除的行不会传送到 MobiLink 服务器，且不会将这些行从统一数据库中删除。

另请参见

- SQL Anywhere 客户端： “STOP SYNCHRONIZATION DELETE 语句 [MobiLink]”一节《SQL Anywhere 服务器 - SQL 参考》
- UltraLite 客户端： “UltraLite STOP SYNCHRONIZATION DELETE 语句”一节 《UltraLite - 数据库管理和参考》

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

处理失败的下载

使用阻塞下载确认

现已不建议使用阻塞下载确认。应尽量使用非阻塞下载确认。

必须在下载事务中维护有关下载内容的簿记信息。此信息随将应用于远程数据库的下载一起自动更新。

如果在将整个下载应用到远程数据库之前发生故障，并且已将 SendDownloadAck 更改为 ON，则 MobiLink 服务器将不获取下载确认而回退下载事务。由于簿记信息是下载事务的一部分，因此它也将被回退。下次建立下载时会使用原始的簿记信息。

请参见“SendDownloadACK (sa) 扩展选项”一节 《MobiLink - 客户端管理》和“发送下载确认同步参数”一节 《UltraLite - 数据库管理和参考》。

测试同步脚本时，应在 end_download 脚本中添加导致偶然失败的逻辑。这可确保脚本能够处理失败的下载。

使用非阻塞下载确认

必须在非阻塞下载确认事务中维护有关下载内容的簿记信息。应在 publication_nonblocking_download_ack 或 nonblocking_download_ack 脚本中更新此信息，远程数据库成功应用下载后会调用该脚本。

如果出现故障或 SendDownloadAck 为 OFF，则不调用这些非阻塞下载确认脚本，也不更新下载时间戳。测试同步脚本时，应在 publication_nonblocking_download_ack 或 nonblocking_download_ack 脚本中添加导致偶然失败的逻辑。这可确保脚本能够处理失败的下载。

恢复失败的下载

下载故障是由下载过程中的通信错误或用户取消下载引起的。MobiLink 服务器保存客户端尚未接收的下载数据以用于可重新启动的下载。使用 -ds 选项，减小分配给可重新启动下载的最大数据量，可降低发生下载故障的可能性。服务器仅在发生以下情况之一时才释放下载数据：

- 用户成功完成下载。
- 用户通过新的同步请求进行恢复，但未启用下载恢复。
- 进来的请求需要使用高速缓存。此时将优先清除最早未成功的下载。

MobiLink 能够帮助恢复下载故障，并防止重新传输整个下载。此功能对 SQL Anywhere 和 UltraLite 远程数据库有不同的实现方法。请参见“-ds 选项”一节第 58 页。

SQL Anywhere 远程数据库

下载过程中同步失败时，下载的数据不应用到远程数据库，但成功传输的下载部分会存储在远程设备的临时文件中。Dbmlsync 使用此文件来避免进行长时间的数据重新传输，以及从下载故障中恢复过来。

有三种方法可以实现此功能。在任何情况下，如果存在要上载的新数据，dbmlsync 都会中止，恢复的下载也会失败。

- **-dc** 下载失败后，下次启动 dbmlsync 时可使用 -dc 恢复下载。如果失败下载的部分内容已被传输，则 MobiLink 服务器将仅传输该下载的剩余部分。

  有关详细信息，请参见“-dc 选项”一节 《MobiLink - 客户端管理》。

- **ContinueDownload (cd) 扩展选项** 在 dbmlsync 命令行上使用时，cd 扩展选项的作用与 -dc 选项相同。也可以将此选项存储在数据库中，或使用 sp_hook_dbmlsync_set_extended_options 在单一同步中设置此选项。

  请参见“ContinueDownload (cd) 扩展选项”一节 《MobiLink - 客户端管理》
  和“sp_hook_dbmlsync_set_extended_options”一节 《MobiLink - 客户端管理》。

- **sp_hook_dbmlsync_end 挂接** 可以使用 restart 参数来恢复下载。如果将 restartable download 参数设置为 true，则下载是可以恢复的。如果下载文件存在而且大小一定，则还可以在挂接中创建逻辑以恢复下载。

  请参见“sp_hook_dbmlsync_end”一节 《MobiLink - 客户端管理》。

UltraLite 远程数据库

可以如下所示控制 UltraLite 应用程序在下载失败后的行为：

- 如果在同步时将 Keep Partial Download 同步参数设置为 true，且下载在完成前失败，则 UltraLite 会应用已下载的部分更改。UltraLite 还会将 Partial Download Retained 同步参数设置为 true。

  此时，UltraLite 数据库可能会处于不一致状态。根据应用程序的不同，您可能希望确保同步成功完成或在允许对数据进行更改之前回退。请参见“Keep Partial Download 同步参数”一节 《UltraLite - 数据库管理和参考》和“Partial Download Retained 同步参数”一节 《UltraLite - 数据库管理和参考》。

- 要恢复下载，请将 Resume Partial Download 同步参数设置为 true 并再次进行同步。请参见“Resume Partial Download 同步参数”一节 《UltraLite - 数据库管理和参考》。

  重新启动的同步不执行上载，而只下载那些本应由失败的下载所下载的更改。也就是说，它将完成失败的下载，但不同步自上一次尝试以来所做的更改。要获得这些更改，您需要在失败的下载完成后再次进行同步，或调用 Rollback Partial Download 并进行同步（将 Resume Partial Download 设置为 false）。

  重新启动下载时，会自动再次使用失败同步的许多同步参数。例如，将忽略发布参数：同步会下载初始下载时要求的那些发布。唯一需要设置的参数是 Resume Partial Download 参数（必须设置为 true）和 User Name 参数。此外，如果设置的话，应遵循对以下参数的设置：

  - Keep Partial Download（在出现进一步中断的情况下）
  - DisableConcurrency
  - observer
  - User Data

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

- 要在不恢复同步的情况下回退失败的下载所做的更改，请调用回退更改的函数。对于嵌入式 SQL，该函数是 ULRollbackPartialDownload 函数。对于 UltraLite 组件，它是一种针对 Connection 对象的方法。
  - **UltraLite.NET** 请参见“RollbackPartialDownload 方法”一节 《UltraLite - .NET 编程》。
  - **嵌入式 SQL** 请参见“ULRollbackPartialDownload 函数”一节 《UltraLite - C 及 C++ 编程》。

  如果由于服务器或网络不可用等原因导致无法完成同步，而您想在允许最终用户继续工作的同时保持一致的数据集，那么您可能希望回退失败的下载所做的更改。

有关通信错误的详细信息，请参见错误消息。

注意
如果 send_download_ack 同步参数设置为 true，则恢复的下载将会忽略该设置。

下载确认

确保在远程数据库上接收到数据时不需要下载确认。

下载确认有两种模式：阻塞模式（现已不建议使用）和非阻塞模式。缺省设置为非阻塞。使用阻塞下载确认对性能有负面影响：建议使用非阻塞模式。

要使用下载确认，可以在客户端和服务器上进行相应设置。

在客户端上，通过 dbmlsync 扩展选项 SendDownloadACK 或 UltraLite 同步参数 Send Download Acknowledgement 指定下载确认。如果不在服务器上更改任何设置，则缺省为非阻塞下载确认。

在服务器上，可以设置 mlsrv11 -nba 选项指定阻塞下载确认。使用非阻塞下载确认时，有两个连接事件可以用来在统一数据库中记录上次成功下载的时间。

另请参见

- “publication_nonblocking_download_ack 连接事件”一节第 442 页
- “nonblocking_download_ack 连接事件”一节第 438 页
- “-nba 选项”一节第 70 页
- dbmlsync： “SendDownloadACK (sa) 扩展选项”一节 《MobiLink - 客户端管理》
- UltraLite： “发送下载确认同步参数”一节 《UltraLite - 数据库管理和参考》

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

从存储过程调用中下载结果集

可以从存储过程调用中下载结果集。例如，您目前可能有下表的 download_cursor：

```
CREATE TABLE MyTable (
    pk INTEGER PRIMARY KEY NOT NULL,
    col1 VARCHAR(100) NOT NULL,
    col2 VARCHAR(20) NOT NULL
)
```

download_cursor 表脚本可能如下所示：

```
SELECT pk, col1, col2
   FROM MyTable
      WHERE last_modified >= {ml s.last_table_download}
      AND employee = {ml s.username}
```

如果希望对 MyTable 的下载使用更高级的业务逻辑，可以按如下所示创建自己的脚本，其中 DownloadMyTable 是一个存储过程，它接受两个参数（上次下载时间戳和 MobiLink 用户名）并返回结果集。（此示例使用 ODBC 调用约定以便可移植）：

```
{call DownloadMyTable( {ml s.last_table_download}, {ml s.username} )}
```

下面是各个受支持的统一数据库的一些简单示例。有关全部详细信息，请查看统一数据库的文档。

以下示例适用于 SQL Anywhere、Adaptive Server Enterprise 和 Microsoft SQL Server。

```
CREATE PROCEDURE SPDownload
    @last_dl_ts DATETIME,
    @u_name VARCHAR( 128 )
AS
BEGIN
     SELECT pk, col1, col2
       FROM MyTable
          WHERE last_modified >= @last_dl_ts
          AND employee = @u_name
END
```

以下示例适用于 Oracle。Oracle 要求定义一个包。此包必须包含结果集的记录类型和一个返回此记录类型的游标类型。

```
Create or replace package SPInfo as
Type SPRec is record (
    pk     integer,
    col1   varchar(100),
    col2   varchar(20)
);
Type SPCursor is ref cursor return SPRec;
End SPInfo;
```

接下来，Oracle 需要一个以该游标类型作为第一个参数的存储过程。注意 download_cursor 脚本只传入两个参数，而不是三个。对于 Oracle 中返回结果集的存储过程，在存储过程定义中声明为参数的游标类型定义结果集的结构，但不这样定义真正的参数。本示例中，存储过程同样将该脚本添加到 MobiLink 系统表中。

```
Create or replace procedure
    DownloadMyTable( v_spcursor IN OUT SPInfo.SPCursor,
                     v_last_dl_ts IN DATE,
```

```
                    v_user_name IN VARCHAR ) As
Begin
    Open v_spcursor For
       select pk, col1, col2
         from MyTable
           where last_modified >= v_last_dl_ts
           and employee = v_user_name;
End;

CALL ml_add_table_script(
  'v1',
  'MyTable',
  'download_cursor',
  '{CALL DownloadMyTable(
     {ml s.last_table_download},{ml s.username} )}'
);
```

以下示例适用于 IBM DB2 LUW。

```
CREATE PROCEDURE DownloadMyTable(
   IN last_dl_ts TIMESTAMP,
   IN u_name VARCHAR( 128 ) )
        LANGUAGE SQL
        MODIFIES SQL DATA
        COMMIT ON RETURN NO
        DYNAMIC RESULT SETS 1

        BEGIN
            DECLARE C1, cursor WITH RETURN FOR
                SELECT pk, col1, col2 FROM MyTable
                WHERE last_modified >= last_dl_ts AND employee = u_name;
            OPEN C1;
        END;
```

以下示例适用于 IBM DB2 主机。

```
CREATE PROCEDURE DownloadMyTable(
   IN last_dl_ts TIMESTAMP,
   IN u_name VARCHAR( 128 ) )
        LANGUAGE SQL
        MODIFIES SQL DATA
        EXTERNAL NAME MYDMT
        WLM ENVIRONMENT MYWLM
        COMMIT ON RETURN NO
        DYNAMIC RESULT SETS 1
        BEGIN
            DECLARE C1, cursor WITH RETURN FOR
                SELECT pk, col1, col2 FROM MyTable
                WHERE last_modified >= last_dl_ts AND employee = u_name;
            OPEN C1;
        END;
```

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

从自引用表上载数据

某些表是自引用表。例如，雇员表可能包含一个列出雇员的列和一个列出每个雇员的经理的列，还可能包含用来管理经理的经理层次。这些表为同步操作提出了一个难题，因为 MobiLink 的缺省行为是合并远程数据库上的所有数据，此行为虽然很有效但会丢失事务顺序。

有两种技术可以处理这种情况：

- 如果您正使用 SQL Anywhere 远程数据库，则可以使用 dbmlsync -tu 选项来指定远程数据库中的每个事务都应作为独立事务发送。

  请参见“-tu 选项”一节 《MobiLink - 客户端管理》。

- 添加一个映射表，使事务与顺序无关。

Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

MobiLink 隔离级别

如果在 RDBM 上启用隔离级别，MobiLink 将在它所能够达到的最优隔离级别连接到统一数据库。选择缺省隔离级别可以提供最佳性能并确保数据的一致性。

通常，MobiLink 上载时使用隔离级别 SQL_TXN_READ_COMMITTED，如果可能，将在下载时使用快照隔离（如果不可能，则使用 SQL_TXN_READ_COMMITTED）。快照隔离可避免下载阻塞问题，直到事务在统一数据库上关闭。

快照隔离可导致下载数据重复（例如长时间运行的事务使同一快照被长期使用），但 MobiLink 客户端会自动处理这种情况，因此唯一的损失就是传输时间和远程的处理工作。

隔离级别 0 (READ UNCOMMITTED) 通常不适用于同步，它会导致数据不一致。

在连接到数据库后立即设置隔离级别。会出现一些其它连接设置，随后将提交事务。大多数 RDBMS 都需要 COMMIT，只有这样隔离级别（可能还有其它设置）才会生效。

SQL Anywhere 版本 10

SQL Anywhere 版本 10 支持快照隔离。缺省情况下，MobiLink 对上载使用 SQL_TXN_READ_COMMITTED 隔离级别，对下载使用快照隔离。

如果在 SQL Anywhere 统一数据库中启用快照隔离，MobiLink 将只能使用快照隔离。如果未启用快照隔离，MobiLink 将使用缺省的 SQL_TXN_READ_COMMITTED。

使数据库启用快照隔离可能会影响性能，因为无论使用快照隔离的事务有多少，都必须维护所有修改行的副本。请参见“启用快照隔离”一节 《SQL Anywhere 服务器 - SQL 的用法》。

使用 mlsrv11 -esu 选项可为上载启用快照隔离，使用 mlsrv11 -dsd 选项可禁用快照隔离。如果需要在连接脚本中更改 MobiLink 缺省隔离级别，应在 begin_upload 或 begin_download 脚本中执行此操作。如果在 begin_connection 脚本中更改缺省隔离级别，则您的设置可能会在上载和下载事务开始时被替换。

请参见“-esu 选项”一节第 62 页和“-dsd 选项”一节第 59 页。

SQL Anywhere 版本 10 之前的版本

如果您使用的是 SQL Anywhere 版本 10 之前的版本，则缺省的 MobiLink 隔离级别为 SQL_TXN_READ_COMMITTED。可在 begin_connection 脚本中更改整个 MobiLink 会话的缺省值，也可在 begin_upload 和 begin_download 脚本中为上载和下载分别更改。

Adaptive Server Enterprise

对于 Adaptive Server Enterprise，缺省的 MobiLink 隔离级别为 SQL_TXN_READ_COMMITTED。可在 begin_connection 脚本中更改整个 MobiLink 会话的缺省值，也可在 begin_upload 和 begin_download 脚本中为上载和下载分别更改。

Oracle

Oracle 支持快照隔离，但将其称为 READ COMMITTED。缺省情况下，MobiLink 在上载和下载时使用快照/READ COMMITTED 隔离级别。

可在 begin_connection 脚本中更改整个 MobiLink 会话的缺省值，也可在 begin_upload 和 begin_download 脚本中为上载和下载分别更改。

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

为使 MobiLink 服务器能够最有效地使用快照隔离，MobiLink 服务器使用的 Oracle 帐户必须拥有对 V_$TRANSACTION Oracle 系统视图的权限。如果没有此权限，将会发出警告而且下载时可能会丢失行。只有 SYS 可授予此访问权限。授予此访问权限的 Oracle 语法为：

```
grant select on SYS.V_$TRANSACTION to user-name
```

Microsoft SQL Server 2005 及更高版本

Microsoft SQL Server 2005 支持快照隔离。缺省情况下，MobiLink 对上载使用 SQL_TXN_READ_COMMITTED 隔离级别，对下载使用快照隔离。

如果在 SQL Server 统一数据库中启用快照隔离，MobiLink 将只能使用快照隔离。如果未启用隔离，MobiLink 将使用缺省的 SQL_TXN_READ_COMMITTED。有关详细信息，请参见 SQL Server 文档。

使用 mlsrv11 -esu 选项可为上载启用快照隔离，使用 mlsrv11 -dsd 选项可禁用快照隔离。如果需要在连接脚本中更改 MobiLink 缺省隔离级别，应在 begin_upload 或 begin_download 脚本中执行此操作。如果在 begin_connection 脚本中更改缺省隔离级别，则您的设置可能会在上载和下载事务开始时被替换。

请参见“-esu 选项”一节第 62 页和“-dsd 选项”一节第 59 页。

要在 SQL Server 上使用快照隔离，则用于连接 MobiLink 服务器与数据库的用户 ID 必须具有访问 SQL Server 系统表 SYS.DM_TRAN_ACTIVE_TRANSACTIONS 的权限。如果未授予此权限，MobiLink 将使用缺省的级别 SQL_TXN_READ_COMMITTED。

统一数据库在同时还运行其它数据库的 Microsoft SQL Server 上运行时，如果您将快照隔离用于上载或下载，并且上载或下载脚本不访问服务器上的任何其它数据库，这时应指定 MobiLink 服务器 -dt 选项。此选项使 MobiLink 忽略除当前数据库中的事务之外的所有事务。这可增加吞吐量并减少行的重复下载。

请参见“-dt 选项”一节第 60 页。

Microsoft SQL Server 版本 2005 之前的版本

如果您使用的是 Microsoft SQL Server 版本 2005 之前的版本，则缺省的 MobiLink 隔离级别为 SQL_TXN_READ_COMMITTED。可在 begin_connection 脚本中更改整个 MobiLink 会话的缺省值，也可在 begin_upload 和 begin_download 脚本中为上载和下载分别更改。

另请参见

- “-dsd 选项”一节第 59 页
- “-esu 选项”一节第 62 页
- “-dt 选项”一节第 60 页
- “同步过程”一节 《MobiLink - 入门》
- “隔离级别和一致性”一节 《SQL Anywhere 服务器 - SQL 的用法》
- “快照隔离”一节 《SQL Anywhere 服务器 - SQL 的用法》

MobiLink 性能

目录

性能提示 160
影响 MobiLink 性能的关键因素 163
监控 MobiLink 性能 166

性能提示

下面列出了一些建议，可以帮助您从 MobiLink 获得最佳性能。

测试

以下方面均会影响到同步系统的吞吐量：运行远程服务器的设备类型、远程数据库的模式、远程数据库的数据量和同步频率、网络特性（包括 HTTP、代理、Web 服务器和重定向器的特性）、运行 MobiLink 服务器的硬件、同步脚本、同步的并发量、使用的统一数据库的类型、运行统一数据库的硬件以及统一数据库的模式。

测试极其重要。在部署前，应使用您计划用于生产的硬件和网络来执行测试。还应该尝试以相同的远程数据库数量、相同的同步频率以及相同的数据量进行测试。此测试期间，您应该实验以下性能提示。

避免争用

避免同步脚本中的争用和最大化同步脚本中的并发。

例如，假定 begin_download 脚本在表中添加一列，用以计算下载总数。如果多个用户同时执行同步，则此脚本可以高效地使他们的下载序列化。在 begin_synchronization、end_synchronization 或 prepare_for_download 脚本中使用相同的计数器将更加有效，因为这些脚本将在提交以前被调用，所以任何数据库锁只被保存极短的时间。

请参见“争用”一节第 163 页。

有关同步的事务结构的信息，请参见“同步过程中的事务”一节 《MobiLink - 入门》。

使用数据库工作线程的最佳数目

使用 MobiLink -w 选项将 MobiLink 数据库工作线程数设置成提供最佳吞吐量的最小数目。您需要通过实验来确定适合您的情况的最佳数目。

较大数目的数据库工作线程可以通过允许同时有更多的同步来访问统一数据库，以提高吞吐量。

保持少量的数据库工作线程可以减小统一数据库中出现争用的可能性，减少到统一数据库的连接数，并减少优化高速缓存所需的内存。

请参见：

- “数据库工作线程数”一节第 164 页
- “-w 选项”一节第 100 页
- “-wu 选项”一节第 101 页

为 SQL Anywhere 客户端启用客户端下载缓冲区

对于 SQL Anywhere 客户端，下载缓冲区使得 MobiLink 数据库工作线程不必等待客户端应用下载就可以传送下载。缺省情况下启用下载缓冲区。但是，如果启用下载确认，则不能使用下载缓冲区（见下一段）。

请参见“DownloadBufferSize (dbs) 扩展选项”一节 《MobiLink - 客户端管理》。

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

使用非阻塞下载确认

缺省情况下，不启用下载确认。但是，下载确认对于在统一数据库中记录远程进度很有用。当远程应用下载时，阻塞下载确认会阻碍数据库连接。如果使用下载确认，您应使用非阻塞下载确认。非阻塞下载确认为缺省设置。

请参见“SendDownloadACK (sa) 扩展选项”一节 《MobiLink - 客户端管理》和“-nba 选项”一节 第 70 页。

避免同步不必要的 BLOB

在频繁同步的行中包括 BLOB 会降低效率。要避免此类情况，可以创建一个包含 BLOB 和 BLOB ID 的表格，然后引用表中需要同步的 ID。

设置数据库连接的最大数目

将 MobiLink 数据库连接的最大数目设置为同步脚本版本的数目与 MobiLink 数据库工作线程数的乘积加 1。这将减少需要 MobiLink 关闭和创建数据库连接的次数。使用 mlsrv11 -cn 选项设置连接的最大数目。

请参见“MobiLink 数据库连接”一节第 165 页和“-cn 选项”一节第 52 页。

具有足够的物理内存

确保运行 MobiLink 服务器的计算机具有足够的物理内存以满足高速缓存以及自身的其它内存需求。

被积极处理的同步数量不受数据库工作线程数的限制。MobiLink 服务器可以同时为较大数量的同步解开上载并发送下载。为得到最佳性能，重要一点是 MobiLink 服务器具有足够大的内存高速缓存来处理这些同步，而不分页到磁盘。-cm 选项用于设置 MobiLink 服务器的内存高速缓存。

请参见“-cm 选项”一节第 51 页。

使用足够的处理能力

应该赋予 MobiLink 足够强的处理能力，以使 MobiLink 服务器处理不会成为瓶颈。通常，MobiLink 服务器所需的 CPU 远小于统一数据库所需的 CPU。但使用 Java 或 .NET 行处理增加了 MobiLink 服务器处理需求。实际上，网络限制或数据库争用更可能成为瓶颈。

使用详细程度最小的记录模式

使用与您的业务需求一致的详细程度最小的记录模式。缺省情况下，详细记录模式处于关闭状态而且 MobiLink 不将日志写入磁盘。您可以使用 -v 选项控制记录的详细程度，并使用 -o 或 -ot 选项将记录写入文件。

作为对详细日志文件的替代方法，您可以使用 MobiLink 监控器来监控同步。MobiLink 监控器不一定要和 MobiLink 服务器位于同一台计算机上，且监控器连接对 MobiLink 服务器的性能影响甚微。请参见“MobiLink 监控器”第 167 页。

操作系统限制计划

操作系统会限制服务器通过 TCP/IP 支持的并发连接的数量。如果达到此限制（超过 1000 个客户端同时尝试进行同步时，可能发生此种情况），操作系统可能会出现意外行为，如意外关闭连接及拒绝尝试连接的其它客户端。为防止此行为的发生，可使用 -sm 选项来指定低于操作系统限制的最大远程连接数。

当客户端尝试与 MobiLink 服务器同步时，如果服务器已接受了由 -sm 选项指定最大数的并发同步，则此客户端会收到错误代码 -85 (SQLE_COMMUNICATIONS_ERROR)。客户端应用程序应该处理该错误并在几分钟后再次尝试连接。

有关 -sm 选项的详细信息，请参见“-sm 选项”一节第 89 页。

有关 SQLE_COMMUNICATIONS_ERROR 的详细信息，请参见“通信错误”一节 《错误消息》。

Java 或 .NET 与 SQL 同步逻辑

目前尚未发现使用 Java 或 .NET 同步逻辑与 SQL 同步逻辑相比存在明显的吞吐量的差别。但 Java 和 .NET 同步逻辑对每个同步都会产生一些额外开销，需要占用更多内存。

另外，SQL 同步逻辑在运行统一数据库的计算机上执行，而 Java 或 .NET 同步逻辑在运行 MobiLink 服务器的计算机上执行。因此，如果统一数据库的负载很重，则可能非常适合使用 Java 或 .NET 同步逻辑。

使用直接行处理的同步增加了 MobiLink 服务器的处理负担，因此您可能需要更多的 RAM，也可能需要更多的磁盘空间和更强的 CPU 处理能力，这取决于实现直接行处理的方式。

优先级同步

如果您所具有的某些表需要比其它表更频繁地进行同步，则为它们单独创建发布和预订。在 Sybase Central 中使用同步模型时，可通过创建多个模型来完成上述操作。您可以比其它发布更频繁地同步此优先级发布，并且在非高峰时间同步其它发布。

只下载所需行

注意仅下载需要的行，例如通过使用时间戳同步而不是快照同步。下载不需要的行是一种浪费，并会导致同步性能降低。

优化脚本执行

统一数据库中脚本的性能是一个很重要的因素。在表中创建索引可以使上载游标脚本和下载游标脚本更有效地定位需要的行。但是，索引过多可能会减慢上载速度。

当使用 Sybase Central 中的 [**创建同步模型向导**] 来创建您的 MobiLink 应用程序时，在部署模型时会自动为每个下载游标定义一个索引。

对于较大的上载，估计行数

对于 SQL Anywhere 客户端，通过向 dbmlsync 提供估计的上载行数，可以显著提高上载大量行的速度。使用 dbmlsync -urc 选项执行此操作。

请参见“-urc 选项”一节 《MobiLink - 客户端管理》。

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

影响 MobiLink 性能的关键因素

任何系统的整体性能，包括 MobiLink 同步的吞吐量，通常都会受限于系统中某一点的瓶颈。对于 MobiLink 同步而言，下列所述或许就是限制同步吞吐量的瓶颈：

- **统一数据库的性能 MobiLink 脚本的执行速度** 统一数据库可以执行 MobiLink 脚本的速度对于 MobiLink 具有特殊的重要性。多个数据库工作线程可以同时执行多个脚本，因此，为获得最佳吞吐量，在同步脚本中要避免数据库的争用。
- **MobiLink 数据库工作线程数** 线程的数目越少，涉及的数据库连接就越少，在统一数据库中发生争用的机会就越小，操作系统的开销也越小。然而，线程数目过少可能会导致客户端等待空闲数据库工作线程，而与统一数据库连接过少将不足以实现高效重叠。
- **客户端与 MobiLink 通信的带宽** 对于慢速连接（如通过拨号或广域无线网实现的连接）而言，网络可能会导致客户端和 MobiLink 服务器对数据传输的等待。
- **客户端处理速度** 因为下载时由于写入行和索引而需要更多的客户端处理量，所以客户端处理速度过慢在下载过程中比上载过程中更容易成为瓶颈。
- **MobiLink 与统一数据库进行通信的带宽** 如果 MobiLink 和统一数据库在同一计算机上运行或两者位于由高速网络连接的不同计算机上，带宽可能不会成为瓶颈。
- **运行 MobiLink 服务器的计算机的速度** 如果运行 MobiLink 的计算机的处理速度缓慢，或没有足够的内存可供 MobiLink 数据库工作线程和缓冲区使用，则 MobiLink 的执行速度就可能成为同步的瓶颈。如果缓冲区和数据库工作线程获得适当的物理内存，则磁盘速度对 MobiLink 服务器的性能几乎没有什么影响。

调优 MobiLink 性能

实现 MobiLink 最优同步吞吐量的关键是同时并高效地执行多个同步过程。要同时启用多个同步过程，MobiLink 将不同的数据库工作线程池用于不同的任务。一个池专用于从网络读取上载数据并将其解开。另一个称为**数据库工作线程**的线程池将该上载应用于统一数据库，并从统一数据库获取要下载的数据。另一个数据库工作线程池专用于将下载数据压缩并发送到远程数据库。每个数据库工作线程通过同步脚本使用到统一数据库的单个连接以便应用和读取更改。

争用

最重要因素是避免同步脚本中的数据库争用。同其他多个客户端使用数据库的情形一样，当客户端同时访问数据库时，您希望将数据库争用降到最低。每个同步都必须修改的数据库行可能导致争用的增加。例如，如果您的脚本在行中包含自动增加的计数器，则对该计数器的更新操作就有可能成为瓶颈。

同步请求被接受（上限为 -sm 选项指定的值），上载的数据被读取并解开，以便针对数据库工作线程做好准备。如果同步请求多于数据库工作线程，多出的请求将排队等待空闲数据库工作线程。

您可以控制数据库工作线程和连接的数目，但 MobiLink 总会确保每个数据库工作线程至少拥有一个连接。如果连接数多于数据库工作线程数，多出的连接将空闲下来。多出的连接可能会对多个脚本版本有用，这一点将在下面进行讨论。

数据库工作线程数

除了同步脚本中的争用以外，影响同步吞吐量的最重要的因素就是数据库工作线程的数目。数据库工作线程的数目将控制统一数据库中可以同时进行的同步的数目。

测试对于确定数据库工作线程的最佳数目十分重要。

数据库工作线程数的增加将允许更多重叠的同步来访问统一数据库，并增加了吞吐量。但也会增加重叠同步之间的资源和数据库争用，因此可能会增加单个同步花费的时间。随数据库工作线程数目的增加，由单个同步更加耗时所产生的开销将逐渐超出由共存同步数目的增加所产生的收益，此时添加更多的数据库工作线程反而降低了吞吐量。可以通过实验确定您所处环境中数据库工作线程的最佳数目，但以下内容可能对您有所帮助。

对于上载，据性能测试显示，当数据库工作线程数目相对较少（绝大多数情况下，三至十个工作线程）时，可以得到最佳的吞吐量。浮动量取决于下列因素：统一数据库的类型、数据量、数据库模式、同步脚本的复杂性和使用的硬件。导致出现瓶颈的通常原因是：在统一数据库中同时执行上载脚本的 SQL 的各数据库工作线程之间的争用。

对于下载，当使用下载确认时，数据库工作线程的最佳数目取决于客户端到 MobiLink 连接的带宽和客户端的处理速度。对速度较慢的客户端而言，获取最佳的下载性能需要较多的数据库工作线程。这是因为下载比上载涉及更多的客户端处理和更少的统一数据库处理。当使用阻塞下载确认时，在远程数据库完成对下载的应用之前，数据库工作线程会一直被阻塞。对于非阻塞下载确认，不需要更多的工作线程。

当不使用下载确认时（缺省设置），客户端到 MobiLink 连接的带宽的影响将减弱，因为数据库工作线程可以在其它线程发送下载时自由处理其它同步。因此，数据库工作线程的数目不再那么重要。

可以并行发送许多下载—远多于数据库工作线程的数目。要获得最佳的下载性能，重要的一点是，MobiLink 服务器具有足够的 RAM 来缓冲这些下载。否则，下载将被分页到磁盘，因此可能会降低下载性能。要指定 MobiLink 服务器的内存高速缓存大小，请使用 -cm 选项。

请参见“-cm 选项”一节第 51 页。

如果 MobiLink 服务器开始分页到磁盘（可能因为并行处理的下载过多），考虑使用 -sm 选项来减少数据库工作线程的数目或限制被主动处理的同步总数。

请参见“-sm 选项”一节第 89 页。

关闭下载确认（缺省设置）可以减少用于下载的数据库工作线程的最佳数目，因为数据库工作线程不必等待客户端来应用下载。

请参见“SendDownloadACK (sa) 扩展选项”一节 《MobiLink - 客户端管理》。

如果要使用下载确认，则使用非阻塞下载确认可获得更好的性能（相对于阻塞下载确认）。在非阻塞确认模式下，当远程数据库应用下载时，服务器会重用数据库工作线程。这意味着可以不必增加数据库工作线程的数目，即可获得更佳的性能。

MobiLink 为获得最佳下载吞吐量和最佳上载吞吐量提供了两个选项。您可以指定数据库工作线程的总数以便优化下载吞吐量。您也可以对同时执行上载的工作线程数加以限制以便优化上载吞吐量。

-w 选项控制数据库工作线程的总数。缺省值为 5。

-wu 选项限制可以将上载同时应用于统一数据库的数据库工作线程的数目。缺省情况下，所有数据库工作线程都可以同时执行上载，但这样做可能导致在统一数据库中出现严重的争用现象。使用 -

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

wu 选项，可以在仍使较大数目的数据库工作线程优化下载数据获取的同时，减少争用情况的发生。-wu 选项仅在其指定的数目少于数据库工作线程的总数时才起作用。

请参见“-w 选项”一节第 100 页和“-wu 选项”一节第 101 页。

MobiLink 数据库连接

MobiLink 将为每个数据库工作线程创建一个数据库连接。您可以使用 -cn 选项指定 MobiLink 创建一个较大的数据库连接池，但是，如果 MobiLink 并不需要关闭连接或使用不同的脚本版本，多余的连接就会空闲下来。

在两种情况下 MobiLink 会关闭一个数据库连接并打开一个新的连接。第一种情况是发生错误时。第二种情况是，如果客户端请求同步脚本版本，但所有可用的连接都没有使用该同步版本。

注意
每个数据库连接都与一个脚本版本相关联。更改版本时，必须关闭连接然后再重新打开。

如果您通常使用多个脚本版本，则可以通过增加连接数以减少需要 MobiLink 关闭和打开连接的次数。如果用于同步的连接数为数据库工作线程数与脚本版本数的乘积，则您完全不需要关闭和打开连接。

以下命令行给出了对具有两个脚本版本的 MobiLink 进行调优的示例：

```
mlsrv11 -c "dsn=SQL Anywhere 11 Demo" -w 5 -cn 10
```

因为用于同步的数据库连接的最大数为脚本版本数与数据库工作线程数的乘积，因此将 -cn 设置为 10 可确保数据库连接不被过多地关闭和打开。

请参见“-cn 选项”一节第 52 页。

Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

监控 MobiLink 性能

您可以借助多种工具来监控同步的性能。

MobiLink 监控器是用于监控同步的图形工具。通过该工具，您可以了解同步的各个方面所占用的时间。

请参见“MobiLink 监控器”第 167 页。

此外，还提供多种 MobiLink 脚本来监控同步。借助这些脚本，您可以在业务逻辑中使用性能统计。例如，您可能想要存储性能信息以供将来分析，或者在某一同步占用太长时间的情况下向 DBA 发出警报。有关详细信息，请参见：

- “download_statistics 连接事件”一节第 376 页
- “download_statistics 表事件”一节第 379 页
- “synchronization_statistics 连接事件”一节第 453 页
- “synchronization_statistics 表事件”一节第 456 页
- “time_statistics 连接事件”一节第 459 页
- “time_statistics 表事件”一节第 462 页
- “upload_statistics 连接事件”一节第 479 页
- “upload_statistics 表事件”一节第 483 页

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

MobiLink 监控器

目录

MobiLink 监控器简介 168
启动 MobiLink 监控器 169
使用 MobiLink 监控器 171
保存 MobiLink 监控器数据 179
自定义您的统计信息 180
MobiLink 统计信息属性 181

MobiLink 监控器简介

MobiLink 监控器是一种 MobiLink 管理工具，可为您提供与同步性能有关的详细信息。

启动 MobiLink 监控器并将其连接到 MobiLink 服务器后，MobiLink 监控器就开始收集与该监控会话中发生的所有同步有关的统计信息。在断开 MobiLink 监控器的连接或关闭 MobiLink 服务器之前，MobiLink 监控器将继续收集数据。

可以在 MobiLink 监控器界面中以表格或以图形形式查看数据。您还可以采用二进制格式保存数据，以便以后使用 MobiLink 监控器进行查看；或者采用 *.csv* 格式保存数据，以便用其它工具（例如 Microsoft Excel）打开；您也可以将数据导出到一个 ODBC 数据源（例如支持 MobiLink 的关系数据库）中。

MobiLink 监控器输出使您可以查看各种关于同步的信息。例如，您可以迅速识别那些导致错误或符合您指定的条件的同步。通过检查不同期间的同步是否有同时结束的阶段，您可以识别同步脚本中可能的争用（因为同步在等待前一个阶段的完成，然后才能继续）。

因为监控不会降低性能（特别当 MobiLink 监控器与 MobiLink 服务器在不同计算机上运行时），所以可以在开发与生产中例行使用 MobiLink 监控器。

SQL Anywhere 监控器

SQL Anywhere 监控器是一种基于浏览器的管理工具，可向您提供与 SQL Anywhere 数据库和 MobiLink 服务器的健康情况和可用性有关的信息。对于访问整个系统的健康情况和可用性信息以及分析整个同步统计信息，SQL Anywhere 监控器很有用。SQL Anywhere 监控器不提供关于单个同步的信息。要获得关于单个同步的详细信息（包括计时和每个同步的其它统计信息），请使用 MobiLink 监控器。

有关 SQL Anywhere 监控器的详细信息，请参见“SQL Anywhere 监控器” 《SQL Anywhere 服务器 - 数据库管理》。

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

启动 MobiLink 监控器

您可为每一 MobiLink 服务器运行多个 MobiLink 监控器实例。

♦ 开始监控数据

1. 如果您的统一数据库和 MobiLink 服务器尚未运行，则启动它们。
2. 从 [开始] 菜单中选择 [**程序**] » [**SQL Anywhere 11**] » [**MobiLink 监控器**]。

   或者，您也可以在命令提示符下键入 **mlmon**。有关详细信息，请参见以下内容。
3. MobiLink 监控器连接的启动类似于与 MobiLink 服务器的同步连接。例如，如果您用 -zu+ 启动了 MobiLink 服务器，则您在此处使用了哪一用户 ID 无关紧要。对于所有 MobiLink 监控器会话，该脚本版本设置为 for_ML_Monitor_only。

   应按如下所示完成 [**连接到 MobiLink 服务器**] 窗口：

   - **主机** 运行 MobiLink 服务器的计算机的网络名称或 IP 地址。缺省情况下，它是运行 MobiLink 监控器的计算机。如果 MobiLink 服务器与 MobiLink 监控器运行在同一台计算机上，则可以使用 [**localhost**]。
   - **协议** 应被设置为与 MobiLink 服务器正用于同步请求的相同网络协议和端口。
   - **端口** 应被设置为与 MobiLink 服务器正用于同步请求的相同网络端口。
   - **加密** 如果您选择 HTTPS 或 TLS 作为协议，则此框将被启用。可以从下拉列表中选择一个加密类型。
   - **其它协议选项** 指定可选参数。您可以设置以下参数，如果需要指定多个参数，可以用分号分隔：
     - **buffer_size=***数字*
     - **client_port=***nnnn*
     - **client_port=***nnnn-mmmmm*
     - **persistent=**{**0**|**1**}（仅适用于 HTTP 和 HTTPS）
     - **proxy_host=***proxy_hostname*（仅适用于 HTTP 和 HTTPS）
     - **proxy_port=***proxy_portnumber*（仅适用于 HTTP 和 HTTPS）
     - **url_suffix=***suffix*（仅适用于 HTTP 和 HTTPS）
     - **version=***HTTP-version-number*（仅适用于 HTTP 和 HTTPS）

   请参见“MobiLink 客户端网络协议选项” 《MobiLink - 客户端管理》。
4. 启动同步。

   在收集了数据后数据将出现在 MobiLink 监控器中。

在命令行中启动 mlmon

命令行选项使您能够让 MobiLink 监控器在启动时打开文件或连接到 MobiLink 服务器。使用以下语法：

mlmon [*connect-options* | *inputfile*.{ **mlm** | **csv** }]

其中：

connect-options 可以是以下一个或多个值：

- **-u** *ml_username*（需要连接到 MobiLink 服务器）
- **-p** *password*
- **-x** { **tcpip** | **tls** | **http** | **https** } [**(** *keyword=value*;... **)**]

  keyword=value 对可以是上面所述的主机、协议和其它网络参数。必须指定 -x 选项才能连接到 MobiLink 服务器。
- **-o** *outputfile*.{ **mlm** | **csv** }

  -o 选项在连接结束时关闭 MobiLink 监控器，同时将会话保存在指定文件中。

您可以键入 **mlmon -?** 查看 mlmon 的语法。

启动 Unix 上的 MobiLink 监控器

如果使用的是支持 Linux 桌面图标的 Linux 版本并且选择了在安装 SQL Anywhere 11 时安装它们，则可使用以下步骤。

♦ 启动 MobiLink 监控器（Linux Desktop 图标）

1. 从 [**应用程序**] 菜单中选择 [**SQL Anywhere 11**] » [**MobiLink 监控器**]
2. 输入在此步骤中介绍的连接 MobiLink 服务器的信息以开始监控数据。

注意
以下步骤假定您已发起了 SQL Anywhere 实用程序。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》 。

♦ 启动 MobiLink 监控器（Unix 命令行）

1. 在终端会话中输入以下命令：

   ```
   mlmon
   ```
2. 如上所述，输入相应信息以便连接到 MobiLink 服务器。

停止 MobiLink 监控器

♦ 停止 MobiLink 监控器

1. 在 MobiLink 监控器中，选择 [**监控器**] » [**断开与 MobiLink 服务器的连接**]。这将停止数据收集。

   也可以通过关闭 MobiLink 服务器或 MobiLink 监控器来停止收集数据。

   在关闭 MobiLink 监控器前，您可以保存此会话的数据。请参见“保存 MobiLink 监控器数据”一节第 179 页。
2. 当您准备好要关闭 MobiLink 监控器时，可选择 [**文件**] » [**关闭**]。

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

使用 MobiLink 监控器

MobiLink 监控器具有以下几个窗格：

- **详细信息表** [**详细信息表**] 是顶部窗格。它是一个电子表格，显示每个同步所用的总时间，并且在它的细分项中显示该同步的各部分所用的时间。

  请参见“[详细信息表] 窗格”一节第 171 页。

- **运用图形** [**运用图形**] 是第二个窗格。它提供对 MobiLink 服务器上不同队列的队列长度的图形表示。[**运用图形**] 窗格和 [**图表**] 窗格使用同一比例尺。[**图表**] 窗格底部的比例尺表示时间。您可以通过在下面的 [**一览表**] 窗格中拖动并选择数据，或通过选择 [**视图**] » [**转到**]，来选择 [运用图形] 中显示的数据。

  请参见“[运用图形] 窗格”一节第 172 页。

- **图表** [**图表**] 是第三个窗格。它提供同步的图形表示形式。此窗格底部的标尺表示时间。您可以通过在下面的 [**一览表**] 窗格中拖动并选择数据，或通过选择 [**视图**] » [**转到**]，来选择 [图表] 中显示的数据。

  请参见“[图表] 窗格”一节第 174 页。

- **一览表** [**一览表**] 是底部窗格。它显示会话内所有同步的总览。此窗格中有一个称为 [**选取框**] 工具的框轮廓，您可以使用它来选择 [**图表**] 窗格和 [**运用图形**] 窗格中显示的数据。

  请参见“[一览表] 窗格”一节第 175 页。

此外，还有一个可用于对显示进行自定义的 [**选项**] 窗口以及用于查看更多详细信息的各种属性窗口。请参见：

- “[选项] 窗口”一节第 176 页
- “会话属性”一节第 176 页
- “示例属性”一节第 177 页
- “同步属性”一节第 177 页

[详细信息表] 窗格

[**详细信息表**] 提供与每一同步部分占用多长时间有关的信息。所有时间均由 MobiLink 服务器测量。即使您没有定义相应的脚本，某些时间也可能是非零的。

可以通过打开 [**工具**] » [**选项**]，然后打开 [**表**] 选项卡，来选择 [**详细信息表**] 窗格中显示的列。有关可用统计信息的说明，请参见“MobiLink 统计信息属性”一节第 181 页。

缺省情况下显示以下列：

- **sync** 标识每个同步。此 ID 由 MobiLink 服务器而非由 MobiLink 监控器指派，因此在任何一个指定 MobiLink 监控器会话中均不必从 1 开始，并且不按数字顺序接收。您可以在 [**同步属性**] 窗口中看到相同的 ID。请参见“同步属性”一节第 177 页。
- **remote_id** 远程数据库的 ID。

- **user** 同步用户。
- **version** 同步脚本的版本。

  请参见“脚本版本”一节第 303 页。
- **download_ack** 下载肯定回答的类型，可以是无、阻塞或非阻塞。
- **start_time** MobiLink 服务器启动同步的日期和时间。（该时间可能比客户端请求同步的时间要晚。）
- **duration** 以秒为单位的同步的总持续时间。
- **sync_request** MobiLink 接收从客户端上载的数据所用的时间，以秒为单位。这是指同步过程中从与远程端建立连接开始并恰好到验证前结束的这段时间。
- **receive_upload** 从建立远程数据库与 MobiLink 服务器之间的网络连接直至接收到上载流的第一批字节为止所花费的时间。如果不将 -sm 设为一个比 -nc 更小的值，那么此时间是没有意义的，此时该时间可包含同步数大于由 -sm 指定的最大活动同步数时同步暂停的时间。
- **get_db_worker** 获得空闲数据库工作线程所需的时间。
- **connect** 在需要新数据库连接时数据库工作线程建立数据库连接所需的时间。例如，出现错误或脚本版本变更的情况下会需要建立新连接。
- **authenticate_user** MobiLink 校验同步请求、校验用户名和校验口令（如果您的同步设置要求身份验证）所用的时间，以秒为单位。这是指验证用户事务所用的时间（从验证时开始并恰好到 begin_synchronization 事件之前结束）。
- **begin_sync** 运行您的 begin_synchronization 脚本（如果该脚本已运行）所用的时间，以秒为单位。
- **apply_upload** 将上载应用于统一数据库所用的时间，以秒为单位。该时间是 begin_upload 脚本和 end_upload 脚本之间的时间。
- **prepare_for_download** 运行您的 prepare_for_download 脚本（如果该脚本已运行）所用的时间，以秒为单位。
- **fetch_download** 下载数据所用的时间，以秒为单位。该时间是 begin_download 脚本和 end_download 脚本之间的时间。如果启用下载确认，则该时间还包括对远程数据库应用下载并返回确认所用的时间。
- **end_sync** 运行 end_synchronization 脚本（如果该脚本已运行）所用的时间，以秒为单位。

若要按特定列对表排序，则单击列标题。如果 MobiLink 监控器中出现新数据，则新数据会在添加时就进行排序。

您可以通过清除 [**视图**] 菜单中的 [**详细信息表**] 选项来关闭 [**详细信息表**] 窗格。

[运用图形] 窗格

[**运用图形**] 是从上算起的第二个窗格。它显示几种类型的工作队列的服务器统计信息，。

有关此窗格中可用数据的详细信息，请参见“使用 [运用图形]”一节第 173 页。

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

[**运用图形**] 使用与 [**图表**] 相同的水平滚动条、水平时间标签以及缩放级别。这表明在时间段的某一瞬间在 [**图形**] 窗格和 [**图表**] 窗格之间垂直对正。

有多种方法可以选择图形中显示的数据：

- 从 [**视图**] 菜单中选择 [**转到**]。
- 在 [**一览表**] 窗格中移动 [**选取框**] 工具。[**选取框**] 工具为 [**一览表**] 窗格中出现的小框。请参见“[一览表] 窗格”一节第 176 页。

可以双击 [**运用图形**] 的某个区域调出用于显示其所代表的示例间隔详细情况的 [**示例属性**] 窗口。示例时间间隔长约一秒。请参见“示例属性”一节第 177 页。

注意
属性列表从 MobiLink 服务器或打开的 *.mlm* 文件中获得，并使用 MobiLink 服务器的语言。如果 MobiLink 监控器使用其它语言，则某些字符可能不会正确显示。

使用 [运用图形]

要查看 [**运用图形**] 的值以及自定义输出，请选择 [**工具**] » [**选项**]，然后打开 [**图形**] 选项卡。此选项卡用颜色来标识 [**运用图形**] 队列，并允许您自定义图形。

属性

- **TCP/IP 工作队列** 此队列表示在 MobiLink 服务器中由低层级网络层完成的工作。该层负责在网络中读写包。队列中充满读写请求。当通知队列从网络读取进入的数据，和/或流工作线程通知队列对网络进行写操作时，该队列将增大。

  如果该队列被进行备份，则通常是由于读或写的积压请求所致—有时是读写积压请求都存在，但通常是由于两者之一。在服务器要使用大量 RAM 且内存页要进行大量出入交换时，读取请求可能会得到备份。请考虑增加更多 RAM。客户端和服务器之间的网络连接很慢时，写入请求可能会得到备份。如果该队列是唯一要备份的队列，则请查看 CPU 使用情况。如果 CPU 使用率高，则可能有读取/内存问题。如果 CPU 使用率低，则会导致写入较慢。请考虑使用更快的网络。

- **流工作队列** 仅适用于版本 10 客户端。此队列表示在 MobiLink 服务器中由高级网络层完成的工作。该层负责高层级网络协议工作，例如，HTTP、加密和压缩。该队列在大量读取请求从 TCP/IP 层进入，以及/或大量写入请求从命令处理器层进入时将增大。如果该队列是唯一要备份的队列，请考虑删除某些网络协议，例如，HTTP 或压缩。如果无法进行此操作，则请考虑使用 -sm 选项减少允许的并发同步的数量。

  请参见“-sm 选项”一节第 89 页。

- **活动工作队列** 此队列表示 MobiLink 服务器中负责在服务器内发送脉冲事件的层。例如，该层负责对连接的 MobiLink 监控器触发示例的每秒一个脉冲。

  备份该队列是非常不可能的，因此缺省情况下，它是不可见的。

- **命令处理器工作队列** 该队列表示由 MobiLink 服务器完成的工作，用以解释内部 MobiLink 协议命令并将这些命令应用于统一数据库。该队列在出现大量请求时将增大。请求类型包括同步

请求、监听器请求、mlfiletransfer 请求等。该队列在统一数据库忙于同步，然而仍有更多的同步请求持续进入时也会增大。

如果该队列是唯一要备份的队列，则请查看 CPU 使用情况。如果 CPU 使用率高，请求量则可能过高。应考虑使用 -sm 选项减少允许的并发同步的数量。如果 CPU 使用率低，则应注意提高统一数据库的性能。

请参见“-sm 选项”一节第 89 页。

- **数据库工作线程忙碌** 此值表明 MobiLink 服务器施压统一数据库的程度。此值的每个部分代表一个在数据库内正在忙碌的数据库工作线程。插入、更新、删除或选择之间没有区别。当此值为零时，服务器并未运行此统一数据库。

  当此数值很高时（接近于 mlsrv10 -w 选项的最大设置），MobiLink 服务器施压统一数据库的程度已达极限。在这种情况下，如果吞吐量令人满意，则无需任何操作。如果对吞吐量不满意，请考虑通过 -w 选项来增加数据库工作线程数量。注意，-w 值越高连接间的争用越激烈。这在当所有连接都在进行上传时尤为不利，因此，您可能需要使用 mlsrv11 -wu 选项设置执行上传的数据库工作线程下限。如果看起来难以确定可提供足够吞吐量的 -w 和 -wu 设置，则请检查您的同步脚本内是否存在可能的争用问题。最后，您可以查阅 RDBMS 文档找到改善统一数据库整体性能的方法。

范围

此栏说明每个属性的当前范围。

[**运用图形**] 上的垂直比例的范围始终为 0 到 100。这表示比例的百分比范围为 0 到 100。每个值都有对应的比例。缺省情况下，所有比例为 5，这表示如果值的范围为 0 到 20 之间，则按比例（乘 5）计算的范围为 0 到 100。如果某个值大于 20，则将自动调整比例以使最大值为 100。

要确定显示中的最大值，请用 100 除以比例。例如，如果 TCP/IP 工作队列的比例为 2.381，则最大值便为 (100/2.381) = 42。实际最大值通常并不重要。重要的是，靠近图形顶部的值正接近给定属性的当前已知最大值—即，在当前的监控会话中观察到的属性的峰值负载。

当图形一直接近显示画面的顶部并注意到同步吞吐量下降时，则可能出现性能问题，需要进行检查。同样，如果一个或多个值随时间向上蠕变而未逐渐减小，则可能存在性能问题。请注意，图形在 MobiLink 服务器执行良好时，可能常常会延伸到显示画面的顶部。这正表明 MobiLink 服务器很忙，并在顺利执行其作业。

抗锯齿功能

自定义的其中一个选项便是抗锯齿功能。抗锯齿功能会得到效果更好的图形外观，但会降低绘图的速度。

[图表] 窗格

[**图表**] 窗格显示与 [**详细信息表**] 相同的信息，只是它以图形格式显示。[**图表**] 中的小条表示每一次同步所用的时间长度，而小条的细部表示同步的各阶段。

查看数据

在 [**详细信息表**] 中单击某一同步可以选择该同步。

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

双击某个同步可以打开 [**同步属性**] 窗口。请参见“同步属性”一节第 177 页。

按远程 ID 或精简模式对数据进行分组

您可以按用户分组数据。选择 [**视图**] » [**通过远程 ID**]。

或者您可以精简模式查看数据，用尽可能少的行数显示所有活动的同步。选择 [**视图**] » [**精简视图**]。在 [**精简视图**] 中，行号没有什么意义。

放大数据

有几种方法可以选择 [**图表**] 窗格中可见的数据：

- **缩放选项** [**视图**] 菜单中的缩放选项和工具栏上的缩放按钮使您可以进行放大和缩小。要使同步填满可用区域，请使用 [**缩放到所选值**]。
- **滚动条** 单击 [**图表**] 窗格底部的滚动条并滑动它。
- **[转到] 窗口** 要打开这个窗口，请单击 [**视图**] » [**转到**]。

  在 [**开始日期和时间**] 中，您可以指定在 [**图表**] 窗格中出现的数据的开始时间。如果您更改此设置，则必须至少指定日期-时间的年、月和日。

  在 [**图表范围**] 中，可以指定显示时间所持续的时间。图表范围可以按毫秒、秒、分钟、小时或天指定。图表范围决定了数据的详细程度：时间长度越短，可以看到的细节越多。
- **[选取框] 工具** 在 [**一览表**] 窗格中移动 [**选取框**] 工具。[**选取框**] 工具为 [**一览表**] 窗格中出现的小框。请参见“[一览表] 窗格”一节第 176 页。

时间轴

在 [**图表**] 窗格的底部，有一个显示时间段的标尺。时间的格式根据显示的时间范围自动重新调整。通过将光标悬停在该标尺上，始终可以看到完整的日期-时间。

缺省的颜色方案

您可以通过打开 [**选项**] 窗口（从 [**工具**] 菜单可以获得该窗口），查看或设置 [**图表**] 窗格中的颜色。[**图表**] 窗格缺省的颜色方案使用石灰绿色表示上载，使用珊瑚红色表示下载，使用蓝色表示开始和结束阶段，使用较深的阴影表示阶段的早期部分。

有关设置颜色的信息，请参见“[选项] 窗口”一节第 176 页。

[一览表] 窗格

[**一览表**] 窗格显示整个 MobiLink 监控器会话的概要情况。可使用 [**选取框**] 工具（[**一览表**] 窗格内的小框）浏览会话。

用多种颜色来表示活动同步、完成的同步和失败的同步。若要设置颜色，请打开 MobiLink 监控器，选择 [**工具**] » [**选项**]，然后单击 [**一览表**] 选项卡。

请参见“[选项] 窗口”一节第 176 页。

可以通过取消选中 [**视图**] 菜单中的 [**一览表**] 窗格来关闭该窗格。

您还可以将 [**一览表**] 窗格与 MobiLink 监控器窗口的其余部分分隔开。在 [**选项**] 窗口中，打开 [**一览表**] 选项卡并清除 [**使一览表窗口附加到主窗口**] 复选框。

[选取框] 工具

[**选取框**] 工具为 [**一览表**] 窗格中出现的小框。您可以用 [**选取框**] 工具查看不同的数据或以不同的详细程度查看数据。该框内表示的区域将显示在 [图表] 和 [图形] 窗格中。可以按以下所述使用 [**选取框**] 工具：

- 在 [**一览表**] 窗格中单击可移动 [**选取框**] 工具和在图表或运用图形中所示数据的开始时间。
- 在 [**一览表**] 窗格中进行拖动以重画 [**选取框**] 工具，从而更改 [**选取框**] 工具的位置和大小，同时更改数据的开始时间和范围。如果您缩小选取框，则可以缩短 [图表] 窗格中可见数据的间隔，从而显示更详细的内容。

♦ 更改 [选取框] 工具的颜色

1. 选择 [**工具**] » [**选项**]。
2. 单击 [**一览表**] 选项卡。
3. 在 [**选取框**] 字段中选择一个新颜色。
4. 单击 [**确定**]。

[选项] 窗口

通过所提供的选项，您可以指定许多设置，包括在 [**图表**] 窗格和 [**一览表**] 窗格中图形显示的颜色和模式。

若要打开 [**选项**] 窗口，请打开 MobiLink 监控器，然后选择 [**工具**] » [**选项**]。

恢复缺省设置

若要恢复缺省设置，请删除文件 *mlMonitorSettings11*。此文件存储在您的用户配置文件目录中。

会话属性

[**会话属性**] 窗口提供与会话有关的统计信息。它可提供 MobiLink 监控器已运行总时间的属性值。若要打开 [**会话属性**] 窗口，请打开 MobiLink 监控器然后选择 [**文件**] » [**属性**]。

[**常规**] 选项卡提供与会话有关的基本信息。

[**统计**] 选项卡显示的统计信息与 [**示例属性**] 相同。请参见“示例属性”一节第 177 页。

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

示例属性

[**示例属性**] 窗口提供了时间间隔的详细统计信息。每个时间间隔长约一秒。示例由 MobiLink 监控器根据其接收的顺序进行编号。

可以自定义 [图形] 的外观来隐藏属性，但所有属性都将显示在 [**示例属性**] 窗口中。如果隐藏了某个属性，则其在 [**示例属性**] 窗口中被标识为 [**隐藏**]，否则将显示颜色。

要打开 [**示例属性**] 窗口，请单击要查看时间段的 [**图形**] 窗格。

[**示例**] 选项卡提供它表示的一秒时间间隔的信息。所显示的是针对时间间隔结束的属性。

[**范围**] 选项卡将显示属性窗口打开时可见示例的整个范围的平均值（水平范围在 [**一览表**] 中可见）。在您单击 [**范围**] 选项卡上的 [**计算**] 之后，才会对范围统计信息进行计算。

[**示例属性**] 包含下列信息：

- **示例** 示例由 MobiLink 监控器根据其接收的顺序进行编号。在 [**示例**] 选项卡上，显示为示例编号。在 [**范围**] 选项卡上，显示为示例范围。
- **开始时间和结束时间** 在 [**示例**] 选项卡上，反映为长约一秒的示例时间段。
- **统计 - [颜色] 列** 针对此属性在图形中所用的颜色。
- **统计 - [属性] 列** 显示该范围的示例或平均队列的队列长度。此列显示以下类型的属性：
  - **TCP/IP 工作队列** 这将列出等待通过从网络读取来填充的缓冲区数以及等待写入网络的缓冲区数。实际数字可能并没有什么意义，但数字过大可能提示存在与网络有关的瓶颈。
  - **流工作队列** 仅适用于版本 10 客户端。此队列表示在 MobiLink 服务器中由高级网络层完成的工作。该层负责高层级网络协议工作，例如，HTTP、加密和压缩。
  - **活动工作队列** 这是除同步外的定期内部 MobiLink 服务器任务的队列长度。
  - **命令处理器工作队列** 此为执行数据库任务以及解释或准备与 MobiLink 客户端进行通信的队列长度。实际数字可能并没有什么意义，但数字过大可能提示存在与数据库有关的瓶颈。
  - **数据库工作线程繁忙** 此值表明 MobiLink 服务器施压统一数据库的程度。此值的每个部分代表一个在数据库内正在忙碌的数据库工作线程。插入、更新、删除或选择之间没有区别。当此值为零时，服务器并未运行此统一数据库。

  注意：属性名从 MobiLink 服务器或 .mlm 文件中获得，并使用 MobiLink 服务器的语言。如果 MobiLink 监控器使用其它语言，则某些字符可能不会正确显示。
- **统计 - [值] 列** 属性值。
- **统计 - [限制] 列** 属性的最大允许值。这很重要，因此图形可对所有属性都使用 0-100% 的范围。对于如页面错误等实质上不绑定的属性，此限制将被忽略。

同步属性

在 [**详细信息表**] 窗格或 [**图表**] 中双击某个同步可以查看该同步的属性。

您可以选择显示所有表的统计信息（这是同步中所有表的总和），也可以选择只显示单独表的统计信息。该下拉列表提供在该同步中涉及的各表的列表。

要了解 [**同步属性**] 窗口的任何页面上所显示的数量的说明，请单击 [**帮助**]。

有关 [**同步属性**] 窗口中统计信息的说明，请参见“MobiLink 统计信息属性”一节第 181 页。

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

保存 MobiLink 监控器数据

您可以将来自 MobiLink 监控器会话的数据另存为一个二进制文件 (.mlm)、由逗号分隔值组成的文本文件 (.csv)、关系数据库表或者 Microsoft Excel 文件。

保存到文件

若要将数据保存为文件，请选择 [**文件**] » [**另存为**]。

- 如果您要在 MobiLink 监控器中查看保存的数据，则将数据另存为二进制 (.mlm) 文件。若要重新打开，请选择 [**文件**] » [**打开**]。二进制文件格式是唯一保存所有监控信息的格式。
- 如果您要在其它工具（例如 Microsoft Excel）中查看数据，则将数据另存为逗号分隔形式的文件 (*.csv*)。这只会保存同步属性窗口中的信息（各表信息以及会话结束时间除外）。您还可以在 MobiLink 监控器中打开 .csv 文件。

  在 *.csv* 文件格式中，持续时间以毫秒为单位存储。

您还可以指定要将数据自动保存到文件中。为此，请选择 [**工具**] » [**选项**]，然后在 [**常规**] 选项卡中输入输出文件名。该输出文件将被新数据覆盖。

导出到关系数据库或 Excel

您也可以使用 ODBC 连接导出 MobiLink 监控器数据。您可以将数据导出到任何 MobiLink 支持的关系数据库中，还可以导出到 Microsoft Excel 电子表格中。

导出数据时，MobiLink 监控器会话中的所有列都会被导出。此外，还会导出一个标识导出执行时间名为 export_time 的列。不导出图形中的数据。

由于某些列是保留字，因此数据源必须启用加引号的标识符。MobiLink 监控器自动为 SQL Anywhere、Adaptive Server Enterprise 和 Microsoft SQL Server 数据库启用加引号的标识符。如果未启用加引号的标识符选项，则导出将失败。

♦ 将数据导出到数据库或 Excel

1. 在收集完 MobiLink 监控器信息后，请断开与 MobiLink 服务器的连接。
2. 在 MobiLink 监控器中，选择 [**文件**] » [**导出到数据库**]。
3. 选择输出选项。
   - 您可以为将要创建的用于保存数据的两个表命名，也可以使用缺省名称。如果这两个表不存在，MobiLink 监控器将创建它们。如果输出到 Excel，这两个表名称将分别标识创建的两个工作表。
   - 选择是否要覆盖现有表中的数据。如果不选择覆盖数据，则新数据会附加到现有数据。
4. 单击 [**确定**] 以打开 [**连接**] 窗口，然后使用 ODBC 连接到数据库或 Excel 电子表格。

自定义您的统计信息

通过 [**监视项目管理器**]，您可以明确区分满足指定条件的同步。例如，您可能要突出显示大同步、长同步、占用较长时间的小同步或收到警告的同步。

若要打开 [**监视项目管理器**]，请打开 MobiLink 监控器，然后单击 [**工具**] » [**监视项目管理器**]。

[**监视项目管理器**] 的左窗格包含所有可用监视项目的列表。右窗格包含活动监视的列表。若要向活动列表添加监视项目或从活动列表中删除监视项目，请在左窗格中选择一个监视项目并单击适当的按钮。

有三种预定义的监视项目（[**活动**]、[**已完成**] 和 [**失败**]）。您可以编辑预定义的监视项目以更改显示这些监视项目的方式，并且可以通过从右窗格中删除监视项目来停用它们。

[图表] 中只显示满足监视项目条件的同步。如果您禁用所有监视项目（通过从 [**当前监视项目**] 列表中删除它们），则在 [**图表**] 或 [**一览表**] 窗格中将不会显示任何同步。

监视项目在右窗格中的顺序十分重要。靠近列表顶部的监视项目先被处理。使用 [**上移**] 和 [**下移**] 按钮可以组织右窗格中监视项目的顺序。

您可以使用预定义的监视项目，也可以创建其它监视项目。若要编辑监视项目条件，请先将其删除然后再添加新监视项目条件。

新的 MobiLink 监控器连接到同一 MobiLink 服务器时，它将在任何已经连接的 MobiLink 监控器上显示为一个短暂的同步过程。MobiLink 监控器同步具有 for_ML_Monitor_only 的版本名称。可以使用监视项目来隐藏此 MobiLink 监控器同步。

♦ 创建新监视

1. 在 [**监视项目管理器**] 中，单击 [**新建**]。
2. 在 [**名称**] 框中提供该监视项目的名称。
3. 选择 [**属性**]、比较 [**运算符**] 和 [**值**]。

   有关属性的完整列表，请参见“MobiLink 统计信息属性”一节第 181 页。
4. 单击 [**添加**]。（必须单击 [**添加**] 才能保存这些设置。）
5. 如果需要，选择其它的 [**属性**]、[**运算符**] 和 [**值**]，然后单击 [**添加**]。
6. 在 [**图表**] 窗格中选择该监视项目的 [**图表模式**]。
7. 在 [**一览表**] 窗格中选择该监视项目的 [**一览表颜色**]。

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

MobiLink 统计信息属性

下面是在 MobiLink 监控器中提供的属性的列表。这些统计信息可以在 [**新建监视项目**] 窗口、[**详细信息表**] 窗格或 [**同步属性**] 窗口中查看。在 [**同步属性**] 窗口中，属性名称不包含下划线。

有关 [**新建监视项目**] 窗口的详细信息，请参见“自定义您的统计信息”一节第 180 页。

有关 [**详细信息表**] 的详细信息，请参见“[详细信息表] 窗格”一节第 171 页。

有关 [**同步属性**] 窗口的详细信息，请参见“同步属性”一节第 177 页。

同步统计信息

当未使用强制冲突模式时，MobiLink 统计信息属性返回同步的下列信息。

有关强制冲突模式的信息，请参见“强制冲突统计信息”一节第 183 页。

属性	说明
active	如果正在进行同步则为真。
apply_upload	上载数据应用到统一数据库所需的时间。
authenticate_user	执行用户验证（包括执行 authenticate_* 事件）的总时间。
begin_sync	begin_synchronization 事件的总时间。
completed	如果同步已成功完成则为真。
conflicted_deletes	始终为零。
conflicted_inserts	始终为零。
conflicted_updates	导致冲突的更新行数目。仅当对某行成功调用了解决冲突脚本时才包括该行。
connect	当需要新的数据库连接时，数据库工作线程建立数据库连接所需的时间。例如，出现错误后或脚本版本发生变化时。
connect_for_download_ack	当需要新的数据库连接时，数据库工作线程建立数据库连接所需的时间。
connection_retries	MobiLink 服务器重试连接统一数据库的次数。
download_bytes	MobiLink 服务器内用来存储下载的内存量。这可以更好地指示同步对服务器内存的影响。
download_deleted_rows	MobiLink 服务器从统一数据库中取得的行删除数（使用 download_delete_cursor 脚本）。
download_errors	下载过程中发生的错误数。

属性	说明
download_fetched_rows	MobiLink 服务器从统一数据库中取得的行数（使用 download_cursor 脚本）。
download_filtered_rows	由于与 MobiLink 客户端上载的行相匹配而没有下载到此客户端的读取的行数。
download_warnings	下载过程中发生的警告数。
duration	由 MobiLink 服务器测量的同步的总时间。
end_sync	end_synchronization 事件的总时间。
fetch_download	读取要从统一数据库下载的行以创建下载流所需的时间。
get_db_worker	获得空闲数据库工作线程所需的时间。
get_db_worker_for_download_ack	收到下载确认后，等待空闲数据库工作线程所花费的时间。
ignored_deletes	如果定义了 handle_error 或 handle_odbc_error 并返回 1000 或者对给定表没有定义 upload_delete 脚本，当调用 upload_delete 脚本时导致出错的上载删除行数。
ignored_inserts	忽略的上载插入行总数。它们被忽略的原因是：1) 在正常模式中没有 upload_insert 脚本或在强制冲突模式中没有 upload_new_row_insert 脚本，2) MobiLink 服务器调用相应脚本时出现了错误并且 handle_error 或 handle_odbc_error 事件返回 1000。
ignored_updates	导致冲突但解决冲突脚本未能成功调用或者未定义 upload_update 脚本的上载更新行数。
nonblocking_download_ack	publication_nonblocking_download_ack 连接事件和 nonblocking_download_ack 连接事件所需的时间。
prepare_for_download	prepare_for_download 事件总时间。
remote_id	用于唯一地标识远程数据库的远程 ID。
send_download	将下载流发送到远程数据库所需的时间。该时间取决于下载流的大小以及网络传输的带宽。对于仅上载同步，下载流只是一个上载确认。
start_time	同步开始的日期时间（ISO-8601 扩展格式）。
sync	唯一地标识 MobiLink 监控器会话中的同步的编号。
sync_deadlocks	统一数据库中为同步已检测到的死锁的数目。

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

属性	说明
sync_errors	同步时发生的错误总数。
sync_request	从建立远程数据库与 MobiLink 服务器之间的网络连接开始直至接收到上载流的第一批字节为止所花费的时间。
sync_tables	同步中涉及的客户端表数。
sync_warnings	同步时发生的警告数。
upload_bytes	MobiLink 服务器内用来存储上载的内存量。这可以更好地指示同步对服务器内存的影响。
upload_deadlocks	上载过程中在统一数据库中检测到的死锁数。
upload_deleted_rows	从统一数据库中成功删除的行数。
upload_errors	上载期间发生的错误的数目。
upload_inserted_rows	在统一数据库中成功插入的行数。
upload_updated_rows	在统一数据库中成功更新的行数。
upload_warnings	上载期间发生的警告的数目。
user	MobiLink 用户名。
version	同步版本的名称。
wait_for_download_ack	等待将下载应用到远程数据库以及远程数据库发送下载确认所花费的时间。

强制冲突统计信息

当处于强制冲突模式时，MobiLink 统计信息属性返回下列信息。

统计属性	说明
conflicted_deletes	使用 upload_old_row_insert 脚本成功插入到统一数据库内的上载删除行数。
conflicted_inserts	使用 upload_new_row_insert 脚本插入到统一数据库内的上载插入行数。
conflicted_updates	使用 upload_new_row_insert 或 upload_old_row_insert 脚本成功应用的上载更新行数。
ignored_deletes	如果定义了 handle_error 或 handle_odbc_error 并返回 1000 或者对给定表没有定义 upload_old_row_insert 脚本，当调用 upload_old_row_insert 脚本时导致出错的上载删除行数。

统计属性	说明
ignored_inserts	如果定义了 handle_error 或 handle_odbc_error 并返回 1000 或者对给定表没有定义 upload_new_row_insert 脚本，当调用 upload_new_row_insert 脚本时导致出错的上载插入行数。
ignored_updates	如果定义了 handle_error 或 handle_odbc_error 并返回 1000 或者对给定表没有已定义的 upload_new_row_insert 和 upload_old_row_insert 脚本，当调用 upload_new_row_insert 或 upload_old_row_insert 脚本时导致出错的上载更新行数。
upload_deleted_rows	始终为零。
upload_inserted_rows	始终为零。
upload_updated_rows	始终为零。

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

用于 MobiLink 的 SQL Anywhere 监控器

目录

SQL Anywhere 监控器简介 186
监控器快速入门 189
教程：使用监控器 190
启动监控器 195
退出监控器 196
连接到监控器 197
断开与监控器的连接 198
监控资源 199
管理资源 206
使用监控器用户 212
警告 216
在一台单独的计算机上安装 SQL Anywhere 监控器 220
监控器疑难解答 221

SQL Anywhere 监控器简介

SQL Anywhere 监控器（也称为监控器）是一种基于 Web 浏览器的管理工具，可提供有关 SQL Anywhere 数据库和 MobiLink 服务器健康情况和可用性的信息。

本章介绍如何使用监控器来收集有关 MobiLink 服务器的度量。有关将监控器与 SQL Anywhere 数据库配合使用的信息，请参见“SQL Anywhere 监控器”《SQL Anywhere 服务器 - 数据库管理》。

此监控器提供以下功能：

- **常量数据收集** 与 SQL Anywhere 11 提供的许多其它管理工具不同，监控器会一直收集度量，甚至在您未登录到 Web 浏览器时也是如此。除非将此监控器关闭，否则其会一直收集度量。
- **电子邮件警告通知** 当收集到度量时，此监控器会对度量进行检查，并在检测到指示 MobiLink 服务器出现错误的条件时发送电子邮件警告。
- **基于浏览器的界面** 您可以随时使用 Web 浏览器连接到此监控器，以查看警告和收集到的度量。
- **监控多个数据库和 MobiLink 服务器** 通过同一个工具，您可以同时监控在一台计算机或不同计算机上运行的 SQL Anywhere 数据库和 MobiLink 服务器。

  有关监控 SQL Anywhere 数据库的信息，请参见“SQL Anywhere 监控器”《SQL Anywhere 服务器 - 数据库管理》。
- **最小化性能影响** 因为监控不会降低性能，所以可以在开发与生产环境中例行使用此监控器。

要求

- 建议安装可用于操作系统的最新版 Adobe Flash Player。监控器向后兼容 Adobe Flash Player 版本 9。要确定正确的版本，请访问 http://www.adobe.com/products/flashplayer/systemreqs/。
- 必须在 Web 浏览器中启用 JavaScript。
- 必须已安装 SQL Anywhere 11.0.1。

在生产环境中运行监控器

可以在一台单独的计算机上安装并运行监控器。这样可以防止监控器资源和配置在后续的 SQL Anywhere 升级或更新过程中被覆盖。如果要在生产环境中使用监控器，建议在一台单独的计算机上安装监控器。请参见“在一台单独的计算机上安装 SQL Anywhere 监控器”一节《SQL Anywhere 服务器 - 数据库管理》。

限制

- 可使用监控器收集有关以下类型 SQL Anywhere 数据库和 MobiLink 服务器的度量：
  - SQL Anywhere 9.0.2、10.0.0、10.0.1、11.0.0 和 11.0.1
  - MobiLink 11.0.0（至少应用了第一个 EBF）和 11.0.1
- 在一台计算机上只能运行一个监控器。

Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

- 此监控器不提供关于单个同步的信息。要获得关于单个同步的详细信息（包括计时和每个同步的其它统计信息），请使用 MobiLink 监控器。请参见“MobiLink 监控器简介”一节第 168 页。

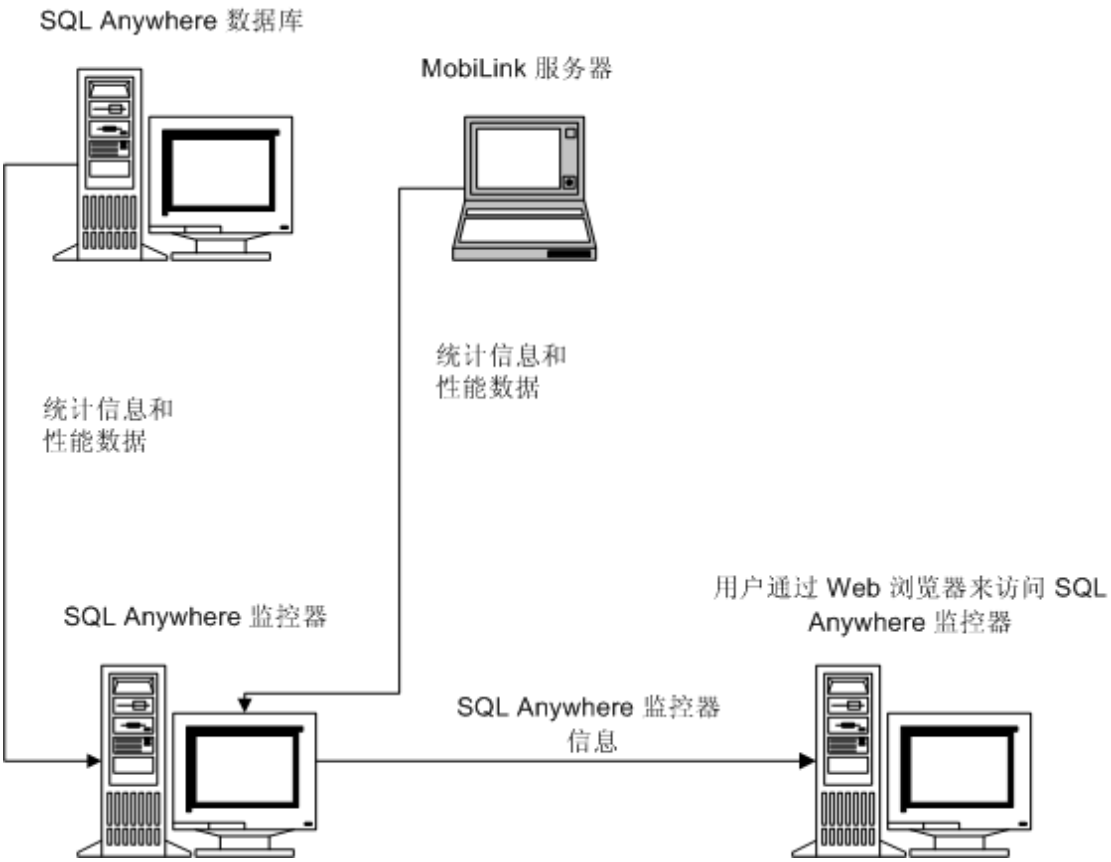
另请参见

有关可用于 MobiLink 服务器的其它管理工具和性能工具的信息，请参见：

- “MobiLink 监控器”第 167 页

监控器体系结构

此监控器从运行于其它计算机上的 SQL Anywhere 数据库和 MobiLink 服务器收集度量和性能数据，而单独的计算机通过 Web 浏览器来访问此监控器。



此监控器设计用于帮助任何类型的用户（无论他们是否为 DBA），这些用户负责如下任务：

- 确保 MobiLink 服务器已连接到网络。
- 确保 MobiLink 服务器有足够的磁盘空间或内存可用。
- 查看在指定时间内 MobiLink 服务器执行的同步数。

另请参见

- “监控器快速入门”一节第 189 页

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

监控器快速入门

设置 MobiLink 服务器监控需要执行以下步骤：

1. 将 SQL Anywhere 11.0.1 安装到始终与网络相连的一台计算机上。此监控器使用 SQL Anywhere 来监控 MobiLink 服务器。

   监控器可以与其所监控的资源运行在同一台计算机上，但建议您在另一台计算机上运行监控器（在生产环境中时尤其如此），以将对 MobiLink 服务器或其它应用程序的影响降至最低。

2. 确保您的 Web 浏览器已安装了适当版本的 Adobe Flash Player 并且已启用了 JavaScript。请参见“要求”一节第 186 页。

3. 启动 MobiLink 服务器（如果其尚未运行）。

4. 启动监控器并在 Web 浏览器中将其打开。请参见“启动监控器”一节第 195 页。

   在其中使用 Web 浏览器来访问监控器的计算机必须连接到此监控器运行的网络。

5. 以管理员身份登录。缺省用户名为 **admin**，缺省口令也是 **admin**。

6. 单击 [**管理**] 选项卡，并添加 MobiLink 服务器作为要监控的资源。请参见“添加资源”一节第 206 页。

7. 添加新用户并更改管理员用户口令。请参见“创建监控器用户”一节 《SQL Anywhere 服务器 - 数据库管理》。

8. 为要监控的 MobiLink 服务器配置警告。请参见“警告”一节 《SQL Anywhere 服务器 - 数据库管理》。

9. 单击 [**监控**] 选项卡可查看为 MobiLink 服务器收集的度量。请参见“监控资源”一节第 199 页。

Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

教程：使用监控器

使用本教程设置对 MobiLink 同步服务器示例的监控。

第 1 课：启动监控器

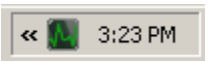
♦ **启动并打开监控器**

1. 启动监控器。选择 [**开始**] » [**程序**] » [**SQL Anywhere 11**] » [**SQL Anywhere 监控器**] » [**启动 SQL Anywhere 监控器**]。

   如果是在一台单独的计算机上安装监控器，则不必执行此步骤。如果安装监控器的计算机不同于 SQL Anywhere 所运行的计算机，则监控器将作为服务来运行并在计算机启动时自动启动。

2. 浏览数据。根据监控器是否在一台单独的计算机上安装，此步骤会有所不同。

   在系统任务栏中，右击 SQL Anywhere 监控器图标并选择 [**浏览数据**]。

   如果监控器安装在一台单独的计算机上，选择 [**开始**] » [**程序**] » [**SQL Anywhere 监控器 11**] » [**浏览数据**]。系统任务栏中不会出现图标。

   或者，您也可以打开 Web 浏览器，然后浏览到 *http://localhost:4950*。

   [**监控**] 选项卡的顶部窗格列出了正在受监控的资源。首次打开监控器时，其只监控它本身。

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

另请参见

- “第 2 课：设置监控器来监控 MobiLink 服务器”一节第 191 页

第 2 课：设置监控器来监控 MobiLink 服务器

此监控器从数据库和 MobiLink 服务器收集度量。在本节中，您将启动 MobiLink 同步服务器示例，然后将此服务器添加为要监控的资源。要从 SQL Anywhere 数据库收集度量，请参见“第 2 课：设置监控器来监控数据库”一节 《SQL Anywhere 服务器 - 数据库管理》。

♦ 将资源添加到监控器

1. 启动 MobiLink 同步服务器示例。

   从 [**开始**] 菜单选择 [**程序**] >> [**SQL Anywhere 11**] >> [**MobiLink**] >> [**同步服务器示例**]。
2. 以缺省管理员身份登录到监控器：
   a. 单击 [**登录**]。
   b. 在 [**用户名**] 字段中，键入 **admin**，然后在 [**口令**] 字段中键入 **admin**。
   c. 单击 [**登录**]。
3. 单击 [**管理**] 选项卡。
4. 单击 [**资源**] 选项卡。
5. 单击 [**添加**]。
6. 选择 [**MobiLink 服务器**]，然后单击 [**下一步**]。
7. 将资源命名为 **MobiLinkServerSample**，然后单击 [**下一步**]。
8. 在 [**主机**] 字段中，键入 **localhost**，然后单击 [**下一步**]。
9. 当系统提示您需要授权时，在 [**用户 ID**] 字段中键入用户名（如 user1），然后在 [**口令**] 字段中键入口令（如 sql）。

   这些证书用于连接到 MobiLink 服务器。用户 ID 和口令是由监控器保存的。
10. 单击 [**创建**]。
11. 将创建新资源 MobiLinkServerSample，并且开始监控。
12. 单击 [**确定**]。
13. 单击 [**监控**] 选项卡。

    MobiLinkServerSample 资源将出现在 [**监控**] 选项卡上，而收集的度量出现在底部窗格的选项卡中。

第 3 课：测试警告

在本课中，您将有意地触发警告，以便能够练习如何处理警告。

♦ **查看和解析警告**

1. 通过关闭 MobiLink 同步服务器示例触发警告。
   a. 在 Windows 上，双击系统任务栏中 MobiLink 服务器的 MobiLink 服务器图标。
   b. 在 MobiLink 服务器窗口中单击 [**关闭**]。
   c. 单击 [**是**]。
2. 在监控器中，单击 [**监控**] 选项卡。

   MobiLinkServerSample 资源的 [**状态**] (State) 将变为 [**服务器关闭**]；MobiLinkServerSample 资源的 [**状态**] (Status) 将变为 [**需要注意！**]。

   状态 (State) 和状态 (Status) 可能在几秒钟之后才会发生变化。缺省情况下，监控器将每隔 30 秒从资源收集一次信息。
3. 在底部窗格中，单击 [**警告**]。
4. 选择 [**可用性警告**]，然后单击 [**详细信息**] 以读取说明。
5. 单击 [**确定**]。
6. 重新启动同步服务器示例。

   从 [**开始**] 菜单选择 [**程序**] ? [**SQL Anywhere 11**] ? [**MobiLink**] ? [**同步服务器示例**]。

   MobiLinkServerSample 资源的 [**状态 (State)**] 将变为 [**活动**]，而 [**状态 (Status)**] 不会发生任何变化。变化要在几分钟之后才会出现。
7. 通过选择警告并单击 [**删除**] 来将其删除。

   [**状态**] 将变为 [**良好**]。

第 4 课：设置监控器以在出现警告时发送电子邮件

当警告出现时，其总是会列在 [**监控**] 选项卡的下部窗格中的 [**警告**] 选项卡内。在以下步骤中，将设置监控器以在出现警告时向您发送电子邮件。

♦ **设置电子邮件通知**

1. 创建一个可接收电子邮件的用户。
   a. 单击 [**管理**] 选项卡。
   b. 单击 [**用户**] 选项卡。
   c. 单击 [**新建**]。
   d. 在 [**用户名**] 字段中键入 **JoeSmith**。
   e. 在 [**口令**] 和 [**确认口令**] 字段中键入 **sql**。
   f. 在 [**电子邮件**] 字段中输入有效的电子邮件地址。
   g. 在 [**首选语言**] 字段中，选择 [**中文**]。

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

h. 为 [**用户类型**] 选择 [**操作员**]。

操作员可通过电子邮件接收警告，并可解析和删除警告。该用户可访问 [**监控**] 选项卡，但是不能访问 [**管理**] 选项卡。

有关不同类型用户的信息，请参见“使用监控器用户”一节第 212 页。

i. 单击 [**保存**]。

即会创建新用户。

2. 将用户与 MobiLinkServerSample 资源相关联。

a. 单击 [**资源**] 选项卡。

b. 选择 [**MobiLinkServerSample**] 资源，然后单击 [**配置**]。

c. 在 [**配置资源**] 窗口中，单击 [**操作员**]。

d. 在 [**可用操作员**] 列表中，选择 **JoeSmith** 并单击 [**添加**]。

e. 单击 [**保存**]。

f. 单击 [**确定**]。

3. 配置电子邮件警告通知。

a. 单击 [**管理**] 选项卡。

b. 单击 [**配置**] 选项卡。

c. 单击 [**编辑**]。

d. 选择 [**通过电子邮件发送警告通知**]。

e. 根据需要配置其它设置。

f. 测试您是否已正确配置了电子邮件通知。

单击 [**发送测试电子邮件**]。

g. 在系统提示时，输入要将测试电子邮件发送到的电子邮件地址，然后单击 [**确定**]。

测试电子邮件将会被发送到指定的电子邮件地址。

h. 单击 [**保存**]。

发生警告时，会将带有关于此警告的信息的电子邮件发送给指定用户。有关设置警告的信息，请参见“第 3 课：测试警告”一节第 191 页。

第 5 课：清除

以下步骤将删除 MobiLinkServerSample 资源，进而删除收集到的度量并停止数据收集。在生产环境中，当您想要持续监控 MobiLink 服务器时，应使 MobiLink 服务器和监控器都处于运行状态。

♦ 停止监控

1. 删除 MobiLinkServerSample 资源。

a. 单击 [**管理**] 选项卡。

b. 单击 [**资源**] 选项卡。

c. 选择 MobiLinkServerSample 资源，然后单击 [**停止**]。

d. 单击 [**删除**]。

e. 单击 [**是**]，确认您确实要删除该资源。

2. 注销监控器。

   单击 [**注销**]。

3. 关闭您正在其中查看监控器的 Web 浏览器窗口。

4. 退出监控器。

   在系统任务栏中，右击 SQL Anywhere 监控器图标并选择 [**退出 SQL Anywhere 监控器**]。

5. 关闭 MobiLink 服务器。

   a. 双击系统任务栏中 MobiLink 同步服务器示例的 MobiLink 服务器图标。

   b. 在 MobiLink 服务器消息窗口中单击 [**关闭**]。

   c. 单击 [**是**]。

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

启动监控器

启动监控器会使监控器开始为此监控器中的*所有*资源收集度量。

根据监控器是否运行在一台单独的计算机上，启动监控器的过程会有所不同。

♦ **启动监控器**

1. 选择 [**开始**] » [**程序**] » [**SQL Anywhere 11**] » [**SQL Anywhere 监控器**] » [**启动 SQL Anywhere 监控器**]。

   SQL Anywhere 监控器图标出现在系统任务栏中。
2. 连接到监控器。请参见“连接到监控器”一节第 197 页。

♦ **启动安装在单独计算机上的监控器**

1. 如果监控器安装在单独的计算机上，则它将作为服务而自动运行。但是，如果停止监控，您可以重新启动它。为此，请到 *install-dir\bin32*。
2. 在 Windows 上，运行以下命令：

   ```
   samonitor.bat start service
   ```

   在 Linux 上，运行以下命令：

   ```
   samonitor.sh start service
   ```

   当监控器作为服务来运行时，系统任务栏中不会出现 SQL Anywhere 监控器图标。
3. 连接到监控器。请参见“连接到监控器”一节第 197 页。

另请参见

- “退出监控器”一节 《SQL Anywhere 服务器 - 数据库管理》
- “连接到监控器”一节 《SQL Anywhere 服务器 - 数据库管理》
- “断开与监控器的连接”一节 《SQL Anywhere 服务器 - 数据库管理》
- “监控资源”一节 《SQL Anywhere 服务器 - 数据库管理》

Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

退出监控器

退出监控器将会停止为所有资源收集度量。建议您使监控器处于运行状态，而将 Web 浏览器关闭。要停止监控特定的 MobiLink 服务器，请参见“停止监控资源”一节第 209 页。

根据监控器是否运行在一台单独的计算机上，退出监控器的过程会有所不同。

♦ 退出监控器

- 在系统任务栏中，右击 SQL Anywhere 监控器图标并选择 [**退出 SQL Anywhere 监控器**]。

♦ 退出安装在单独计算机上的监控器

1. 到 *install-dir\bin32*。
2. 在 Windows 上，运行以下命令：

   ```
   samonitor.bat stop service
   ```

   在 Linux 上，运行以下命令：

   ```
   samonitor.sh stop service
   ```

另请参见

- “启动监控器”一节 《SQL Anywhere 服务器 - 数据库管理》
- “连接到监控器”一节 《SQL Anywhere 服务器 - 数据库管理》
- “断开与监控器的连接”一节 《SQL Anywhere 服务器 - 数据库管理》
- “监控资源”一节 《SQL Anywhere 服务器 - 数据库管理》

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

连接到监控器

用于连接到监控器的计算机必须连接到此监控器正在运行的网络。

♦ 连接到监控器

1. 启动监控器（如果其尚未运行）。请参见“启动监控器”一节第 195 页。
2. 浏览数据。根据监控器是否在一台单独的计算机上安装，此步骤会有所不同。

   从 [**开始**] 菜单中，选择 [**程序**] » [**SQL Anywhere 11**] » [**SQL Anywhere 监控器**] » [**浏览数据**]。

   如果监控器安装在一台单独的计算机上，选择 [**开始**] » [**程序**] » [**SQL Anywhere 监控器 11**] » [**浏览数据**]。

   Web 浏览器将打开用于连接到此监控器的缺省 URL：**http://***computer-name***:4950**，其中 *computer-name* 是正在运行监控器的计算机的名称。例如，*http://localhost:4950*。
3. 如果系统提示，则输入监控器的用户名和口令。监控器的用户名和口令是区分大小写的。请参见“使用监控器用户”一节第 212 页。

另请参见

- “启动监控器”一节 《SQL Anywhere 服务器 - 数据库管理》
- “退出监控器”一节 《SQL Anywhere 服务器 - 数据库管理》
- “断开与监控器的连接”一节 《SQL Anywhere 服务器 - 数据库管理》
- “监控资源”一节 《SQL Anywhere 服务器 - 数据库管理》

Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

断开与监控器的连接

可通过注销或关闭 Web 浏览器断开与监控器的连接。

断开与监控器的连接并不会对度量的收集造成影响。如果要停止对度量的收集，则请停止对资源的监控或退出监控器。请参见“停止监控资源”一节第 209 页或“退出监控器”一节第 196 页。

♦ 断开与监控器的连接

- 单击 [**注销**]。

另请参见

- “启动监控器”一节 《SQL Anywhere 服务器 - 数据库管理》
- “退出监控器”一节 《SQL Anywhere 服务器 - 数据库管理》
- “连接到监控器”一节 《SQL Anywhere 服务器 - 数据库管理》
- “监控资源”一节 《SQL Anywhere 服务器 - 数据库管理》

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

监控资源

在监控器中，[**监控**] 选项卡会提供有关所监控 MobiLink 服务器的健康和可用性的概要信息。

[监控] 选项卡

顶部窗格包含一个列有正在受监控的资源的表格。**资源**是指 MobiLink 服务器。该表还指明资源当前是否运行以及资源是否需要用户对其执行任何操作。请参见“解释资源状态 (states) 和状态 (status)”一节第 200 页。

[**监控**] 选项卡的底部窗格包含所选 MobiLink 服务器的警告和多种当前度量。这些选项卡中的大部分选项卡都包含指向图形的链接。可使用每个图形右上方的下拉列表和箭头更改图形的范围。

[管理] 选项卡

[**管理**] 选项卡为管理员所保留。在该选项卡上，您可以选择所要监控的 MobiLink 服务器，添加和编辑用户，以及配置监控器。

另请参见

- “使用监控器用户”一节第 212 页
- “监控器度量”一节第 200 页

解释资源状态 (states) 和状态 (status)

[**监控**] 选项卡的顶部窗格包含一个列出了正在受监控的 MobiLink 服务器的表格。在此表中，[**状态 (State)**] 列提供了关于监控器与其资源之间的连接的信息。[**状态 (Status)**] 列指明该资源是否需要操作员或管理员用户对其执行操作。请参见“使用监控器用户”一节第 212 页。

资源状态 (state)

资源始终处于以下状态之一：

- **活动** 资源已连接且监控器正在收集度量。
- **封锁** 监控器在其重新开始监控资源之前，正在等待封锁期结束。
- **服务器关闭** 正在受监控的 MobiLink 服务器已停止。
- **主机关闭** 监控器无法找到承载资源的计算机。
- **未知** 此监控器没有监控资源。

资源状态 (status)

资源具有以下状态之一：

- **良好** 资源没有未解决的警告。
- **需要注意** 资源有一个或多个警告。
- **监控已停止** 资源未受监控。
- **未知** 资源未活动且没有警告。

监控器度量

监控器从 MobiLink 服务器收集并存储度量，其中包括（但不局限于）：

- 资源是否正在运行。
- 资源正在其中运行的计算机是否运行正常以及是否连接到网络。
- 资源是否正在监听和处理请求。
- MobiLink 服务器在一段时间内执行的同步数。

度量的收集速率是由管理员设置的收集间隔设置所决定的。请参见“收集间隔”一节第 206 页。

收集哪些度量以及使用什么阈值来发出警告，由管理员设置的度量设置所决定。请参见“指定要收集的度量”一节第 207 页。

显示度量

监控器的显示将每分钟自动刷新一次。可通过单击 [**用户设置**] 更改刷新间隔。该设置与资源的收集间隔速率无关，收集间隔速率用于指定监控器从所监控资源收集度量的频率。

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

♦ **设置刷新率**

1. 单击右上角的 [**用户设置**]。
2. 为 [**刷新间隔**] 设置时间。缺省值为一分钟。
3. 单击 [**确定**]。

在 [**监控**] 选项卡上单击 [**刷新数据**] 时，监控器会检索并显示最新的度量。

♦ **刷新度量**

- 单击 [**刷新数据**]。

按 F5 键时，监控器会重装 Web 浏览器，以及检索和显示监控器到目前为止所收集的度量。

♦ **重装监控器**

- 按 F5。

[度量] 选项卡说明

以下选项卡由 SQL Anywhere 和 MobiLink 服务器资源使用。

- “[监控] 选项卡：[警告] 选项卡”一节第 202 页
- “[监控] 选项卡：[服务器] 选项卡”一节第 202 页

以下选项卡仅由 SQL Anywhere 资源使用。

- “[监控] 选项卡：[CPU] 选项卡”一节 《SQL Anywhere 服务器 - 数据库管理》
- “[监控] 选项卡：[未调度的请求] 选项卡”一节 《SQL Anywhere 服务器 - 数据库管理》
- “[监控] 选项卡：[内存] 选项卡”一节 《SQL Anywhere 服务器 - 数据库管理》
- “[监控] 选项卡：[磁盘] 选项卡”一节 《SQL Anywhere 服务器 - 数据库管理》
- “[监控] 选项卡：[HTTP] 选项卡”一节 《SQL Anywhere 服务器 - 数据库管理》
- “[监控] 选项卡：[连接] 选项卡”一节 《SQL Anywhere 服务器 - 数据库管理》
- “[监控] 选项卡：[失败的连接] 选项卡”一节 《SQL Anywhere 服务器 - 数据库管理》
- “[监控] 选项卡：[查询] 选项卡”一节 《SQL Anywhere 服务器 - 数据库管理》
- “[监控] 选项卡：[镜像] 选项卡”一节 《SQL Anywhere 服务器 - 数据库管理》

以下选项卡仅由 MobiLink 服务器资源使用。

- “[监控] 选项卡：[同步] 选项卡”一节第 203 页
- “[监控] 选项卡：[统一数据库] 选项卡”一节第 203 页

Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

- “[监控] 选项卡：[计算机资源] 选项卡”一节第 204 页

[监控] 选项卡：[警告] 选项卡

列出前五十个最新的警告。一旦列表中的警告数超过 50 个，则旧的警告即会在新的警告出现时被删除。请参见“警告”一节第 216 页。

[监控] 选项卡：[服务器] 选项卡

MobiLink 服务器

- **服务器名** 由 -zs 选项所指定的用于已连接服务器的 MobiLink 服务器的名称。缺省值为 **<default>**。请参见“-zs 选项”一节第 112 页。
- **版本** 显示正在运行的软件版本。
- **开始时间** 显示 MobiLink 服务器启动的时间。
- **未提交的错误报告** 显示服务器的未提交错误报告数。SQL Anywhere 软件崩溃时将提交错误报告。请参见“取消对资源的未提交错误报告的警告”一节第 219 页。

许可

- **被授权公司的名称** 显示被授权公司的名称。
- **被授权用户的名称** 显示被授权用户的名称。

Host

- **名称** 显示运行 MobiLink 服务器的计算机的名称。通常，这是计算机的主机名。
- **操作系统平台** 显示运行软件的操作系统。
- **处理器体系结构** 显示用于标识处理器类型的字符串。
- **CPU 数** 显示运行该软件的计算机所包含的 CPU 的数量。

其它信息

- **统一数据库类型** 显示统一数据库的类型。例如 SQL Anywhere。
- **到数据库的最大并发上载数** 显示到数据库的最大并发上载数。请参见“-wu 选项”一节第 101 页。
- **数据库工作线程** 显示数据库工作线程的数目。请参见“-w 选项”一节第 100 页。
- **最大高速缓存大小** 由 mlsrv11 的 -cm 选项所设置的 MobiLink 服务器内存高速缓存的最大大小。请参见“-cm 选项”一节第 51 页。
- **高速缓存中的最大页数** MobiLink 内存高速缓存中的页数。它由 mlsrv11 的 -cm 选项隐式设置。请参见“-cm 选项”一节第 51 页。

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

- **最大数据库连接数** 最大数据库连接数由 mlsrv11 的 -cn 选项或 -w 选项设置。请参见“-cn 选项”一节第 52 页和“-w 选项”一节第 100 页。
- **最大 TCP 连接数** 最大 TCP 连接数由 mlsrv11 的 -nc 选项设置。请参见“-nc 选项”一节第 71 页。
- **最大客户端数** 最大客户端数量。请参见“-sm 选项”一节第 89 页。
- **统一版本** 显示统一数据库的版本。
- **驱动程序版本** 显示统一数据库的驱动程序版本。
- **驱动程序名称** 显示统一数据库的驱动程序的名称。
- **群中的主服务器** 指出此服务器是主服务器还是次服务器。

另请参见

- “[监控] 选项卡：[服务器] 选项卡”一节第 202 页

[监控] 选项卡：[同步] 选项卡

监控 MobiLink 服务器时使用此选项卡。

- **同步完成率** 显示服务器的同步完成率（以每秒的同步数表示）。
- **同步失败率** 显示服务器的同步失败率（以每秒的同步数表示）。
- **同步错误率** 显示同步的错误率（以每秒的错误数表示）。
- **同步警告率** 显示服务器的同步警告率（以每秒的警告数表示）。
- **最长活动同步时间** 以秒为单位显示服务器的最早活动同步所历经的时间。
- **活动请求** 显示服务器中的活动请求的数量。
- **应用上载** 显示服务器中当前处于请求上载或开始同步阶段的请求数。有关同步阶段的说明，请参见“-v 选项”一节第 97 页。
- **生成下载** 显示服务器中当前处于准备下载、读取下载、等待下载确认或结束同步阶段的请求数。有关同步阶段的说明，请参见“-v 选项”一节第 97 页。
- **活动验证** 显示服务器中当前处于验证用户阶段的请求数。有关同步阶段的说明，请参见“-v 选项”一节第 97 页。

[监控] 选项卡：[统一数据库] 选项卡

监控 MobiLink 服务器时使用此选项卡。

- **使用中的连接** 显示当前正在由此 MobiLink 服务器使用的数据库连接的数量。
- **数据库工作线程的最长活动等待** 显示活动请求等待此服务器中的数据库工作线程的最长时间。
- **等待数据库工作线程** 显示服务器中当前正在等待数据库工作线程的请求数量。

- **使用中的上载连接数** 显示服务器中当前正在使用的上载连接数。

[监控] 选项卡：[计算机资源] 选项卡

监控 MobiLink 服务器时使用此选项卡。

- **CPU 使用** 显示 MobiLink 服务器使用的 CPU 时间的百分比。
- **CPU 总时间** 以秒为单位显示 MobiLink 服务器所使用的 CPU 的总时间。
- **使用的 MobiLink 高速缓存页面** 显示由服务器使用的 MobiLink 高速缓存页面的百分比。它由 mlsrv11 的 -cm 选项隐式设置。请参见“-cm 选项”一节第 51 页。
- **锁定的 MobiLink 高速缓存页面** 显示装载到服务器内存中的 MobiLink 高速缓存页面的百分比。它由 mlsrv11 的 -cm 选项隐式设置。请参见“-cm 选项”一节第 51 页。
- **转入的 MobiLink 高速缓存页面** 显示服务器每秒从磁盘中读取的 MobiLink 高速缓存页面数。
- **转出的 MobiLink 高速缓存页面** 显示每秒与磁盘交换的 MobiLink 高速缓存页面数。
- **内存使用** 显示服务器所使用的 RAM 的字节数（仅限 Windows 服务器）。
- **MobiLink 高速缓存的可用磁盘空间** 以字节为单位显示 MobiLink 高速缓存在临时磁盘上的可用磁盘空间。
- **打开的网络连接** 显示当前由服务器打开的 TCP 连接的数量。
- **拒绝的网络连接** 显示服务器每秒拒绝的网络连接总数。

删除旧的监控器度量

您可以自定义监控器保留历史度量的持续时间。可以选择使用部分或全部设置。缺省情况下，监控器会在每天午夜执行一次对其自身的维护操作。维护会影响度量而不影响警告。

♦ **配置对历史度量的删除操作**

1. 单击 [**管理**]。
2. 单击 [**配置**] 选项卡。
3. 单击 [**编辑**]。
4. 单击 [**维护**]。
5. 指定监控器应执行维护操作的时间。缺省情况下，维护操作在午夜执行。该时间是运行监控器的计算机的本地时间。
6. 自定义 [**数据缩减**] 设置：
   - **采用超过以下天数的平均值** 选择该选项时，将取超过指定天数的所有数字度量的平均值，然后删除这些数字度量。非数字度量并不删除。
   - **删除超过以下天数的值** 选择该选项时，将删除超过指定时间长度的所有度量。

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

- **当 SQL Anywhere 监控器使用的总磁盘空间大于以下值 (MB) 时删除旧值: *X*** 选择该选项时，需指定可用于存储度量的最大空间量。当使用的磁盘空间数量达到或超过指定数量时，监控器将从最早的度量开始删除，从而防止监控器为其度量使用更多的磁盘空间。删除度量，直到有足够的空间来存储新度量时为止。

7. 单击 [保存]。

管理资源

资源是指 MobiLink 服务器。向监控器添加资源，然后开始监控它们。

缺省资源名为 **SQL Anywhere Monitor**，可报告监控器本身的健康情况。您不能修改该资源，也不能停止对它的监控。

开始监控资源

在您开始监控资源时，监控器便开始收集度量。

对资源的监控将在：

- 添加资源时自动开始。请参见“添加资源”一节第 206 页。
- 启动监控器时自动开始。缺省情况下，启动监控器时会自动开始对所有现有资源的监控。
- 封锁期结束时自动开始。监控器会自动尝试连接到资源并重新开始监控。
- 当管理员打开 [**管理**] 选项卡时，单击 [**资源**]，从列表中选择一个资源，然后单击 [**启动**]。

添加资源

要监控 MobiLink 服务器，您必须首先将资源添加到监控器。

将 MobiLink 服务器添加为要监控的资源时，服务器不会被以任何方式进行修改。添加资源时，应提供用于连接到 MobiLink 服务器的用户 ID 和口令。用户 ID 和口令是由监控器保存的。

只有管理员才能添加资源。缺省情况下，添加资源后即会开始对资源进行监控。

♦ 将资源添加到监控器

1. 登录到监控器。
2. 单击 [**管理**] 选项卡。
3. 在 [**资源**] 选项卡上，单击 [**添加**]。
4. 按照 [**添加资源**] 窗口中的说明添加资源来监控 MobiLink 服务器。

   添加 MobiLink 服务器时，必须提供该资源的用户 ID 和口令。这些证书用于连接到 MobiLink 服务器。用户 ID 和口令是由监控器保存的。
5. 单击 [**创建**]。

   即会添加资源并启动对该资源的监控。
6. 单击 [**确定**]。

收集间隔

有三种类型的收集间隔：

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

- **高收集间隔** 该等级用于频繁发生变化的信息，如同步完成率。
- **中收集间隔** 该等级用于变化不太频繁的信息，如可用的磁盘空间量。
- **低收集间隔** 该等级用于很少发生变化的信息，如未提交的错误报告。

管理员可以配置监控器收集资源的度量的频率。为每个资源设置收集间隔。不能配置缺省资源（SQL Anywhere 监控器）。

♦ 编辑收集间隔

1. 单击 [**管理**] 选项卡。
2. 单击 [**资源**] 选项卡，然后从列表中选择一个资源。
3. 单击 [**配置**]。
4. 单击 [**收集间隔**]。
5. 根据需要配置其它设置，然后单击 [**保存**]。
6. 单击 [**确定**]。

另请参见

- “监控器度量”一节第 200 页
- “指定要收集的度量”一节第 207 页

指定要收集的度量

管理员可以配置监控器收集哪些度量以及何时应发出警告。不能配置缺省资源（SQL Anywhere 监控器）。

♦ 配置所要收集的度量

1. 单击 [**管理**] 选项卡。
2. 单击 [**资源**] 选项卡，然后从列表中选择一个资源。
3. 单击 [**配置**]。
4. 单击 [**度量**]。选择度量和警告。有关度量和警告的定义，请参见“度量和警告的类型。”一节第 208 页。
5. 根据需要配置其它设置。
6. 单击 [**保存**]。
7. 单击 [**确定**]。

另请参见

- “监控器度量”一节第 200 页
- “收集间隔”一节第 206 页

Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

度量和警告的类型。

下表介绍了 [**配置资源**] 窗口中可用于资源的度量：[**度量**] 选项卡。许多缺省设置是随意的，因为各个同步系统都具有不同的行为和约束，所以缺省设置可能并不适合于您的环境。您应认真考虑每个度量并根据您的需要来设置它们。

- **CPU 使用（高收集间隔）**
  - **当 CPU 使用超过给定阈值达到给定秒数时发出警告** 缺省**阈值**为 100%。缺省**秒数**为 300。
- **内存使用（中收集间隔）**
  - **当使用的高速缓存页的百分比 (%) 大于以下值时发出警告: *X*** 缺省值为 100。
  - **当锁定的高速缓存页的百分比 (%) 大于以下值时发出警告: *X*** 缺省值为 80。
  - **当每秒转入和转出的页数超过给定阈值持续给定秒数时发出警告** 缺省阈值为 256。缺省秒数为 120。
- **网络使用（高收集间隔）** 选择此选项可收集关于服务器中的网络使用情况的度量。可在 [**计算机资源**] 选项卡上查看这些度量。请参见“[监控] 选项卡：[计算机资源] 选项卡”一节第 204 页。
- **同步（高收集间隔）** 选择此选项可收集关于服务器中的同步情况的度量。可在 [**同步**] 选项卡上查看这些度量。请参见“[监控] 选项卡：[同步] 选项卡”一节第 203 页。
  - **当最长活动同步时间多于以下时间（秒）时发出警告: *X*** 缺省值为 600。
  - **当失败的同步数超过给定阈值持续给定分钟数时发出警告** 失败同步数的缺省阈值为 20。缺省分钟数为 60。
- **同步吞吐量（高收集间隔）** 选择此选项可收集关于服务器中的同步吞吐量的度量。可在 [**同步**] 选项卡上查看这些度量。请参见“[监控] 选项卡：[同步] 选项卡”一节第 203 页
- **错误率（高收集间隔）** 选择此选项可收集关于服务器中的错误率的度量。
  - **当错误数超过给定阈值持续给定分钟数时发出警告** 缺省阈值（错误数）为 50。缺省分钟数为 60。
- **警告率（高收集间隔）** 选择此选项可收集关于警告率的度量。
- **使用中的数据库连接（高收集间隔）** 选择此选项可收集关于服务器中正在使用的数据库连接数的度量。可在 [**统一数据库**] 选项卡上查看这些度量。请参见“[监控] 选项卡：[统一数据库] 选项卡”一节第 203 页。
- **MobiLink 高速缓存的可用磁盘空间（中收集间隔）** 选择此选项可收集关于服务器上的 MobiLink 高速缓存的可用磁盘空间的度量。可在 [**计算机资源**] 选项卡上查看这些度量。请参见“[监控] 选项卡：[计算机资源] 选项卡”一节第 204 页。
  - **当 MobiLink 高速缓存的可用磁盘空间小于以下值 (MB) 时发出警告: *X*** 缺省值为 100。
- **数据库工作线程的最长活动等待（高收集间隔）** 选择此选项可收集关于服务器中的数据库工作线程的最长活动等待时间的度量。
  - **当数据库工作线程的最长活动等待大于以下时间（秒）时发出警告: *X*** 缺省值为 300。

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

- **最长活动同步时间（高收集间隔）** 选择此选项可收集关于服务器中的最长活动同步时间的度量。可在 [**同步**] 选项卡上查看这些度量。此度量应配置为高收集间隔。请参见“[监控] 选项卡：[同步] 选项卡”一节第 203 页。
- **当在以下环境中发生相同情况时取消发出警告** 选择此选项以阻止在指定时间内接收重复的警告。缺省值为 30 分钟。

停止监控资源

当您不想让监控器从 MobiLink 服务器收集度量时，可停止监控资源。例如，您想要在您知道资源将会不可用时停止监控；否则，您在资源可用前收到警告。除了缺省监控器资源以外，您可在随时停止监控任何资源。

当您停止监控资源时，监控器将：

- 停止为资源收集度量。
- 停止为资源发出警告。

停止监控资源的方法共有两种：

- **调度有规律的、重复的封锁期限** 当符合以下条件时最好选择此方法：
  - 必须反复地停止监控 MobiLink 服务器。例如，在每个月的月末执行常规维护。
  - 事先知道 MobiLink 服务器不可用的时间。例如，您知道常规维护需要四个小时。
  - 需要监控以自动重启。当封锁结束时，监控器尝试重新连接到资源及继续收集数据。

  要使用此方法，可创建封锁来使监控器在指定的时间停止监控。请参见“使用封锁自动停止监控资源”一节第 210 页。
- **手工停止监控** 当符合以下条件时最好选择此方法：
  - 需要停止监控很少出现的任务或一次性任务。例如，因为正在运行资源的计算机需要离线才能进行特殊维护，所以您需要停止监控。
  - 可以随后重新启动监控。当资源已手工停止时，监控器等待您重新启动监控。

  要使用此方法，请参见“手工停止监控资源”一节第 209 页。

如果要永久停止监控一个资源，可将它从监控器中删除。请参见“删除资源”一节第 210 页。

手工停止监控资源

以下过程介绍如何手工停止资源。有关停止资源时所发生的情况的信息，请参见“停止监控资源”一节第 209 页。

♦ **手工停止资源**

1. 单击 [**管理**] 选项卡。

2. 选择要停止的资源。
3. 在 [**资源**] 选项卡上，单击 [**停止**]。

另请参见

- “开始监控资源”一节第 206 页
- “使用封锁自动停止监控资源”一节第 210 页

使用封锁自动停止监控资源

以下过程介绍如使用封锁停止资源。有关停止资源时所发生的情况以及应该使用封锁的时间的信息，请参见“停止监控资源”一节第 209 页。

封锁是指您不想让监控器收集度量的时间。当封锁结束时，监控器尝试重新连接到资源及继续收集数据。

封锁出现于资源的本地时间。

♦ **配置封锁时间**

1. 以管理员身份登录到监控器。
2. 单击 [**管理**] 选项卡。
3. 在 [**资源**] 选项卡上，选择要为其指定封锁时间的资源。
4. 单击 [**配置**]。
5. 单击 [**封锁**] 选项卡。
6. 单击 [**新建**]。
7. 在 [**新建封锁期**] 窗口中，指定封锁的日期和时间。

   对于资源 MobiLink 服务器所在的计算机，此时间为本地时间。
8. 单击 [**保存**]。
9. 单击 [**保存**]。
10. 单击 [**确定**]。

另请参见

- “开始监控资源”一节第 206 页
- “手工停止监控资源”一节第 209 页

删除资源

仅当您确定不需要监控资源时，您才应将其删除；例如，当您不再使用服务器时。

删除资源会使监控器：

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

- 永久停止监控该资源。
- 丢弃为该资源收集的度量。

只有管理员才能删除资源。不能删除 [**SQL Anywhere 监控器**] 资源。

♦ **删除资源**

1. 单击 [**管理**] 选项卡。
2. 在 [**资源**] 选项卡上，选择一个资源，然后单击 [**删除**]。
3. 单击 [**是**]。

另请参见

- “停止监控资源”一节第 209 页

使用监控器用户

监控器支持三类用户：

- **只读用户** 具有监控器资源的只读权限。只读用户可以在 [**监控**] 选项卡上查看度量，但不能访问 [**管理**] 选项卡。需要用户名和口令。
- **运算符** 具有监控器资源的只读权限并可接收警告。这些用户可以在 [**监控**] 选项卡上查看度量，可以接收电子邮件警告，同时可解析和删除警告。但是，操作员不能访问 [**管理**] 选项卡。需要用户名和口令。
- **管理员** 与操作员具有的权限相同，还可以配置资源和添加用户。管理员还可以访问 [**管理**] 选项卡。缺省用户 [**admin**] 是管理员。需要用户名和口令。

用于登录到监控器的用户名和口令区分大小写。

缺省用户

缺省情况下，当您第一次启动监控器时，它有一个管理员用户，用户名为 **admin**，口令为 **admin**。缺省情况下，此用户具有完全权限。建议您更改缺省的管理员口令，以限制对监控器的访问。请参见“编辑监控器用户”一节第 213 页。

只读访问（无用户名）

缺省情况下，任何用户即使不登录到监控器，也拥有只读权限。但是，出于安全及其它原因，管理员可以要求用户登录。请参见“要求监控器用户登录”一节第 214 页。

创建监控器用户

您必须是管理员才能添加监控器用户。

♦ **添加新的监控器用户**

1. 单击 [**管理**] 选项卡。
2. 单击 [**用户**] 选项卡。
3. 单击 [**新建**]。
4. 填写有关新用户的信息。仅对于应接收来自监控器的电子邮件警告的用户而言，电子邮件地址才为必填项。

   单击 [**保存**]。
5. 如果创建了操作员或管理员，可将该用户与资源相关联。请参见“使监控器用户与资源相关联”一节第 213 页。

另请参见

- “编辑监控器用户”一节 《SQL Anywhere 服务器 - 数据库管理》

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

使监控器用户与资源相关联

如果希望用户接收关于关联资源的电子邮件警告，则必须使用户与资源相关联。只能使操作员或管理员与资源相关联。

♦ 使操作员或管理员与资源相关联

1. 单击 [**管理**] 选项卡。
2. 单击 [**资源**] 选项卡。
3. 选择资源，然后单击 [**配置**]。
4. 单击 [**操作员**]。
5. 在 [**可用操作员**] 列表中，选择用户并单击 [**添加**]。
6. 单击 [**保存**]。
7. 单击 [**确定**]。
8. 检查监控器是否已设置为通过电子邮件发送警告通知。请参见“发送警告电子邮件”一节 第 217 页。

另请参见

- “使用监控器用户”一节第 212 页

编辑监控器用户

作为管理员，您可以编辑监控器用户以更改他们的：

- 口令
- 电子邮件地址
- 语言设置
- 用户类型

♦ 编辑现有监控器用户

1. 单击 [**管理**] 选项卡。
2. 单击 [**用户**] 选项卡。
3. 选择要编辑的用户。
4. 单击 [**编辑**]。
5. 根据需要更改用户的设置。
6. 单击 [**保存**]。

Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

7. 如果正在编辑操作员或管理员，则可将该用户与资源相关联。请参见“使监控器用户与资源相关联”一节第 213 页。

另请参见

- “使用监控器用户”一节第 212 页
- “创建监控器用户”一节第 212 页
- “删除监控器用户”一节第 214 页

删除监控器用户

删除用户时会将用户从监控器中删除并取消用户与任何资源的关联。

您必须是管理员才能删除监控器用户。

♦ **删除现有监控器用户**

1. 单击 [**管理**] 选项卡。
2. 单击 [**用户**] 选项卡。
3. 选择要删除的用户。
4. 单击 [**删除**]。
5. 单击 [**是**] 删除所选用户。单击 [**全部删除**] 删除所有用户。

   用户将从监控器中删除。

另请参见

- “创建监控器用户”一节第 212 页
- “编辑监控器用户”一节第 213 页
- “使监控器用户与资源相关联”一节第 213 页

要求监控器用户登录

缺省情况下，任何用户都具有监控器的只读访问权限。您可以更改此行为，以使用户每次在 Web 浏览器中打开监控器时，都必须提供用户名和口令，然后才能看到任何监控数据。

♦ **限制对监控器的访问**

1. 单击 [**管理**] 选项卡。
2. 在 [**配置**] 选项卡上，单击 [**编辑**]。
3. 单击 [**验证**]。
4. 清除 [**允许所有人以只读方式访问 SQL Anywhere 监控器**] 选项。
5. 单击 [**保存**]。

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

另请参见

- “创建监控器用户”一节第 212 页
- “编辑监控器用户”一节第 213 页

警告

警告是所关注且应引起管理员或操作员注意的某一条件或状态。警告包括有关问题起因的信息，并提供有关如何解决问题的建议。

对于诸如磁盘空间少、重要软件更新、登录尝试失败以及内存利用率高之类的条件，有多种预定义的警告。当满足某个警告条件时，[**监控**] 选项卡上的底部窗格会列出该警告。在顶部窗格中，MobiLink 服务器的 [**状态**] 将发生变化来指示存在警告。您可以对监控器进行配置，使其在出现警告时向操作员和管理员发送电子邮件。请参见“发送警告电子邮件”一节第 217 页。

监控器基于所收集的度量对警告进行检测。而非在受监控的 MobiLink 服务器上检测警告。您可更改缺省阈值，并通过编辑资源来选择启用哪些警告。请参见“监控器度量”一节第 200 页。

查看警告

任何用户都可查看警告；但是，只有操作员和管理员才能解析和删除警告。

♦ 查看警告

1. 单击 [**监控**] 选项卡。
2. 从列表中选择一个资源。
3. 在底部窗格中，单击 [**警告**] 选项卡。
4. 在警告列表中选择一个行。
5. 单击 [**详细信息**]。
6. 单击 [**确定**]。

另请参见

- “解析警告”一节第 216 页
- “删除警告”一节第 217 页
- “发送警告电子邮件”一节第 217 页

解析警告

一旦触发警告的问题得以解决，即可将警告标记为已解析。解析警告会致使监控器更改警告的状态列，但警告仍在警告列表中。如果要移除该警告，必须将它删除。请参见“删除警告”一节第 217 页。

只有操作员和管理员才能解析警告。

♦ 解析警告

1. 单击 [**监控**] 选项卡。
2. 从列表中选择一个资源。

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

3. 在底部窗格中，单击 [**警告**] 选项卡。
4. 在警告列表中选择该行。
5. 单击 [**标记已解决**] 解析所选的警告。单击 [**标记已全部解决**] 解析列表中的所有警告。

   在 [**警告**] 选项卡上 [**状态**] 列中的值更改为 [**已解决**]。

   如果此警告是这个资源唯一的未解析警告，则资源的状态会更改为 [**良好**]。

另请参见

- “删除警告”一节第 217 页
- “解析警告”一节第 216 页
- “发送警告电子邮件”一节第 217 页
- “查看警告”一节第 216 页
- “警告”一节第 216 页

删除警告

在警告列表中监控器只保留最新的 50 个警告。如果不希望某个警告再出现在警告列表中，可将其删除。您可以删除警告，而无论其状态如何。

只有操作员和管理员才可以删除警告。

♦ 删除警告

1. 单击 [**监控**] 选项卡。
2. 从列表中选择一个资源。
3. 在底部窗格中，单击 [**警告**] 选项卡。
4. 在警告列表中选择一个行。
5. 单击 [**删除**]。

   警告即会从警告列表中删除。

另请参见

- “解析警告”一节第 216 页
- “发送警告电子邮件”一节第 217 页
- “查看警告”一节第 216 页
- “警告”一节第 216 页

发送警告电子邮件

您可以对监控器进行配置，使其在出现警告时向操作员和管理员发送电子邮件。

要使监控器通过电子邮件发送警告通知，您必须：

1. 创建具有电子邮件地址的管理员或操作员。请参见“创建监控器用户”一节第 212 页。
2. 使管理员或操作员与某个资源相关联。请参见“使监控器用户与资源相关联”一节第 213 页。
3. 使监控器能够发送电子邮件。请参见“使监控器能够发送警告电子邮件”一节第 218 页。

使监控器能够发送警告电子邮件

作为管理员，您可以配置监控器，使其在出现警告时发送电子邮件。监控器支持用于发送电子邮件的 SMTP 和 MAPI 协议。

♦ 使监控器能够通过电子邮件发送警告通知

1. 单击 [**管理**] 选项卡。
2. 单击 [**配置**] 选项卡。
3. 单击 [**编辑**]。
4. 单击 [**警告通知**]。
5. 选择 [**通过电子邮件发送警告通知**]。
6. 为 [**您要使用哪个协议来通过电子邮件发送警告？**] 字段选择 SMTP 或 MAPI。
7. 根据需要配置其它设置。
   - **MAPI**
     - **用户名**　键入 MAPI 服务器的用户名。
     - **口令**　键入 MAPI 服务器的口令。
   - **SMTP**
     - **服务器**　指定要使用哪个 SMTP 服务器。键入 SMTP 服务器的服务器名或 IP 地址。例如 *SMTP.yourcompany.com*。
     - **端口**　指定连接至 SMTP 服务器的端口编号。缺省值为 25。
     - **发送者姓名**　为发送者的电子邮件地址指定别名。例如 *JoeSmith*。
     - **发送者地址**　指定发送者的电子邮件地址。例如 *jsmith@emailaddress.com*。
     - **此 SMTP 服务器要求验证**　如果 SMTP 服务器需要验证，则选择此选项。
       - **用户名**　指定要提供给需要验证的 SMTP 服务器的用户名。
       - **口令**　指定要提供给需要验证的 SMTP 服务器的口令。
8. 测试您是否已正确配置了电子邮件通知。

   单击 [**发送测试电子邮件**]。
9. 在系统提示时，输入要将测试电子邮件发送到的电子邮件地址，然后单击 [**确定**]。

   测试电子邮件将会被发送到指定的电子邮件地址。
10. 单击 [**保存**]。

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

另请参见

- “解析警告”一节第 216 页
- “删除警告”一节第 217 页
- “查看警告”一节第 216 页

取消对资源的未提交错误报告的警告

作为管理员，您可以配置在资源有未提交的错误报告时监控器是否发出警告。缺省情况下，监控器并不发送这些警告。有关错误报告及其提交方式的信息，请参见“SQL Anywhere 中的错误报告”一节 《SQL Anywhere 服务器 - 数据库管理》。

♦ **取消对未提交的错误报告的警告**

1. 单击 [**管理**] 选项卡
2. 单击 [**配置**] 选项卡。
3. 单击 [**编辑**]。
4. 单击 [**选项**]。
5. 单击 [**保存**]。

在一台单独的计算机上安装 SQL Anywhere 监控器

以下说明将介绍如何在未运行 SQL Anywhere 的另外一台单独计算机上安装 SQL Anywhere 监控器。

在单独的计算机上运行 SQL Anywhere 监控器具有以下优势：

- 监控器可以在后台作为服务来运行。
- 计算机启动时，监控器将自动启动。
- 如果监控器安装在单独的计算机上，SQL Anywhere 的升级和更新不会覆盖监控器。当该单独的计算机处于生产环境中时，这一点很重要。

♦ 在单独的计算机上安装监控器

- 运行安装介质的 *Monitor* 目录下的 *setup.exe* 文件，并遵照提供的说明进行安装。

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

监控器疑难解答

问题	建议
在您按 F5 键刷新 Web 浏览器窗口时，您需要登录到监控器。	在 Web 浏览器中启用 JavaScript。
在尝试连接到监控器时收到网络通信错误。	启动监控器。请参见“启动监控器”一节 第 195 页。
在升级到 Adobe Flash Player 的最新版本后，继续接收说明以便可以持续升级 Adobe Flash Player。	验证您的操作系统是否支持已安装的 Adobe Flash Player 版本。监控器向后兼容 Adobe Flash Player 版本 9。要确定正确的版本，请访问：http://www.adobe.com/products/flashplayer/systemreqs/。
监控器无法开始监控 SQL Anywhere 数据库资源。	检查该资源的口令验证功能和登录过程是否允许用户 sa_monitor_user 连接到该资源。
未收到任何警告电子邮件。	检查监控器是否已被正确配置为发送电子邮件及发送测试电子邮件。请参见“使监控器能够发送警告电子邮件”一节第 218 页。 检查来自监控器的警告电子邮件是否未被病毒扫描程序拦截。请参见“xp_startsmtp 系统过程”一节 《SQL Anywhere 服务器 - SQL 参考》。
监控器所报告的未调度的请求数似乎少于实际的未调度请求数。	当收集有关未调度请求数的度量时，监控器执行了关于资源的查询。此查询可能是未调度的请求。 在未调度的查询出现时依序对其进行处理。因此，当监控器尝试执行其查询时，如果存在未调度的请求，则此查询必须等待现有未调度的请求完成后，才能够加以执行。 因此，当监控器收集未调度的请求数时，此数量并不包括监控器发出其查询的时间与请求被执行的时间之间所存在的未调度的请求。
当数据库磁盘空间超过指定阈值时，您没有收到警告。	在监控器的收集间隔之间，数据库可以超过指定的磁盘空间警告阈值和可用的空间大小。在这种情况下，数据库将会在监控器能够收集磁盘使用情况度量并发出警告之前停止响应。 如果数据库快速增大，请将磁盘空间警告阈值设置为较大的数值，以便您能够在数据库运行空间不足之前收到警告。请参见“度量和警告的类型。”一节第 208 页。

问题	建议

在“非英文”计算机的 Firefox Web 浏览器中打开监控器时，监控器将以英文显示。	Firefox 无法正确使用您的计算机上的首选区域设置。您可以使用 Internet Explorer，或在 Firefox 中尝试以下的变通解决方法：
	1. 在 Firefox 中打开一个新选项卡。
	2. 在地址栏中，键入以下内容：
	`about:config`
	按 **Enter** 键。
	出现提示时，单击 [**I'll Be careful, I Promise!**]
	3. 在 [**Filter**] 字段中，键入以下内容：
	`general.useragent.locale`
	4. 在首选项列表中，双击 [**general.useragent.locale**]。
	5. 在 [**Enter String Value**] 窗口中，输入您的区域设置。例如，键入 fr-FR 表示法语；键入 de-DE 表示德语；键入 zh-CN 表示简体中文；键入 ja-JP 表示日语。
	6. 单击 [**OK**]。

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

中继服务器

目录

中继服务器简介 224
中继服务器配置文件 227
出站启动器 231
中继服务器状态管理器 234
部署中继服务器 236
更新中继服务器群配置 241
Sybase 中继服务器托管服务 243
将 MobiLink 与中继服务器结合使用 245

中继服务器简介

中继服务器使移动设备和 MobiLink、Afaria 与 iAnywhere Mobile Office 服务器之间通过 Web 服务器进行的通信变得安全且负载平衡。中继服务器可提供以下内容：

- 移动设备与 MobiLink、Afaria 和 iAnywhere Mobile Office 服务器进行通信的公共通信体系结构。
- 为 MobiLink、Afaria 和 iAnywhere Mobile Office 服务器启用负载平衡和容错环境的机制。
- 以易于与现有公司防火墙配置和策略集成的方式，有助于移动设备与 MobiLink、Afaria 和 iAnywhere Mobile Office 服务器之间的通信的方法。

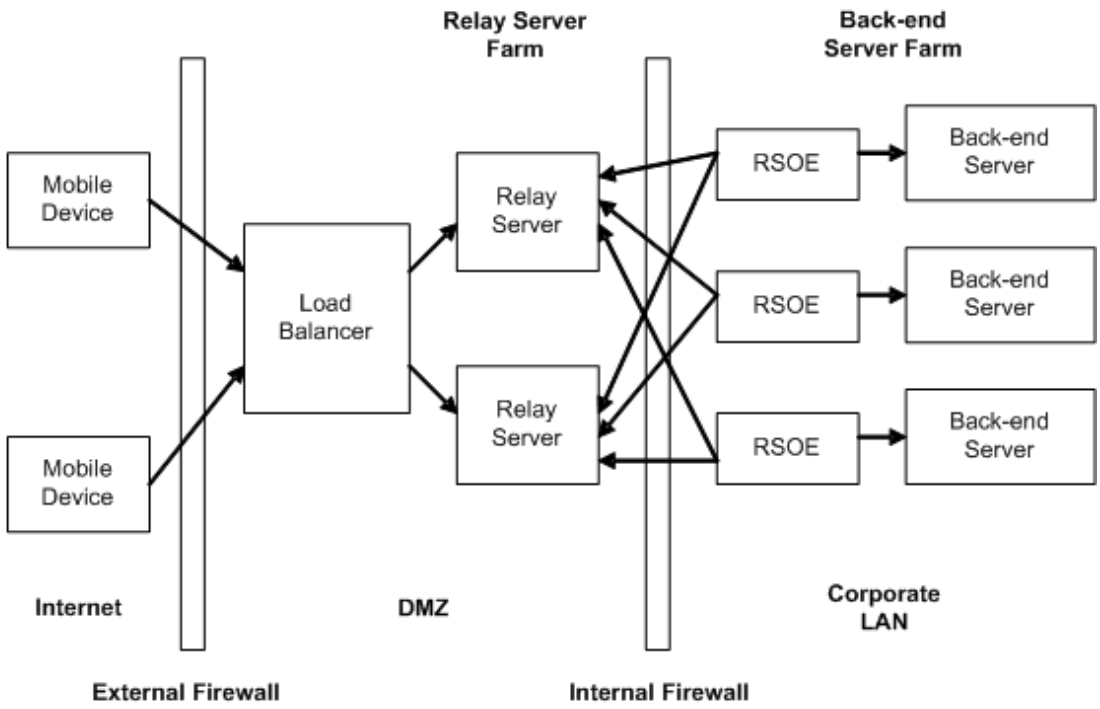
中继服务器体系结构

中继服务器部署包括以下内容：

- 运行客户端应用程序的移动设备和在公司 LAN 中运行的需要与后端服务器进行通信的服务。
- 用于将移动设备的请求定向到一组中继服务器的可选负载平衡器。
- 在公司 DMZ 中运行的一个或多个中继服务器。
- 负责为客户端请求提供服务并在公司 LAN 中运行的后端服务器。
- 每个后端服务器拥有一个中继服务器出站启动器（Relay Server Outbound Enabler，简称 RSOE）。该出站启动器管理后端服务器和中继服务器群之间的所有通信。

下图显示了中继服务器的体系结构。

Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1



中继服务器包含一组 web 扩展、一个用于维护状态信息的后台进程和一台 web 服务器。

因为中继服务器是在 web 服务器中运行的 web 扩展，所以所有通信都使用 HTTP 或 HTTPS 来执行。使用 HTTP 易于与现有的公司防火墙配置和策略集成。中继服务器要求公司 LAN 与中继服务器之间的连接从公司 LAN 内部启动。这可以提供更安全的部署环境，因为不需要从 DMZ 到公司 LAN 的入站连接。

中继服务器包含两种 web 扩展：客户端扩展和服务器扩展。客户端扩展处理由在移动设备上运行的应用程序发出的客户端请求。服务器扩展处理由代表后端服务器的出站启动器所发出的请求。

中继服务器群

中继服务器群是使用前端负载平衡器的任意数量的中继服务器。可以用一个单独的中继服务器建立一个中继服务器群，这时不需要负载平衡器。在这种情况下，移动设备可以直接连接到中继服务器。

后端服务器群

后端服务器群是一组同质后端服务器。通过中继服务器群进行请求的客户端必须指定它要指向的后端服务器群。

负载平衡器

负载平衡器将来自移动设备的请求定向到一台在中继服务器群中运行的中继服务器。如果只有一台中继服务器，则不需要负载平衡器。

中继服务器出站启动器

中继服务器出站启动器和后端服务器在同一台计算机上运行。其主要功能是代表后端服务器启动到中继服务器群中的所有中继服务器的出站连接。每个后端服务器拥有一个出站启动器。请参见“出站启动器”一节第 231 页。

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

中继服务器配置文件

中继服务器配置文件用于定义中继服务器群和与中继服务器群相连的后端服务器群。中继服务器配置文件分为以下几部分：

- “中继服务器部分”一节第 227 页
- “后端群部分”一节第 228 页
- “后端服务器部分”一节第 228 页
- “选项部分”一节第 229 页

各部分以部分标记开头。部分标记采用将标识部分名的关键字用方括号括起来的形式。例如，[relay_server] 表示中继服务器部分的开头。

部分标记后面的几行内容定义与正在被定义的部分相关的各种属性。定义属性的方法是在等号左侧指定属性名称，在等号右侧指定属性值。例如，`property name = value`。所有部分名称和属性名称都不区分大小写。注释以行首的井号 (#) 字符标记。

配置文件应只包含 7-字节 ASCII 字符。可按任何顺序来指定这些部分。

中继服务器部分

中继服务器部分用于定义单个中继服务器，所以群中每个中继服务器都必须具有中继服务器部分。此部分以 relay_server 关键字来标识。

中继服务器部分属性

可在中继服务器部分中指定以下属性：

- **enable** 指定此中继服务器是否包括在中继服务器群中。可能的值为：
  - **Yes** 指示此中继服务器将包括在中继服务器群中。
  - **No** 指示此中继服务器将不包括在中继服务器群中。

  缺省值是 Yes。此属性是可选的。
- **host** 出站启动器直接连接到中继服务器所应该使用的主机名或 IP 地址。
- **http_port** 出站启动器直接连接到中继服务器所应该使用的 HTTP 端口。值 **0** 或 **off** 禁用 HTTP 连接。缺省情况下，启用此属性并将其设置为 80。
  - **0 或 off** 禁用来自出站启动器的 HTTP 访问。
  - **1 到 65535** 在指定端口启用 HTTP。
- **https_port** 出站启动器直接连接到中继服务器所应该使用的 HTTPS 端口。值 **0** 或 **off** 禁用 HTTPS 连接。缺省情况下，启用此属性并将其设置为 443。
  - **0 或 off** 禁用来自出站启动器的 HTTPS 访问。
  - **1 到 65535** 在指定端口启用 HTTPS。
- **description** 输入最多 2048 个字符的自定义说明。此属性是可选的。

Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

后端群部分

后端群部分指定后端服务器群的属性。后端服务器群是一组同质后端服务器。通过中继服务器群进行请求的客户端必须指定它要指向的后端服务器群。每个后端服务器群都有一个后端群部分。

此部分以 backend_farm 关键字来标识。

后端群部分属性

可在后端群部分中指定以下属性：

- **enable** 指定是否允许来自此后端服务器群的连接。可能的值为：
  - **是** 允许来自此后端服务器群的连接。
  - **否** 禁止来自此后端服务器群的连接。

  缺省值是 Yes。此属性是可选的。
- **id** 指派给后端服务器群的名称，最多包含 2048 个字符。
- **client_security** 指定后端服务器群要求其客户端提供的安全级别。可能的值为：
  - **on** 表示客户端必须使用 HTTPS 进行连接。
  - **off** 表示客户端必须使用 HTTP 进行连接。

  此属性是可选的。如果没有指定任何值，则客户端可以使用 HTTP 或 HTTPS 进行连接。
- **backend_security** 指定后端服务器群中的出站启动器为连接到中继服务器群所需的安全级别。这些可能的值为：
  - **on** 表示后端服务器群的所有连接必须使用 HTTPS。
  - **off** 表示后端服务器群的所有连接必须使用 HTTP。

  此属性是可选的。如果没有指定任何值，则可以使用 HTTP 或 HTTPS 进行连接。
- **description** 输入最多 2048 个字符的自定义说明。此属性是可选的。

后端服务器部分

后端服务器部分定义后端服务器连接。该部分指定出站启动器代表后端服务器连接到中继服务器群时所使用的信息。连接到中继服务器群的每个出站启动器都有一个后端服务器部分。后端服务器部分还将后端服务器指派给后端服务器群。

此部分以 backend_server 关键字来标识。

后端服务器部分属性

可在后端服务器部分中指定以下属性：

- **enable** 指定是否允许来自此后端服务器的连接。可能的值为：
  - **是** 允许来自此后端服务器的连接。

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

- **否** 禁止来自此后端服务器的连接。

缺省值是 Yes。此属性是可选的。

- **id** 指派给后端服务器连接的名称，最多包含 2048 个字符。
- **farm** 此后端服务器所属的后端服务器群的名称。
- **MAC** 出站启动器与中继服务器进行通信所使用的网络适配器的 MAC 地址。该地址以 IEEE 802 MAC-48 格式来指定。要获得正确格式的 MAC 地址，则在中继服务器出站启动器控制台或日志中查找。此属性是可选的。如果未指定此属性，则不会发生 MAC 地址检查。
- **token** 中继服务器验证后端服务器连接所使用的安全标识，最多可包含 2048 个字符。此属性是可选的。
- **description** 输入最多 2048 个字符的自定义说明。此属性是可选的。

选项部分

选项部分用于指定适用于群中各个中继服务器的属性。只允许有一个选项部分。

此部分以 options 关键字来标识。

选项部分属性

可在选项部分中指定以下属性：

- **start** 用于启动状态管理器的方法。这些可能的值为：
  - **auto** 状态管理器使用状态管理器命令行缺省值自动启动。
  - **no** 状态管理器作为 Windows 服务从外部启动。
  - **full path** 指定状态管理器可执行文件 (*rshost*) 的完整路径。

  缺省值为 auto。此属性是可选的。
- **shared_mem** 指定中继服务器用于状态跟踪的最大共享内存量。缺省值是 10 MB。此属性是可选的。
- **verbosity** 可以将 verbosity 设置为以下级别：
  - **0** 仅记录错误。此记录级别用于部署。这是缺省设置。
  - **1** 请求级记录。所有 HTTP 请求都被写入日志文件。

  无论指定哪个记录级别都显示错误，而只有记录级别大于 0 时才显示警告。

中继服务器配置文件格式

以下是中继服务器配置文件的基本格式：

```
#
# Options
#
```

```
[options]
# List of Relay Server properties that apply to all Relay Servers
option = value

#
# Define a Relay Server section, one for each
# Relay Server in the Relay Server farm
#
[relay_server]
# List of properties for the Relay Server
property = value

#
# Define a backend server farm section, one for each backend
# server farm
#
[backend_farm]
# List of properties for a backend server farm
property = value

#
# Define a backend server section, one for each
# Outbound Enabler connecting to the Relay Server farm
#
[backend_server]
# List of properties for the backend server connection
property = value
```

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

出站启动器

出站启动器和后端服务器在同一台计算机上运行。其用途是：

- 打开在公司 LAN 中运行的计算机到在 DMZ 中运行的中继服务器群的出站连接。
- 将从中继服务器接收的客户端请求转发到后端服务器，并通过中继服务器将后端服务器响应转发回客户端。

出站启动器启动后，它发送 HTTP 请求以检索群中正在运行的中继服务器的列表。使用映射到中继服务器的 web 服务器扩展组件的服务器 URL 来完成此操作。服务器 URL 可以直接映射到中继服务器，也可以映射到负载平衡器。如果服务器 URL 映射到负载平衡器，则该负载平衡器将请求转发到群中正在运行的中继服务器之一。接收到出站启动器请求的中继服务器返回群中所有中继服务器的连接信息。然后，出站启动器创建两个与返回的各中继服务器的出站连接（称为信道）。一个信道（称为上行信道）使用具有实质上无限响应的 HTTP 请求来创建。该响应是从中继服务器到出站启动器的客户端请求的连续流。另一个信道（称为下行信道）使用具有实质上无限内容长度的 HTTP 请求来创建。该请求由客户端请求的服务器响应的连续流形成。

出站启动器在接收到来自它已连接的中继服务器之一的上行信道上的客户端请求时，会将请求转发到它正在服务的后端服务器。接收到来自后端服务器的响应后，出站启动器会使用下行信道将响应转发到从中接收相应请求的中继服务器。

出站启动器语法

rsoe [*option*]+

rsoe @{ *filename* | *environment-variable* } ...

参数

选项　以下选项可与出站启动器一起使用。这些选项都是可选的。

rsoe 选项	说明
@data	读取来自指定的环境变量或配置文件的选项。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)”一节第 232 页。
-f *farm*	后端服务器所属的群的名称。
-id *id*	分配给后端服务器的名称。
-cs *"connection-string"*	用于连接到后端服务器的主机和端口。缺省值为 `"host=localhost;port=80"`.

rsoe 选项	说明
-cr *"connection-string"*	中继服务器连接字符串。中继服务器连接字符串的格式是以分号分隔的 "名称 - 值" 对的列表。"名称 - 值" 对包含以下内容： ● **host** 中继服务器的 IP 地址或主机名。缺省值为 localhost。 ● **port** 中继服务器正在监听的端口。这是必需项。 ● **url_suffix** 中继服务器的服务器扩展的 URL 路径。 对于 Windows，缺省值为 */ias_relay_server/server/rs_server.dll*。 对于 Linux，缺省值为 */srv/iarelayserver*。 ● **https** 0 - HTTP（缺省值） 1 - HTTPS 对于 **https=1**，还可以指定以下选项： ● **tls_type** RSA ● **certificate_name** 证书的公用名字段。 ● **certificate_company** 证书的组织名字段。 ● **certificate_unit** 证书的组织单位字段。 ● **trusted_certificates** 包含受信任根证书列表的文件。
-t *token*	要传递给中继服务器的安全标识。
-v *level*	设置用于记录的详细程度级别。*level* 可以设为 **0**、**1** 或 **2**： ● **0** 仅记录错误。此记录级别用于部署。 ● **1** 会话级记录。这是同步会话的更高级视图。 ● **2** 请求级记录。提供同步会话中 HTTP 请求的更详细视图。
-d *seconds*	中继服务器连接重试间隔。缺省值是 5 秒。
-s	停止出站启动器。

文件隐藏实用程序 (dbfhide)

文件隐藏实用程序 (dbfhide) 使用简单加密来对配置文件和初始化文件的内容进行模糊处理。

语法

dbfhide *original-configuration-file encrypted-configuration-file*

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

选项	说明
original-configuration-file	指定原始文件的名称。
encrypted-configuration-file	指定模糊处理后新文件的名称。

中继服务器和出站启动器检测配置文件是否已被用 dbfhide 进行了模糊处理，并对该文件进行了相应的处理。

此实用程序不接受用于从配置文件中读入选项的 @data 参数。

部署注意事项

使用出站启动器时应注意以下事项：

- **出站启动器作为 Windows 服务** 还可以使用服务实用程序将出站启动器作为 Windows 服务来设置和维护。请参见“中继服务器状态管理器”一节第 234 页。
- **验证** 不能使用简单或摘要式验证。无论 web 服务器类型或操作系统为何，*rsoe.exe* 都不支持使用 web 服务器进行的简单或摘要式验证。

中继服务器状态管理器

中继服务器状态管理器是一个进程，负责维护客户端请求和出站启动器会话间的中继服务器状态信息。状态管理器还负责管理中继服务器所使用的日志文件。状态管理器可以由中继服务器来自动启动，或作为 Windows 服务启动（仅在 Windows 上）。

日志文件的缺省名是 *ias_relay_server_host.log*。在 Windows 上，该文件位于由 TEMP 环境变量所指定的目录中。在 Linux 上，该文件位于由 TMP、TEMP 或 TMPDIR 环境变量所指定的目录中。如果这些变量都未设置，则在根目录中创建一个日志文件。

注意
在所有情况下，Apache 用户进程都必须对您所选择的 *tmp* 目录位置具有写权限。

正常关闭时，状态管理器以 <*yymmdd*><*nn*>*.log* 形式重命名该日志文件，其中，<*yymmdd*> 代表重命名日志文件的日期，<*nn*> 代表日志文件在该日的顺序版本号。

建议将状态管理器作为 Windows 服务予以启动。请注意，不支持在命令行上手动启动状态管理器。

可以指定中继服务器所使用的选项来启动状态管理器。要更改这些选项，请在中继服务器配置文件的选项部分中设置 **start** 属性。例如：

```
[options]
start = "rshost -o c:\temp\myrshost.log"
```

请注意，必须在选项前指定中继服务器状态管理器可执行文件 (`rshost`) 的名称。

将中继服务器状态管理器作为 Windows 服务进行启动

仅对于 Windows，可以通过使用服务实用程序 *dbsvc.exe* 将状态管理器作为 Windows 服务来启动。中继服务器配置文件的选项部分中的 start 属性应设置为 **no**。请参见“选项部分”一节第 229 页。

服务实用程序 (dbsvc) 用于创建、修改和删除服务。要获得使用信息的完整列表，请运行不带任何选项的 *dbsvc.exe*。

要设置名为 rs 的自动启动的状态管理器服务：

```
dbsvc -as -s auto -w rs "C:\inetpub\wwwroot\ias_relay_server\server
\rshost.exe" -q -qc -f c:\inetpub\wwwroot\ias_relay_server\server
\rstest.config -o c:\temp\rs.log
```

要启动服务：

```
dbsvc.exe -u rs
```

要停止服务：

```
dbsvc.exe -x rs
```

要卸载服务：

```
dbsvc.exe -d rs
```

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

自动启动中继服务器状态管理器

当第一个出站启动器连接到中继服务器时，状态管理器进程自动启动。这是当中继服务器配置文件的选项部分中的 start 属性未指定或显式指定为 auto 时的缺省行为。日志文件的缺省位置是 *%temp%\ias_relay_server_host.log*。请参见“选项部分”一节第 229 页。

通过自定义选项自动启动中继服务器状态管理器

当需要自动启动，但您想替换一些缺省行为（如详细程度级别或日志文件位置）时，可以使用中继服务器配置文件的选项部分中的 start 属性来显式指定状态管理器命令行。此种情况下不能使用 -f 选项，且配置文件必须命名为 *rs.config* 并与服务器扩展位于同一目录中。请参见“中继服务器状态管理器命令行语法”一节第 235 页。

注意
不要在 wwwroot 目录下指定日志文件位置。IIS 不允许工作进程在已发布树下创建文件。

中继服务器状态管理器命令行语法

rshost [*option*]+

参数

选项 以下选项可用于配置状态管理器。这些选项都是可选的。

rshost 选项	说明
-f *filename*	中继服务器配置文件的文件名。
-o *filename*	用于记录的文件名。
-oq	防止在出现启动错误时弹出窗口。
-q	以最小化窗口运行。
-qc	完成时关闭窗口。
-u	更新正在运行的中继服务器的配置。
-ua	将日志文件存档到 <yymmdd><nn>.log 并截断。

部署中继服务器

以下是有关如何将中继服务器部署到 Windows 上的 IIS 的概述：

1. 部署中继服务器组件。请参见“将中继服务器组件部署到 Windows 上的 IIS”一节第 236 页。
2. 部署 web 服务器扩展和状态管理器。请参见“部署 web 服务器扩展和状态管理器”一节第 237 页。
   a. 创建应用程序池。请参见“创建应用程序池”一节第 237 页。
   b. 启用中继服务器 web 扩展并部署中继服务器配置文件。请参见“部署 web 服务器扩展和状态管理器”一节第 237 页。
3. 在必要时部署中继服务器配置更新。请参见“为 Windows 上的 IIS 更新中继服务器配置”一节第 241 页。

以下是有关如何将中继服务器部署到 Linux 上的 Apache 的概述：

1. 部署中继服务器组件。请参见“将中继服务器组件部署到 Linux 上的 Apache”一节第 239 页。
2. 部署 web 扩展文件和状态管理器。请参见“部署 web 扩展文件和状态管理器”一节第 239 页。
3. 在必要时部署中继服务器配置更新。请参见“为 Linux 上的 Apache 更新中继服务器配置”一节第 242 页。

将中继服务器组件部署到 Windows 上的 IIS

Windows 上的中继服务器包含以下可执行文件：

- *rs_client.dll*
- *rs_server.dll*
- *rshost.exe*
- *dblgen11.dll*
- *dbsvc.exe*
- *dbfhide.exe*
- *dbicu11.dll*
- *dbicudt11.dll*
- *dbsupport.exe*
- *dbghelp.dll*

另请参见

- “中继服务器状态管理器”一节第 234 页
- “文件隐藏实用程序 (dbfhide)”一节第 232 页

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

部署 web 服务器扩展和状态管理器

♦ **部署中继服务器文件**

1. 在用于中继服务器的 web 站点主目录下创建以下目录：
   - *ias_relay_server*
   - *ias_relay_server\client*
   - *ias_relay_server\server*
2. 将 *rs_client.dll* 复制到 *ias_relay_server\client* 目录。
3. 创建中继服务器配置文件 *rs.config*。请参见“中继服务器配置文件”一节第 227 页。
4. 将 *rs_server.dll*、*rshost.exe* 和 *rs.config* 复制到 *ias_relay_server\server* 目录。
5. 确保将 web 站点主目录的连接超时属性设置为 60 秒或更长时间。

创建应用程序池

必须为 *rs_server.dll* 和 *rs_client.dll* web 服务器扩展创建专用应用程序池。在中继服务器使用长时间运行的工作进程时，所有工作循环选项都需要关闭。

♦ **创建应用程序池**

1. 启动 IIS 管理器控制台。
2. 右键单击 [**Application Pools**] 并创建一个新应用程序池，如 RS_POOL。
3. 编辑您所创建的应用程序池的属性：
   a. 选择 [**Recycling**] 选项卡并关闭所有循环选项。
   b. 选择 [**Performance**] 选项卡并执行以下操作：
      i. 关闭 [**Shutdown Worker Processes After Being Idle**]。
      ii. 将工作进程数设置为处理核心的总数。可以根据您的使用和性能首选项来进一步调整该数值。有关详细信息，请参见关于 Web 园大小的 IIS 性能说明。

启用中继服务器 web 扩展

以下步骤介绍了启用中继服务器 web 扩展的过程。

♦ **编辑 ias_relay_server 的属性并启用中继服务器 web 扩展**

1. 选择 [**目录**] 选项卡并执行以下操作：
   a. 将执行权限设置为 [**脚本和可执行文件**]。
   b. 单击 [**应用程序设置**] 下的 [**创建**]。在“创建应用程序池”一节第 237 页中选择您所创建的应用程序池作为关联的应用程序池。

2. 选择 [**目录安全**] 选项卡并执行以下操作：
   a. 在 [**验证和访问控制**] 中单击 [**编辑**]。
   b. 启用匿名访问并填写属于管理员组的帐户的用户名和口令。

      或者，可以将设置保留为内置用户 **IUSR_%computername%**，并执行以下命令来授予访问 IIS metabase 的权限。

      ```
      C:\Windows\Microsoft.Net\Framework\<Version>\aspnet_regiis.exe -ga
      IUSR_%computername%
      ```
3. 在 IIS 管理器中的 [**Web Server Extensions**] 下，允许 *rs_server.dll* 和 *rs_client.dll* 作为 ISAPI 运行。
4. 部署中继服务器配置文件，方法是：创建一个中继服务器配置文件，并将其复制到 *ias_relay_server\server* 目录。

另请参见

- “中继服务器配置文件”一节第 227 页

性能提示

将中继服务器部署到 Windows 上的 IIS 时，请牢记以下内容：

- 中继服务器 web 扩展不依赖于 ASP.NET。删除 ASP.NET ISAPI 过滤器可以产生更佳的性能。缺省情况下，过滤器在标准 IIS 安装中开启。要关闭过滤器，请执行以下操作：
  1. 启动 IIS 管理器控制台。
  2. 编辑 [**缺省 Web 站点**] 的属性。
  3. 在 [**ISAPI 过滤器**] 选项卡下删除 ASP.NET 过滤器。
- 为获得更佳性能，可以关闭 IIS 访问日志。要关闭访问日志，请执行以下操作：
  1. 启动 IIS 管理器控制台。
  2. 在 [**缺省 Web 站点**] 下编辑 *ias_relay_server* 目录的属性。
  3. 在 [**目录**] 选项卡下清除 [**日志访问**] 选项。
- 在生产环境中，可以通过中继服务器配置文件将中继服务器详细程度设置为 0。这在高负载下会产生更佳的性能。
- 中继服务器不会强制限制 Web 园大小。一个工作进程可以为来自所有出站启动器和所有客户端的请求提供服务。但是，可在该进程中创建的线程数将会受到为创建线程而留出的进程堆空间的限制。由 IIS 创建的线程的堆栈大小为 256k。如果您的计算机具有足够的资源，并且您在服务器装载了数千个并发请求时怀疑自己达到并发限制，则可用数目更多的进程进行实验。

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

将中继服务器组件部署到 Linux 上的 Apache

Linux 上的中继服务器包含以下可执行文件：

- *mod_rs_ap_client.so*
- *mod_rs_ap_server.so*
- *rshost*
- *dblgen11.res*
- *libdbtasks11_r.so*
- *libdbicudt11.so*
- *libdbicu11_r.so*
- *libdblib11_r.so*
- *dbsupport*
- *dbfhide*

另请参见

- “中继服务器状态管理器”一节第 234 页
- “文件隐藏实用程序 (dbfhide)”一节第 232 页

部署 web 扩展文件和状态管理器

♦ **部署中继服务器文件**

1. 将以上文件复制到 Apache 安装 *modules* 目录中。
2. 创建中继服务器配置文件 *rs.config*。请参见“中继服务器配置文件”一节第 227 页。
3. 将 *rs.config* 复制到 *modules* 目录中。服务器模块预期 *rshost* 可执行文件位于在其中复制 *rs.config* 文件的目录中。
4. 设置 PATH 和 LD_LIBRARY_PATH 环境变量以包含 Apache *modules* 目录。
5. 编辑 Apache *conf/httpd.conf* 文件。
   a. 添加以下行来装载中继服务器客户端和服务器模块：

```
LoadModule iarelayserver_client_module modules/mod_rs_ap_client.so

LoadModule iarelayserver_server_module modules/mod_rs_ap_server.so
```

> 客户端和服务器模块可使用不同的 URL 来调用。客户端模块在 URL 路径中显式查找字符串 *iarelayserver*。那部分 URL 不需要更改。

   b. 添加以下行为客户端模块创建 <*location*> 部分：

```
<LocationMatch /cli/iarelayserver/* >
    SetHandler iarelayserver-client-handler
</LocationMatch>
```

   c. 添加以下行为服务器模块创建 <*location*> 部分：

```
<Location /srv/iarelayserver/* >
    SetHandler iarelayserver-server-handler
    RSConfigFile "/<apache-install>/modules/rs.config"
</Location>
```

必须指定 RSConfigFile 指令，它指定了中继服务器配置文件 *rs.config* 的位置。*rs.config* 文件必须驻留在部署 rshost 可执行文件的目录中。

d. 设置 TimeOut 指令时，确保将其设置为至少 60 秒。

6. 在 Linux 上，如果在 Apache 生成进程时，以下环境变量中的任何一个变量已设置为全局变量，则不需要进一步配置 Apache：$TMP、$TMPDIR 或 $TEMP。

   如果以上环境变量都未设置为全局变量，或者如果您希望缺省中继服务器日志文件位于特定的临时目录（例如，当状态管理器自动启动但未进行自定义设置时）中，则编辑要设置的文件 */<apache-dir>/bin/envvars*，然后导出 TMP。

   例如，要编辑 envvars 文件中的 $TMP，请执行以下操作：

   ```
   set TMP="/tmp"
   export TMP
   ```

   这会在 shell 中设置 Apache 在生成其进程以前所创建的环境变量。

注意
在所有情况下，Apache 用户进程都必须对您所选择的 *tmp* 目录位置具有写权限。

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

更新中继服务器群配置

中继服务器群配置由中继服务器配置文件的内容进行定义。中继服务器群中的各个中继服务器共享同一中继服务器配置文件，所以当您更新中继服务器群配置时，必须更新群中每个中继服务器的中继服务器配置文件。更新包括以下任何操作：

- 向中继服务器群中添加新的中继服务器。
- 创建新的后端服务器群并允许其访问中继服务器群。
- 向现有后端服务器群添加新的后端服务器。
- 更改中继服务器、后端服务器群或后端服务器的属性。
- 更改选项。

更新中继服务器配置的一种方法是：关闭所有中继服务器，用已更新版本替换中继服务器配置文件，然后重新启动所有中继服务器。但是，关闭和重新启动中继服务器将意味着中继服务器的用户可能会遭遇服务中断问题。

更新中继服务器配置的首选方法是：在中继服务器群正在运行且不中断服务时，使用中继服务器状态管理器来更新中继服务器配置。

可通过使用以下命令行格式启动中继服务器状态管理器的新实例，来完成中继服务器配置的更新：

```
rshost -u -f <filename>
```

-u 选项指示中继服务器状态管理器执行更新操作。-f 选项指定包含已更新配置的配置文件的名称。请参见“中继服务器状态管理器”一节第 234 页。

下文概述了更新中继服务器群配置所需的步骤：

1. 对中继服务器配置文件的主副本进行更改。
2. 在每一台运行属于正在被更新的中继服务器群的中继服务器的实例的计算机上，请执行以下操作：
   a. 用已更新的配置文件替换旧配置文件。
   b. 使用已更新的配置文件运行中继服务器状态管理器。

为 Windows 上的 IIS 更新中继服务器配置

♦ **为 Windows 上的 IIS 更新中继服务器配置**

1. 针对属于您正在更新的中继服务器群的每台计算机，将更新的配置文件复制到中继服务器 web 站点主目录下的 *ias_relay_server\server* 目录中。如果使用自动启动，则必须将配置文件命名为 *rs.config*。
2. 从 *ias_relay_server\server* 目录中，运行以下命令行来应用配置更新：

   ```
   rshost -u -f rs.config
   ```
3. 针对正在被更新的中继服务器群中的每台计算机重复上一步骤。

为 Linux 上的 Apache 更新中继服务器配置

♦ **为 Linux 上的 Apache 更新中继服务器配置**

1. 将更新的配置文件复制到 Apache 安装目录下的 */modules* 目录中。如果使用自动启动，则必须将配置文件命名为 *rs.config*。
2. 从 */<Apache-install>/modules* 目录中，运行以下命令行来应用配置更新：

   ```
   rshost -u -f rs.config
   ```
3. 针对正在被更新的中继服务器群中的每台计算机重复上一步骤。

Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

Sybase 中继服务器托管服务

Sybase 中继服务器托管服务是一个由 Sybase 托管的中继服务器群。该服务旨在便于开发使用 MobiLink 数据同步的移动应用程序，以及简化开发人员的评估过程，特别是在使用公共无线网络发送数据的情况下。您不需要让 IT 部门安装任何东西或在贵公司的防火墙中开启任何漏洞。MobiLink 和托管服务之间的所有通信都通过由 MobiLink 启动的出站连接来使用 HTTP(S)。

Sybase 中继服务器托管服务不用于生产部署。部署生产应用程序之前，必须先在您自己公司的基础结构中安装中继服务器。

使用中继服务器托管服务

以下几节介绍如何执行某些基本任务。

预订中继服务器托管服务

要使用 Sybase 中继服务器托管服务，必须先预订它。

- **预订 Sybase 中继服务器托管服务**

1. 从 web 浏览器转到 http://relayserver.sybase.com/account。即可访问 Sybase 中继服务器托管服务主页。
2. 通过单击 [**注册**] 来创建一个帐户。
3. 您需要指定 [**预订 ID**]（选择贵组织所独有的预订 ID）和 [**口令**]，提供您本人和贵组织的联系信息，并同意 [**托管中继服务的服务条款**]。单击 [**提交**]。

   成功注册之后，您会收到一封确认您的注册的电子邮件。

登录中继服务器托管服务

- **登录中继服务器托管服务**

1. 通过单击 [**登录**] 来登录到您新创建的帐户。
2. 输入您在注册过程中输入的 [**预订 ID**] 和 [**口令**]。登录后，您将被带到 [**帐户信息**] 页面。此帐户信息页面允许您修改预订者信息，并指定将访问此服务的后端服务器群。

添加服务器群

- **添加服务器群**

1. 单击 [**添加新群**]。
2. 输入唯一的 [**群名称**] 以描述服务器群。
3. 从下拉列表中选择 [**类型**]。
4. 为群中的每一台服务器提供唯一名称。最多可指定两台服务器。

5. 单击 [**创建群**]。如果群已添加成功，则将显示确认消息。
6. 单击 [**配置说明**] 以了解有关如何使用服务的详细信息。该说明基于您所提供的信息。
7. 完成后单击 [**注销**]。

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

将 MobiLink 与中继服务器结合使用

以下几节提供了有关将中继服务器与 MobiLink 结合使用的信息。

有关中继服务器支持哪些操作系统和浏览器的信息，请参见 http://www.sybase.com/detail?id=1062617。

有关部署出站启动器的信息，请参见“部署 MobiLink 服务器”一节第 751 页。

将客户端连接到中继服务器群

中继服务器群已正确配置后，客户端使用以下 URL 连接到该中继服务器群：

http://*<Relay Server client extension URI>*/*<farm>*

选项

选项	说明
<中继服务器客户端扩展 URI>	对于 Windows 上的 IIS，*<域名>/ias_relay_server/client/rs_client.dll* 对于 Linux 上的 Apache，*<域名>/cli/iarelayserver*
<群>	标识中继服务器将客户端请求转发到的后端群（一组后端服务器）。

SQL Anywhere MobiLink 客户端连接示例

SQL Anywhere MobiLink 客户端应指定以下选项来连接到服务器群 **F1**：

```
-e "ctp=http;
     adr='host=relayserver.sybase.com;
        url_suffix=/ias_relay_server/client/rs_client.dll/F1'"
```

对于 HTTPS，将 `http` 更改为 `https`。

UltraLite/UltraLiteJ MobiLink 客户端连接示例

UltraLite/UltraLiteJ MobiLink 客户端应在 ULSyncParms 类中设置以下属性来连接到服务器群 **F1**：

- 将流类型设置为 HTTPS。
- 将流参数设置为以下内容：

```
"host=testrelay.ianywhere.com; url_suffix=/ias_relay_server/client/
rs_client.dll/F1"
```

QAnywhere 客户端连接示例

QAnywhere 客户端应指定以下选项来连接到服务器群 **F1**：

```
-x "http(host=relayserver.sybase.com;url_suffix=/ias_relay_server/client/
rs_client.dll/F1"
```

方案示例

假设公司 ABC 开发了移动应用程序，现在想设置部署运行环境来服务于该移动应用程序。最初，移动部署包含 10000 台设备并会在将来逐渐增多。因此，客户希望容错和负载平衡环境能够处理当前负载，并且易于扩展以便将来处理更多的移动部署。基于移动应用程序的数据同步特性，客户确定需要以下配置：

- 2 个 MobiLink 服务器
- 2 个中继服务器
- 1 个负载平衡器

因为公司使用 IIS 作为其 web 服务器，所以使用中继服务器的 IIS 版本。

注意

- 每个中继服务器都部署在它自己的计算机上。使用两台主机名为 **rs1.abc.com** 和 **rs2.abc.com** 的计算机。
- 每个 MobiLink 服务器都部署在它自己的计算机上。这两个 MobiLink 服务器命名为 **ml1** 和 **ml2**，并属于名为 abc.mobilink 的后端服务器群。
- 可使用主机名 **www.abc.com** 寻址负载平衡器。
- 为最大程度地保证安全性，连接到中继服务器的所有客户端和出站启动器都使用 HTTPS。我们假定所有 Web 服务器都配备有知名证书颁发机构 (CA) 颁发的证书，并且所有后端服务器计算机的标准证书存储库中都有相应的受信任根证书。

♦ 设置中继服务器群

1. 第一步是创建中继服务器配置文件。

   包含配置的文件名必须是 ***rs.config***。对于此特定方案，使用以下配置文件：

```
#
# Options
#
[options]
verbosity = 1

#
# Define the Relay Server farm
#
[relay_server]
host = rs1.abc.com

[relay_server]
host = rs2.abc.com

#
# Define the MobiLink backend server farm
#
```

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

```
[backend_farm]
id = abc.mobilink
client_security = on
backend_security = on

#
# List MobiLink servers that are connecting to the Relay Server farm
#
[backend_server]
farm = abc.mobilink
id = ml1
token = mltoken1

[backend_server]
farm = abc.mobilink
id = ml2
token=mltoken2
```

2. 将配置文件 *rs.config* 以及中继服务器组件部署到两台运行中继服务器的计算机上。
3. 在正在运行 MobiLink 服务器的两台计算机上启动 MobiLink 服务器，然后使用以下命令启动相应的出站启动器。

   在运行 id 为 ml1 的 MobiLink 服务器的计算机上：

   ```
   mlsrv11 -x http -z ml1 -ss <other ML options>
   rsoe -f abc.mobilink -id ml1 -t mltoken1 -cr
   "host=www.abc.com;port=443;https=1"
   ```

   在运行 id 为 ml2 的 MobiLink 服务器的计算机上：

   ```
   mlsrv11 -x http -z ml2 -ss <other ML options>
   rsoe -f abc.mobilink -id ml2 -t mltoken2 -cr
   "host=www.abc.com;port=443;https=1"
   ```

4. 所有服务器和出站启动器运行后，MobiLink 客户端能够使用以下连接信息连接到群：
   - **HTTPS** protocol
   - **host** www.abc.com
   - **url_suffix** /ias_relay_server/client/rs_client.dll/abc.mobilink

重定向器（不建议使用）

目录

重定向器（不建议使用）简介 250
设置重定向器 252
为重定向器配置 MobiLink 客户端和服务器 253
配置重定向器属性 255
Windows 上 Netscape/Sun Web 服务器的 NSAPI 重定向器（不建议使用） 261
Unix 上 Netscape/Sun Web 服务器的 NSAPI 重定向器（不建议使用） 263
用于 Microsoft Web 服务器的 ISAPI 重定向器（不建议使用） 265
Servlet 重定向器（不建议使用） 267
Apache 重定向器（不建议使用） 269
M-Business Anywhere 重定向器（不建议使用） 271

注意
现已不建议使用重定向器。可以使用中继服务器代替重定向器。请参见“中继服务器”第 223 页。

重定向器（不建议使用）简介

注意
现已不建议使用重定向器。可以使用中继服务器代替重定向器。请参见“中继服务器”第 223 页。

MobiLink 包括一个称为 **Redirector** 的 Web 服务器扩展，它会对客户端与 MobiLink 服务器之间的请求和响应进行路由。这样的插件通常也称为 **reverse proxy**。

之所以要通过 Web 服务器路由请求，主要是因为可以利用 Web 服务器和防火墙的现有配置进行 HTTP 或 HTTPS 同步。不过，Web 服务器可在没有重定向器的情况下作为代理来运行。重定向器在有多个 MobiLink 服务器时最有用处。

请参见“使用 Web 服务器时的选项”一节第 251 页。

通过使用重定向器，可以将 Web 服务器配置为将特定 URL 请求路由到运行 MobiLink 服务器的一台或多台计算机。

可以将 Web 服务器配置为将具有特定 URL 或特定范围 URL 的请求传递到通常以 Perl CGI 脚本、DLL 或其它扩充机制形式编写的扩充程序。这些扩充程序可以访问外部数据源并提供响应，以供 Web 服务器传送给其客户端。

负载平衡和故障转移

重定向器使用简单的轮转调度算法（按固定的循环顺序选择服务器）来实现负载平衡和故障转移。重定向器会强制每个 MobiLink 服务器做出回应并停止向不响应的服务器发送请求。MobiLink 服务器再次运行时重定向器会检测到这种情况，并在那时恢复向其发送请求。

HTTPS 同步

在 MobiLink 客户端上指定 HTTPS 协议时，远程数据库与 Web 服务器之间的连接使用的是 HTTPS：HTTP 标头在发送到 Web 服务器或从 Web 服务器发出之前，会通过使用 RSA 加密的 TLS 进行加密，之后 Web 服务器会对 HTTPS 进行解密并通过重定向器将 HTTP 发送到 MobiLink。所有重定向器都支持此版本的 HTTPS，该版本的 HTTPS 仅用于 MobiLink 客户端与 Web 服务器之间的连接。

HTTPS 协议的速度慢于其它安全协议。

完全 HTTPS

一些重定向器（如 Apache 重定向器、ISAPI 重定向器和 Windows 上的 NSAPI 重定向器）提供了一个选项，通过它可以将流重新加密为 HTTPS，然后将其发送到 MobiLink 服务器。

有关支持从重定向器到 MobiLink 服务器使用 HTTPS 的重定向器的列表，请参见 http://www.sybase.com/detail?id=1062632。

支持的 Web 服务器

下列 Web 服务器可以使用插件：

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

重定向器插件	...支持
ISAPI 重定向器	Microsoft Web 服务器
NSAPI 重定向器	Windows 和 Unix 上的 Sun One (Netscape) Web 服务器与 iPlanet Web 服务器
Servlet 重定向器	支持 Java Servlet API 2.3 的 Web 服务器，包括 Apache Tomcat 和 Unix 上的 Sun One Web 服务器
本机 Apache 重定向器	Apache Web 服务器
M-Business Anywhere 重定向器	M-Business Anywhere Web 服务器

有关重定向器支持的详细信息，请参见：

- http://www.sybase.com/detail?id=1062632

使用 Web 服务器时的选项

重定向器是一种通过 Web 服务器对 MobiLink 同步进行路由的方式。重定向器在跨防火墙进行同步或与多台 MobiLink 服务器进行同步时特别有用。

之所以要通过 Web 服务器路由请求，主要是因为可以利用 Web 服务器和防火墙的现有配置进行 HTTP 或 HTTPS 同步。重定向器在有多个 MobiLink 服务器时最有用处。

您也可以不使用重定向器，而是通过 Web 服务器对同步进行路由。在这种情况下，您可以将 Web 服务器配置为代理，以将同步路由到 MobiLink 服务器。有关如何通过 Web 服务器执行此操作的详细信息，请参见 Web 服务器文档。

下表中包含的建议可以帮助您决定如何最佳地对 MobiLink 同步进行路由。

	可以进行直接连接	不可进行直接连接
One MobiLink server	使用 TCP/IP 而非 HTTP	使用 HTTP 或 HTTPS 代理，将消息经由 Web 服务器传递到 MobiLink 服务器
Multiple MobiLink servers	将重定向器与 HTTP 或 HTTPS 配合使用	将重定向器与 HTTP 或 HTTPS 配合使用

请参见“重定向器（不建议使用）”第 249 页。

设置重定向器

注意
现已不建议使用重定向器。可以使用中继服务器代替重定向器。请参见“中继服务器”第 223 页。

下面的章节介绍如何配置 Web 服务器以管理同步请求。

♦ **配置过程概述**

1. 配置 MobiLink 服务器。

   请参见“为重定向器配置 MobiLink 客户端和服务器”一节第 253 页。

2. 修改重定向器配置文件。执行此操作有两种方法，采用哪一种方法取决于您使用的重定向器是否支持服务器组。请参见：

   - “配置重定向器属性（适用于支持服务器组的重定向器）”一节第 256 页
   - “配置重定向器属性（适用于不支持服务器组的重定向器）”一节第 258 页

3. 进行特定于 Web 服务器的配置。

   参见下列内容之一：

   - “Windows 上 Netscape/Sun Web 服务器的 NSAPI 重定向器（不建议使用）”一节第 261 页
   - “用于 Microsoft Web 服务器的 ISAPI 重定向器（不建议使用）”一节第 265 页
   - “Servlet 重定向器（不建议使用）”一节第 267 页
   - “Apache 重定向器（不建议使用）”一节第 269 页
   - “M-Business Anywhere 重定向器（不建议使用）”一节第 271 页

4. 配置 MobiLink 客户端。

   请参见“为重定向器配置 MobiLink 客户端和服务器”一节第 253 页。

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

为重定向器配置 MobiLink 客户端和服务器

注意
现已不建议使用重定向器。可以使用中继服务器代替重定向器。请参见“中继服务器”第 223 页。

本节说明要如何配置 MobiLink 客户端和 MobiLink 服务器，才能通过 Web 服务器进行同步。以下过程设置通过 Web 服务器定向请求时所需的参数。

MobiLink 客户端

♦ 配置 MobiLink 客户端（SQL Anywhere 和 UltraLite 客户端）

1. 将 MobiLink 客户端的通信类型指定为 HTTP 或 HTTPS。

   有关设置 SQL Anywhere 客户端通信类型的详细信息，请参见“CommunicationType (ctp) 扩展选项”一节 《MobiLink - 客户端管理》。

   有关设置 UltraLite 客户端通信类型的详细信息，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》。

2. 为 MobiLink 客户端指定以下 HTTP/HTTPS 同步协议选项：
   - **host** Web 服务器的名称或 IP 地址。
   - **port** Web 服务器接受 HTTP 或 HTTPS 请求的端口。
   - **url_suffix** 此设置取决于所用重定向器的类型：
     - 如果是 ISAPI 重定向器，则为：

       ```
       exe_dir/iaredirect.dll/ml/[server-group/]
       ```

       其中 *exe_dir* 是 *iaredirect.dll* 的位置，*server-group*（可选项）是组的名称。
     - 如果是 NSAPI 重定向器，则为：

       ```
       mlredirect/ml/[server-group/]
       ```

       其中 *mlredirect* 是在 *obj.conf* 文件中映射的名称。
     - 如果是 servlet 重定向器，则为：

       ```
       iaredirect/ml/
       ```
     - 如果是 Apache 的本地重定向器，请将此项设置为您在 *httpd.conf* 文件中重定向器的 <location> 标记内选择的任何内容。例如，如果该位置为 `<Location /iaredirect/ml>`，则 url_suffix 是：

       ```
       iaredirect/ml/
       ```
     - 如果是 M-Business Anywhere 重定向器，请将此项设置为您在 *sync.conf* 文件中重定向器的 <location> 标记内选择的任何内容。例如，如果该位置为 `<Location /iaredirect/ml>`，则 url_suffix 是：

       ```
       iaredirect/ml/
       ```

请参见“url_suffix”一节 《MobiLink - 客户端管理》。

有关设置 UltraLite 客户端协议选项的详细信息，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》。

有关设置 SQL Anywhere 客户端协议选项的详细信息，请参见“MobiLink 客户端网络协议选项汇总”一节 《MobiLink - 客户端管理》。

MobiLink 服务器

♦ 配置 MobiLink 服务器

1. 对于支持 HTTPS 的重定向器，可以使用 HTTPS 协议启动 MobiLink 服务器。有关支持 HTTPS 的重定向器的列表，请参见 http://www.sybase.com/detail?id=1062632。

   对于不支持 HTTPS 的重定向器，必须使用 HTTP 协议启动 MobiLink 服务器，在客户端与代理之间进行通信时才能使用 HTTP 或 HTTPS。这些重定向器无法直接使用 HTTPS。

   例如，可以按照以下方法在 mlsrv11 命令行上指定 HTTP 协议：

   ```
   mlsrv11 -x http
   ```

   请参见“-x 选项”一节第 102 页。

2. 此外，还可能需要为 MobiLink 服务器设置以下参数：

   - **port** 对于 HTTP 协议，MobiLink 缺省使用端口 80。对于 HTTPS 协议，MobiLink 缺省使用端口 443。如果 MobiLink 服务器与 Web 服务器在同一台计算机上运行，端口 80 通常由 Web 服务器使用。在此种情况下，您必须指定其它端口。例如，您可以使用端口 2439，它是在 Internet 编号授权委员会（Internet Assigned Numbers Authority，简称 IANA）注册的 MobiLink 服务器端口号。

   有关端口的详细信息，请参见“-x 选项”一节第 102 页。

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

配置重定向器属性

注意
现已不建议使用重定向器。可以使用中继服务器代替重定向器。请参见“中继服务器”第 223 页。

支持服务器组的重定向器和不支持服务器组的重定向器具有不同的属性。

MobiLink 服务器组

可以将 MobiLink 服务器分区，形成若干个服务器组。这样您便拥有了能够访问不同 MobiLink 服务器组的客户端组。

有关支持服务器组的重定向器的列表，请参见 http://www.sybase.com/detail?id=1062632。

创建服务器组的方法是在重定向器配置文件 (*redirector_server_group.config*) 中添加一个区段，区段的内容是加有方括号的组名，其后是该组的设置。一个组必须至少指定一个 ML 指令。还可以为组设置 ML_CLIENT_TIMEOUT 选项。在客户端上通过 url_suffix 选项引用组。

您可以通过在文件中指定任何已命名组之前指定无名称的组来创建缺省服务器组。此缺省组对保持向后兼容性有用处。客户端未在其 url_suffix 选项中指定服务器组名时，便会使用此缺省组。

请参见“url_suffix”一节 《MobiLink - 客户端管理》。

您还可以为所有服务器组指定 SLEEP 和 LOG_LEVEL 属性的缺省设置。可以在配置文件中的任何位置指定这些设置。

支持新旧客户端

如果 MobiLink 服务器需要支持版本 8 或 9 的远程数据库以及版本 10 或更高版本的远程数据库，则至少需要打开两个端口：使用 mlsrv11 -x 选项打开供新客户端使用的端口，使用 mlsrv11 -xo 选项打开供旧客户端使用的端口。如果同时使用重定向器，则需要对服务器组进行设置，以使重定向器将客户端导向相应的端口。

在典型的重定向器设置下，您会启动多个 MobiLink 服务器。最简单的情况是，您运行一个 MobiLink 服务器，分别使用 -x 和 -xo 打开了两个端口，并创建两个服务器组，每组对应一个端口。以下是 mlsrv11 命令行的一部分，用于为 MobiLink 服务器打开两个端口：

```
mlsrv11 -c "dsn=YourDSN" -x http(port=111) -xo http(port=222)
```

在重定向器配置文件中添加用于这两个服务器组的区段：

```
[v10service]
    ML="host=mySrv.myCorp.com;port=111"
[v9service]
    ML="host=mySrv.myCorp.com;port=222"
```

启动客户端时，以服务器组的名称指定 url_suffix 选项。例如，对于 SQL Anywhere 客户端和 ISAPI Web 服务器，版本 10 客户端的 dbmlsync 命令行的一部分将如下所示：

```
dbmlsync -e "adr='host=somehost;port=5001;url_suffix=scripts/iaredirect.dll/
ml/v10service'"...
```

对于版本 9 的客户端，dbmlsync 命令行的部分内容如下：

```
dbmlsync -e "adr='host=somehost;port=5001;url_suffix=scripts/iaredirect.dll/
ml/v9service'"...
```

另请参见

- “-x 选项”一节第 102 页
- “-xo 选项”一节第 107 页
- “配置重定向器属性（适用于支持服务器组的重定向器）”一节第 256 页
- “url_suffix”一节 《MobiLink - 客户端管理》

示例

以下是一个 *redirector_server_group.config* 示例文件，显示了服务器组的一些典型设置以及创建方式。

```
#
# Set up the default server group:
#
  ML="host=mySrv1.myCorp.com;port=222"
  ML="host=mySrv2.myCorp.com;port=222"
#
# Set up a server group named myOldGroup:
#
[myOldGroup]
  ML="host=myOldSrv1.myCorp.com;port=111"
  ML="host=myOldSrv2.myCrop.com;port=111"
  ML_CLIENT_TIMEOUT=30
#
# Set up a server group named myNewGroup:
#
[myNewGroup]
  ML="host=myNewSrv1.myCorp.com;port=333"
  ML="host=myNewSrv2.myCorp.com;port=555"
  ML_CLIENT_TIMEOUT=240
#
# Set up a server group named mlSecureGroup:
#
[theirSecureGroup]
  ML="https=true;Srv1.Corp.com;trusted_certificates=c:\Corp\publicRoot.crt"
  ML="https=true;Srv2.Corp.com;trusted_certificates=c:\Corp\publicRoot.crt"
#
# Set global properties:
#
    LOG_LEVEL=5
    SLEEP=15
```

配置重定向器属性（适用于支持服务器组的重定向器）

本节介绍配置重定向器属性的通用 Web 服务器配置步骤。它适用于支持服务器组的重定向器。

有关支持服务器组的重定向器的列表，请参见 http://www.sybase.com/detail?id=1062632。

有关服务器组的信息，请参见“MobiLink 服务器组”一节第 255 页。

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

♦ 配置重定向器属性

1. 完成“为重定向器配置 MobiLink 客户端和服务器”一节第 253 页中的步骤。
2. 配置 **Redirector configuration file**。名为 *redirector_server_group.config* 的模板文件位于 *install-dir\MobiLink\redirector* 中。配置重定向器配置文件最简单的方法是修改 *redirector_server_group.config*。

   重定向器配置文件受限于以下规则：

   - 行最大长度为 2000 个字符。
   - 注释以散列字符 (#) 开头。
   - 如果是 ISAPI 重定向器，必须将配置文件命名为 *redirector.config*，而且该文件必须与 *iaredirect.dll* 处在同一目录中。

   可在此文件中设置以下指令：

   - **服务器组** 要创建服务器组，请在 *redirector_server_group.config* 中创建区段，区段开头是加有方括号的服务器组名，然后定义服务器组。

     请参见“MobiLink 服务器组”一节第 255 页。
   - **LOG_LEVEL** 用于控制写入日志文件的输出量。取值范围是 0 到 7，数字越大，生成的输出量越大。缺省情况下，日志文件的名称是 *redirector.log*，位于与重定向器配置文件相同的位置。如果是 NSAPI 重定向器，可以使用 logFile 指令更改 magnus.conf 中的名称和位置。
   - **ML** 可以通过两种方式使用 ML 指令：
     - 如果重定向器不支持从重定向器到 MobiLink 服务器使用 HTTPS，或您未使用 HTTPS，则可以使用 ML 指令以 `ML=host:port` 的格式指定运行 MobiLink 服务器的计算机列表。要指定多台计算机，请在不同的行中重复使用此语法。例如：

       ```
       ML=209.123.123.1:8080
       ML=myCompany.com:8081
       ```

     - 如果重定向器支持从重定向器到 MobiLink 服务器使用 HTTPS，并且正在使用 HTTPS，则您应该以由分号分隔的列表形式指定 MobiLink 客户端网络协议选项，如下所示：

       ML="https=true;*network-client-options***;..."**

       例如，

       ```
       ML="https=true;host=My-pc;port=82;trusted_certificates=rsaroot.crt"
       ```

       有关网络客户端选项的列表，请参见“MobiLink 客户端网络协议选项”《MobiLink - 客户端管理》。

       有关重定向器中 HTTPS 支持情况的说明，请参见“完全 HTTPS”一节第 250 页。

       有关支持从重定向器到 MobiLink 服务器使用 HTTPS 的重定向器的列表，请参见 http://www.sybase.com/detail?id=1062632。

   MobiLink 服务器启动时使用的协议和端口号必须与 ML 指令中所指定的相同。如果有差异，必须停止 MobiLink 服务器并以正确的信息重新启动它。

- **ML_CLIENT_TIMEOUT** 用于确保 MobiLink 服务器能够从同一远程数据库中检测出重复的同步。应将该超时设置为任何使用同一服务器组的客户端的最大超时。如果将此属性设置为 0，便会立即允许重新同步至不同的服务器。缺省值为 240 秒。
- **SLEEP** 用于设置重定向器检查服务器是否工作正常的间隔秒数。重定向器会检查一个服务器，等待此选项中设置的时间后再检查下一个服务器，如此循环。例如，`SLEEP=10`。SLEEP 区分大小写。缺省值为 20 秒。

3. 将重定向器配置文件复制到 Web 服务器。

   如果 MobiLink 服务器未安装在 Web 服务器所在的计算机上，请将重定向器配置文件复制到 Web 服务器所在的计算机（或复制到计算机能够访问的驱动器）。

   如果是 ISAPI Web 服务器，请将重定向器配置文件复制到目录 *Inetpub\scripts*，然后将其重命名为 *redirector.config*。

   如果是其它 Web 服务器，可以将重定向器配置文件复制到任何目录。

4. 在以下区段之一中完成 Web 服务器特定的配置：
   - “用于 Microsoft Web 服务器的 ISAPI 重定向器（不建议使用）”一节第 265 页
   - “Windows 上 Netscape/Sun Web 服务器的 NSAPI 重定向器（不建议使用）”一节第 261 页

示例

下面是一个示例重定向器配置文件。此文件指定以下内容：

- 重定向器应在确认服务器工作正常后休眠 10 秒。
- 三台运行能够处理请求的 MobiLink 服务器的计算机。

```
SLEEP=10
ML=myServ-pc:80
ML=209.123.123.1:8080
ML=myCompany.com:8081
```

配置重定向器属性（适用于不支持服务器组的重定向器）

本节介绍配置重定向器属性的通用 Web 服务器配置步骤。它适用于不支持服务器组的重定向器。

有关支持服务器组的重定向器的列表，请参见 http://www.sybase.com/detail?id=1062632。

♦ 配置重定向器属性

1. 完成“为重定向器配置 MobiLink 客户端和服务器”一节第 253 页中的步骤。
2. 将 *redirector.config* 复制到 Web 服务器。

   文件 *redirector.config* 位于 *install-dir\MobiLink\redirector* 中。

   如果 MobiLink 服务器不是安装在 Web 服务器所在的计算机上，请将 *redirector.config* 复制到 Web 服务器所在的计算机。

3. 配置重定向器配置文件。

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

要配置 Web 服务器与 MobiLink 服务器之间的通信，您必须编辑 Web 服务器所在计算机上的 *redirector.config* 文件。

以下规则适用于 *redirector.config*：

- 行的最大长度为 300 个字符。
- 注释以散列字符 (#) 开头。
- 指令定义中不能包含空格和制表符。

可在此文件中设置以下指令：

- **LOG_LEVEL** 用于控制写入日志文件的输出量。值可为 0、1 和 2，其中 1 为缺省值，2 所产生的输出量最大。如果是 Apache 重定向器，则此设置不起作用；请在 Apache 配置文件 *httpd.conf* 的 LogLevel 区段设置日志级别。
- **ML** ML 区分大小写。可以通过两种方式使用 ML 指令。

  如果重定向器不支持 HTTPS，或您未使用 HTTPS，则可以使用 ML 指令以 `ML=host:port` 的格式指定运行 MobiLink 服务器的计算机列表。要指定多台计算机，请在不同的行中重复使用此语法。例如：

  ```
  ML=209.123.123.1:8080
  ML=myCompany.com:8081
  ```

  如果重定向器支持从重定向器到 MobiLink 服务器使用 HTTPS，则可以通过以分号分隔的列表指定 MobiLink 客户端网络协议选项，如下所示：

  ML="https=true;*network-client-options*;..."

  例如，

  ```
  ML="https=true;host=My-pc;port=82;trusted_certificates=rsaroot.crt"
  ```

  有关网络客户端选项的列表，请参见“MobiLink 客户端网络协议选项”《MobiLink - 客户端管理》。

  有关支持从重定向器到 MobiLink 服务器使用 HTTPS 的重定向器的列表，请参见 http://www.sybase.com/detail?id=1062632。

  MobiLink 服务器启动时使用的协议和端口号必须与 ML 指令中所指定的相同。如果有差异，必须停止 MobiLink 服务器并以正确的信息重新启动它。

- **ML_CLIENT_TIMEOUT** 用于确保单个同步的每一步都定向到同一 MobiLink 服务器。重定向器为 ML_CLIENT_TIMEOUT 过程维持客户端和服务器之间的关联。此值还用于确保 MobiLink 服务器能够从同一远程数据库中检测出重复的同步。此参数的值应该大于任何用户同步中的最长步骤所用的时间。

  缺省值为 600 秒（10 分钟）。

- **SLEEP** 用于设置重定向器检查服务器是否工作正常的间隔秒数。缺省值为 1800 （30 分钟）。例如，`SLEEP=3600`。SLEEP 区分大小写.

4. 在以下区段之一中完成 Web 服务器特定的配置：
   - “Unix 上 Netscape/Sun Web 服务器的 NSAPI 重定向器（不建议使用）”一节第 263 页
   - “Servlet 重定向器（不建议使用）”一节第 267 页

- “Apache 重定向器（不建议使用）”一节第 269 页
- “M-Business Anywhere 重定向器（不建议使用）”一节第 271 页

示例

下面是一个 *redirector.config* 示例文件。此文件指定以下内容：

- 重定向器应每隔 1800 秒检查一次服务器是否处于工作状态。
- 三台运行能够处理请求的 MobiLink 服务器的计算机。指定多个服务器时系统会自动启用负载平衡。

```
SLEEP=1800
ML=myServ-pc:80
ML=209.123.123.1:8080
ML=myCompany.com:8081
```

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

Windows 上 Netscape/Sun Web 服务器的 NSAPI 重定向器（不建议使用）

注意
现已不建议使用重定向器。可以使用中继服务器代替重定向器。请参见“中继服务器”第 223 页。

NSAPI 重定向器是为 Sun Java System Web 服务器提供的，该服务器以前称作 Sun One 和 Netscape iPlanet Enterprise Edition Web 服务器。

有关所支持版本的信息，请参见 http://www.sybase.com/detail?id=1062632。

要在 Unix 上使用此重定向器，请参见“Unix 上 Netscape/Sun Web 服务器的 NSAPI 重定向器（不建议使用）”一节第 263 页。

要将该重定向器与其它平台上的 Netscape/Sun Web 服务器配合使用，可以使用 Servlet 重定向器。请参见“Servlet 重定向器（不建议使用）”一节第 267 页。

♦ 配置 NSAPI 重定向器

1. 完成“配置重定向器属性（适用于支持服务器组的重定向器）”一节第 256 页中的步骤。
2. 如有必要，请将文件 *iaredirect.dll* 复制到 Web 服务器所在的计算机。此文件位于 *install-dir\MobiLink\redirector\web-server* 下，其中 *web-server* 是 NSAPI web 服务器的名称。
3. 如果 Web 服务器不是位于重定向器所在的计算机，则必须将以下文件复制到该计算机，并确保这些文件位于您的路径 中。您所需的文件视所使用的加密方式（如果有）而定。

   以下文件位置相对于 *install-dir*：

设置	所需文件
全部	● *bin32\dblgen11.dll*[1] ● *bin32\dbicu11.dll* ● *bin32\dbicudt11.dll*
ECC 加密	● *bin32\mlcecc11.dll*
RSA 加密	● *bin32\mlcrsa11.dll*
经 FIPS 认可的 RSA 加密	● *bin32\mlcrsafips11.dll* ● *bin32\sbgse2.dll*

   [1] 对于德语、日语和中文版本，请将 en 分别替换为 de、ja 和 zh。

   有关如何更改语言的信息，请参见“了解区域设置语言”一节 《SQL Anywhere 服务器 - 数据库管理》。
4. 按如下说明更新 NSAPI Web 服务器的配置文件 *magnus.conf* 和 *obj.conf*。

所提供的示例文件
为 MobiLink 服务器预配置的 *magnus.conf* 和 *obj.conf* 的示例副本，位于 *install-dir\MobiLink\redirector\web-server* 中，其中 *web-server* 是 NSAPI Web 服务器的名称。

更新 *magnus.conf* 和 *obj.conf* 的以下区段。

- 在 *magnus.conf* 中，指定 *iaredirect.dll* 和重定向器配置文件的位置。

  在 Init 区段结尾添加以下文本，其中 *location* 是文件的实际位置（虽然 *iaredirect.dll* 和重定向器配置文件必须都位于 Web 服务器所在的计算机上或 Web 服务器可以访问的驱动器上，但它们可以处于不同的位置）：

```
Windows:
Init fn="load-modules" shlib="location/iaredirect.dll"
funcs="redirector,initialize_redirector"
Init fn="initialize_redirector" configFile="location/redirector.config"
```

- 在 *obj.conf* 中，指定要在 URL 中使用的重定向器的名称。

  在缺省对象区段的开头添加以下文本。除了可以将 *mlredirect* 更改为所需的任何内容外，此区段的其它内容应与下面所示的完全相同。所有 *http://host:port/mlredirect/ml/** 形式的请求都将发送到与重定向器配合运行的其中一个 MobiLink 服务器。

```
<Object name=default>
NameTrans fn="assign-name" from="/mlredirect/ml/*" name="redirectToML"
```

- 在 *obj.conf* 中，指定重定向器调用的对象。在 default object 区段后添加以下区段：

```
<Object name="redirectToML">
Service fn="redirector" serverType="ml"
</Object>
```

示例

以下是 *magnus.conf* 中需要进行自定义的区段的示例。

```
Init fn="load-modules" shlib="D:/iaredirect.dll"
funcs="redirector,initialize_redirector"
Init fn="initialize_redirector" configFile="D:/redirector.config"
```

以下是 *obj.conf* 中对重定向器有重要意义的区段的示例：

```
<Object name=default>
NameTrans fn="assign-name" from="/mlredirect/ml/*" name="redirectToML"
...
<Object name="redirectToML">
Service fn="redirector" serverType="ml"
</Object>
```

♦ 测试您的配置

1. 使用以下语法调用重定向器：

```
http://host:port/mlredirect/ml/
```

2. 检查日志文件以了解重定向器是否记录了请求。

   注意：此测试不会建立到 MobiLink 服务器的连接。

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

Unix 上 Netscape/Sun Web 服务器的 NSAPI 重定向器（不建议使用）

注意
现已不建议使用重定向器。可以使用中继服务器代替重定向器。请参见“中继服务器”第 223 页。

NSAPI 重定向器是为 Sun Java System Web 服务器提供的，该服务器以前称作 Sun One 和 Netscape iPlanet Enterprise Edition Web 服务器。

有关所支持版本的信息，请参见 http://www.sybase.com/detail?id=1062632。

要在 Windows 上使用此重定向器，请参见“Windows 上 Netscape/Sun Web 服务器的 NSAPI 重定向器（不建议使用）”一节第 261 页。

要将该重定向器与其它平台上的 Netscape/Sun Web 服务器配合使用，可以使用 Servlet 重定向器。请参见“Servlet 重定向器（不建议使用）”一节第 267 页。

♦ 配置 NSAPI 重定向器

1. 完成“配置重定向器属性（适用于不支持服务器组的重定向器）”一节第 258 页中的步骤。
2. 如有必要，请将文件 *iaredirect.so* 复制到 Web 服务器所在的计算机。此文件位于 *install-dir\MobiLink\redirector\web-server* 下，其中 *web-server* 是 NSAPI web 服务器的名称。
3. 按如下说明更新 NSAPI Web 服务器的配置文件 *magnus.conf* 和 *obj.conf*。

所提供的示例文件
magnus.conf 和 *obj.conf* 的示例副本位于 *install-dir\MobiLink\redirector\web-server* 中，其中 *web-server* 是 NSAPI Web 服务器的名称。可使用这些示例文件确认将以下部分放置于该文件的何处。

更新文件 *magnus.conf* 和 *obj.conf* 的以下区段。

- 在 *magnus.conf* 中，指定 *iaredirect.so* 和 *redirector.config* 的位置。

  在 Init 区段的尾部添加以下文本，其中 *location* 为文件的实际位置。（尽管 *iaredirect.so* 和 *redirector.config* 都必须位于 Web 服务器所在的计算机上，但它们可以处于不同位置。）

  ```
  Solaris:
  Init fn="load-modules" shlib="location/iaredirect.so"
  funcs="redirector,initialize_redirector"
  Init fn="initialize_redirector" configFile="location/redirector.config"
  ```

- 在 *obj.conf* 中，指定要在 URL 中使用的重定向器的名称。

  在 "default object" 区段的起始处添加以下文本。除了可以将 *mlredirect* 更改为所需的任何内容外，此区段的其它内容应与下面所示的完全相同。所有 *http://host:port/mlredirect/ml/** 形式的请求都将发送到与重定向器配合运行的其中一个 MobiLink 服务器。

  ```
  <Object name=default>
  NameTrans fn="assign-name" from="/mlredirect/ml/*" name="redirectToML"
  ```

- 在 *obj.conf* 中，指定重定向器调用的对象。在 "default object" 区段后添加以下区段：

```
<Object name="redirectToML">
Service fn="redirector" serverType="ml"
</Object>
```

示例

以下是 *magnus.conf* 中需要进行自定义的区段的示例。

```
Init fn="load-modules" shlib="location/iaredirect.so"
funcs="redirector,initialize_redirector"
Init fn=" initialize_redirector " configFile="location/redirector.config"
```

以下是 *obj.conf* 中对重定向器有重要意义的区段的示例。

```
<Object name=default>
NameTrans fn="assign-name" from="/mlredirect/ml/*" name="redirectToML"
...
<Object name="redirectToML">
Service fn="redirector" serverType="ml"
</Object>
```

♦ 测试您的配置

1. 使用以下语法调用重定向器：

   ```
   http://host:port/mlredirect/ml/
   ```

2. 检查日志文件以了解重定向器是否记录了请求。

   注意：此测试不会建立到 MobiLink 服务器的连接。

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

用于 Microsoft Web 服务器的 ISAPI 重定向器（不建议使用）

注意
现已不建议使用重定向器。可以使用中继服务器代替重定向器。请参见“中继服务器”第 223 页。

如果使用的是 Microsoft Web 服务器，则可以使用 ISAPI 版本的重定向器。

有关所支持版本的信息，请参见 http://www.sybase.com/detail?id=1062632。

♦ 为 Microsoft Web 服务器配置 ISAPI 重定向器

1. 完成“配置重定向器属性（适用于支持服务器组的重定向器）”一节第 256 页中的步骤。
2. 将文件 *iaredirect.dll* 复制到 Web 服务器所在计算机的 *Inetpub\scripts* 中。

   文件 *iaredirect.dll* 位于 *install-dir\MobiLink\redirector\IIS5* 中。

   目录 *Inetpub\scripts* 位于 Microsoft Web 服务器安装目录中。
3. 如果 Web 服务器不是位于重定向器所在的计算机，则必须将以下文件复制到该计算机，并确保这些文件位于您的路径 中。您所需的文件视所使用的加密方式（如果有）而定。

   以下文件位置相对于 *install-dir*：

Setup	所需文件
全部	● *bin32\dblgen11.dll*[1] ● *bin32\dbicu11.dll* ● *bin32\dbicudt11.dll*
ECC 加密	● *bin32\mlcecc11.dll*
RSA 加密	● *bin32\mlcrsa11.dll*
经 FIPS 认可的 RSA 加密	● *bin32\mlcrsafips11.dll* ● *bin32\sbgse2.dll*

[1] 对于德语、日语和中文版本，请将 en 分别替换为 de、ja 和 zh。

有关如何更改语言的信息，请参见“了解区域设置语言”一节 《SQL Anywhere 服务器 - 数据库管理》。

注意

♦ 测试您的配置

1. 使用以下语法调用 ISAPI 重定向器：

```
protocol://host[:port]/exec_dir/iaredirect.dll/ml/
```

其中：

- **protocol** 为 http 或 https。
- **host** 为 Web 服务器的主机名。
- **port** 为 Web 服务器监听的端口（如果不是缺省端口）。
- **exec_dir** 为重定向器 DLL *iaredirect.dll* 的安装目录。缺省目录为 scripts。

  例如，

  ```
  http://server:8080/scripts/iaredirect.dll/ml/
  ```

2. 检查日志文件以了解重定向器是否记录了请求。

   注意：此测试不会建立到 MobiLink 服务器的连接。

3. 如果配置不成功，请检查以下内容：
   - 目录 *Inetpub\scripts* 应在安装 Web 服务器的过程中使用执行权限进行创建。
   - 仅当使用 Internet 信息服务为其它目录授予执行权限后，才可以将重定向器配置文件和 *iaredirect.dll* 置于该目录中。
   - 必须具有指向 *Inetpub\scripts* 目录的虚拟目录。如果没有该目录，则必须打开 Internet 信息服务，手工创建一个虚拟目录。此虚拟目录应指向 *Inetpub\scripts*，且其 [执行权限] 设置为 [脚本和可执行文件]。有关说明，请参见 IIS 联机帮助。

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

Servlet 重定向器（不建议使用）

注意
现已不建议使用重定向器。可以使用中继服务器代替重定向器。请参见“中继服务器”第 223 页。

为支持 Java servlet 规范版本 2.3 及更高版本的 Web 服务器提供了 servlet 重定向器。以下过程是如何设置适用于 Tomcat 版本 5.5.9 和 Apache 2.0.55 的 Servlet 重定向器的一个示例。

有关所支持版本的信息，请参见 http://www.sybase.com/detail?id=1062632。

还有一个适用于 Apache Web 服务器的本地重定向器。有关详细信息，请参见“Apache 重定向器（不建议使用）”一节第 269 页。

概述

本节介绍如何安装 Servlet 版本的重定向器以在 Apache Web 服务器上与 Tomcat Servlet 容器配合使用。

安装所需的步骤如下：

♦ 为 Apache Tomcat 配置 Servlet 重定向器

1. 完成“配置重定向器属性（适用于不支持服务器组的重定向器）”一节第 258 页中的步骤。
2. 在 Tomcat 中安装重定向器的 servlet 版本。
3. 配置 Apache Web 服务器，以使其作为代理运行。

在 Tomcat 中安装 servlet 重定向器

在以下步骤中，*%CATALINA_HOME%* 是 Tomcat 安装的根目录。

♦ 在 Tomcat 中安装 servlet 重定向器

1. 将 Tomcat 作为独立服务器安装。
2. 或者，设置所需的 Tomcat HTTP 端口。

   缺省情况下，Tomcat 绑定到端口 8080。如果存在冲突，可能是因为其它 Web 服务器正在使用此端口，

   - 打开文件：*%CATALINA_HOME%/conf/server.xml*。
   - 搜索 8080（位于 <Connector> 标记内）。
   - 将其更改为一个空闲端口。
3. 将 Servlet 重定向器作为一个 Web 应用程序进行安装。
   - 将 *iaredirect.war* 文件复制到 *%CATALINA_HOME%/webapps*。
   - 关闭并重新启动 Tomcat。

     Tomcat 会展开该 war 文件，并为重定向器 Web 应用程序创建目录 *iaredirect*。

Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

- 编辑文件 *%CATALINA_HOME%/webapps/iaredirect/WEB-INF/web.xml*。搜索 **redirector.config**（在 <init-param> 标记内），并更正 *redirector.config* 文件的路径。

  将条目 **redirector.config** 更改为 `drive:/path/redirector.config`。即使是在 Windows 操作系统上，也请使用正斜线作为路径分隔符，例如 *d:/redirector.config*。
- 关闭 Tomcat 并重新启动，以便更改生效。

  更改生效后，部署的位置不再需要 war 文件。
- 现在，重定向器可以通过以下 URL 来调用：

  http://*tc-host*:*tc-port*/iaredirect/ml/

  其中 *tc-host* 是计算机，*tc-port* 是 Tomcat 监听的端口。

将 Apache Web 服务器配置为代理

在以下步骤中，*%APACHE_HOME%* 是 Apache 安装的根目录。

♦ 将 Apache Web 服务器配置为代理

1. 安装 Apache Web 服务器。
2. （可选步骤）更改 Apache Web 服务器端口：
   - 编辑文件 *%APACHE_HOME%/conf/httpd.conf*，更改 **Port** 设置。
3. 配置 Apache 以作为代理运行：

   在 *%APACHE_HOME%/conf/httpd.conf* 中，添加以下指令：

   ```
   LoadModule proxy_module module-path/mod_proxy.so
   LoadModule proxy_connect_module module-path/mod_proxy_connect.so
   LoadModule proxy_http_module module-path/mod_proxy_http.so
   ```

   其中 *module-path* 是模块的位置。例如，路径可以是 *modules/mod_proxy.so*（缺省值）。
4. 配置 Apache 以将重定向器 URL 转发给 Tomcat。

   在 *%APACHE_HOME%/conf/httpd.conf* 中，添加以下指令，以便 Apache 将 http://localhost/iaredirect/* 形式的 URL 转发给监听端口 8080 的 Tomcat 5 Connector：

   ```
   ProxyPass /iaredirect http://localhost:8080/iaredirect
   ```

   端口号必须与 Tomcat 所用的端口号匹配。如果 Tomcat 和 Apache 不是运行在同一台计算机上，请提供运行 Tomcat 的计算机名以代替 **localhost**。

验证设置

♦ 检查配置

1. 使用以下语法调用重定向器：

   ```
   http://host:port/iaredirect/ml/
   ```
2. 检查日志文件以了解重定向器是否记录了请求。

   注意：此测试不会建立到 MobiLink 服务器的连接。

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

Apache 重定向器（不建议使用）

注意
现已不建议使用重定向器。可以使用中继服务器代替重定向器。请参见“中继服务器”第 223 页。

以下是为 Apache Web 服务器编写的设置说明。

有关所支持版本的信息，请参见 http://www.sybase.com/detail?id=1062632。

如果使用的是 Tomcat，则可以同时使用 Servlet 重定向器。有关详细信息，请参见“Servlet 重定向器（不建议使用）”一节第 267 页。

♦ 配置 Apache 重定向器

1. 完成“配置重定向器属性（适用于不支持服务器组的重定向器）”一节第 258 页中的步骤。
2. 按如下说明将文件 *mod_iaredirect.dll* 或 *mod_iaredirect.so* 复制到 Web 服务器中的相应目录：
   - 对于 Windows 上的 Apache，文件 *mod_iaredirect.dll* 位于 *install-dir\MobiLink\redirector\apache\v20* 中。将此文件复制到 Web 服务器所在计算机上的 *%apache-home%\modules* 目录。
   - 对于用于 Solaris 或 Linux 的 Apache，文件 *mod_iaredirect.so* 位于 *install-dir\MobiLink\redirector\apache\v20* 中。将它复制到 Web 服务器所在计算机上的 *$APACHE_HOME/modules* 目录。
3. 如果 Web 服务器不是位于重定向器所在的计算机，则必须将以下文件复制到该计算机，并确保这些文件位于路径 (Windows) 或共享路径 (Unix) 中。您所需的文件视所使用的加密方式（如果有）而定。

   以下文件位置相对于 *install-dir*：

Setup	所需文件
ECC 加密	● Windows：*bin32\mlcecc11.dll* ● Unix：*lib32/libmlcecc11_r.so*
RSA 加密	● Windows：*bin32\mlcrsa11.dll* ● Unix：*lib32/libmlcrsa11_r.so*
经 FIPS 认可的 RSA 加密	● Windows：*bin32\mlcrsafips11.dll* 和 *bin32\sbgse2.dll* ● Unix：*lib32/libmlcrsafips11_r.so* 和 *libsbgse2_r.so*

4. 按如下说明更新 Apache Web 服务器的配置文件 *httpd.conf*。
   - 在 LoadModule 区段，如果是 Windows 平台，请添加以下行：

     ```
     LoadModule iaredirect_module modules/mod_iaredirect.dll
     ```

     如果是 Solaris 和 Linux 平台，请添加以下行：

     ```
     LoadModule iaredirect_module modules/mod_iaredirect.so
     ```

Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

- 在该文件中添加以下区段：

  ```
  <Location /iaredirect/ml>
     SetHandler iaredirect-handler
     iaredirectorConfigFile location/redirector.config
  </Location>
  ```

  其中 */iaredirect/ml* 是在调用重定向器时使用的相对 URL 路径，*location* 是 *redirector.config* 所在的目录。

- 如果是在 Solaris 或 Linux 上使用 Apache，则可能还需要将以下可选指令添加到刚创建的 <Location> 区段中：

  - **MaxSyncUsers *number*** 通过重定向器进行同步的 MobiLink 用户数量上限。此数量用于给重定向器分配必要的资源。此数量不能小于 60，其缺省值是 1000。请仅在缺省用户数量小于实际数量时再更改此设置。

  - **ShmemDiagnosis on|off** 如果设置为 on，则允许对内存资源进行调试。缺省值是 off。

5. 为便于调试，最好增加重定向器输出的记录信息量。要执行此操作，请修改 *httpd.conf* 中的 LogLevel 指令，将其设置为 **LogLevel info**。日志级别可以是（按详细程度由高到低的顺序）：debug、info、notice、warn、error、crit、alert 和 emerg。

示例

以下是 *httpd.conf* 一些区段的示例，这些示例将 Apache Web 服务器配置为把请求路由到 MobiLink 服务器。此示例适用于 Windows。如果使用的是 Unix 和 Linux，请将 *mod_iaredirect.dll* 更改为 *mod_iaredirect.so*。

```
LoadModule iaredirect_module modules/mod_iaredirect.dll
...

<Location /iaredirect/ml>
    SetHandler iaredirect-handler
    iaredirectorConfigFile c:/redirector.config
</Location>
```

♦ 测试您的配置

1. 使用以下语法调用重定向器：

   ```
   http://host:port/iaredirect/ml/
   ```

   其中 *iaredirect/ml* 是您在 *httpd.conf* 的 <Location> 标记内指定的相对 URL 路径。

2. 检查日志文件以了解重定向器是否记录了请求。日志文件的缺省位置是 *$APACHE_HOME/logs/error.log*。

   注意：此测试不会建立到 MobiLink 服务器的连接。

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

M-Business Anywhere 重定向器（不建议使用）

注意
现已不建议使用重定向器。可以使用中继服务器代替重定向器。请参见“中继服务器”第 223 页。

以下是在 Windows、Solaris 或 Linux 上设置 M-Business Anywhere 的说明。

有关所支持版本的信息，请参见 http://www.sybase.com/detail?id=1062632。

♦ 配置 M-Business Anywhere 重定向器

1. 完成“配置重定向器属性（适用于不支持服务器组的重定向器）”一节第 258 页中的步骤。
2. 将文件 *mod_iaredirect.dll* 或 *mod_iaredirect.so* 复制到 Web 服务器所在计算机上的 M-Business Anywhere *\bin* 目录。此文件位于 *install-dir\MobiLink\redirector\MBusinessAnywhere* 中。
3. 如果 Web 服务器不是位于重定向器所在的计算机，则必须将以下文件复制到该计算机，并确保这些文件位于路径 (Windows) 或共享路径 (Unix) 中。您所需的文件视所使用的加密方式（如果有）而定。

   以下文件位置相对于 *install-dir*：

Setup	所需文件
ECC 加密	● Windows：*bin32\mlcecc11.dll* ● Unix：*lib32/libmlcecc11_r.so*
RSA 加密	● Windows：*bin32\mlcrsa11.dll* ● Unix：*lib32/libmlcrsa11_r.so*
经 FIPS 认可的 RSA 加密	● Windows：*bin32\mlcrsafips11.dll* 和 *bin32\sbgse2.dll* ● Unix：*lib32/libmlcrsafips11_r.so* 和 *libsbgse2_r.so*

4. 如果使用的是 Windows，请按以下步骤更新 M-Business Anywhere 的 Web 服务器的配置文件 *sync.conf.default*：
   - 在 LoadModule 区段中添加以下行：

     ```
     LoadModule iaredirect_module @@ServerRoot@@/bin/mod_iaredirect.dll
     ```

   - 在 SyncLoadFile 部分，添加以下行：

     ```
     SyncLoadFile @@ServerRoot@@/bin/mod_iaredirect.dll
     ```

   - 在该文件中添加以下区段：

     ```
     <Location /iaredirect/ml>
        SetHandler iaredirect-handler
        iaredirectorConfigFile @@ServerRoot@@/conf/redirector.config
     </Location>
     ```

   - 运行 *setup_defaults.bat* 命令文件，将这些更改构建到 *sync.conf* 文件中。

5. 如果使用的是 Solaris 和 Linux，请按以下步骤更新 M-Business Anywhere Web 服务器的配置文件 *sync.conf*：
   - 在 LoadModule 区段中添加以下行：

     ```
     LoadModule iaredirect_module path/bin/mod_iaredirect.so
     ```

     其中 *path* 是 M-Business Anywhere *bin* 目录的位置。
   - 在该文件中添加以下区段：

     ```
     <Location /iaredirect/ml>
        SetHandler iaredirect-handler
        iaredirectorConfigFile location/redirector.config
     </Location>
     ```

     其中 */iaredirect/ml* 是在调用重定向器时使用的相对 URL 路径，*location* 是 *redirector.config* 所在的目录。
   - 您可能还需要将以下可选指令添加到刚创建的 <Location> 区段中：
     - **MaxSyncUsers *n*** 通过重定向器进行同步的 MobiLink 用户数量上限。此数量用于给重定向器分配必要的资源。此数量不能小于 60，其缺省值是 1000。请仅在缺省用户数量小于实际数量时再更改此设置。
     - **ShmemDiagnosis on|off** 如果设置为 on，则允许对内存资源进行调试。缺省值是 off。
6. 为便于调试，最好增加重定向器输出的记录信息量。要执行此操作，请修改 *sync.conf* 中的 LogLevel 指令，将其设置为 **LogLevel info**。日志级别可以是（按详细程度由高到低的顺序）：debug、info、notice、warn、error、crit、alert 和 emerg。
7. 重新启动 M-Business 同步服务器，以使更改生效。

示例

以下是 *sync.conf* 区段的示例，这些示例将 M-Business Anywhere Web 服务器配置为把请求路由到 MobiLink 服务器。

此示例在 Windows 上运行：

```
LoadModule iaredirect_module "c:\program files\M-Business Anywhere/bin/
mod_iaredirect.dll"
...
SyncLoadFile "c:\program files\M-Business Anywhere/bin/mod_iaredirect.dll"
...
<Location \iaredirect\ml>
    SetHandler iaredirect-handler
    iaredirectorConfigFile "c:\AvantGoServer\conf/redirector.config"
</Location>
```

以下示例适用于 Unix 和 Linux：

```
LoadModule iaredirect_module modules/mod_iaredirect.so
...
<Location /iaredirect/ml>
    SetHandler iaredirect-handler
    iaredirectorConfigFile "/redirector.config"
</Location>
```

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

♦ **测试您的配置**

1. 使用以下语法调用重定向器：

   ```
   http://host:port/iaredirect/ml/
   ```

   其中 *iaredirect* 是您在 *sync.conf* 的 <Location> 标记内指定的路径。

2. 检查日志文件 *sync_access.log* 和 *sync_error.log* 以验证重定向器记录了请求。

   注意：此测试不会建立到 MobiLink 服务器的连接。

MobiLink 基于文件的下载

目录

基于文件的下载简介 276
建立基于文件的下载 277
校验检查 280
基于文件的下载示例 283

基于文件的下载简介

基于文件的下载是另一种将数据下载到 SQL Anywhere 远程数据库的方法：可以将下载作为文件分发，从而实现脱机分发同步更改。这样您只需创建一次文件即可将其分配到多个远程数据库。

使用基于文件的下载，您可以将下载同步更改保存到文件中，然后可以通过任何文件传送方式将其传送到 SQL Anywhere 远程数据库。例如，您可以：

- 通过卫星多路广播来广播数据
- 使用 Sybase Afaria 应用更新
- 将文件通过电子邮件或 FTP 发送给用户

您可以选择要接收该文件的用户。基于文件的下载可以保持完整同步的完整性，包括冲突检测和解决办法。可以通过对文件应用第三方加密来确保文件的安全。

何时使用

基于文件的下载在下列情况下非常有用：统一数据库上有大量数据更改，但远程数据库不经常更新数据或根本不进行任何更新。例如，价格列表、产品列表和代码表。

基于文件的下载在以下情况下没有用处：下载的数据在远程数据库中经常更新，或者您在运行频繁的仅上载同步。在这些情况下，由于应用下载文件时执行的完整性检查的原因，远程站点可能无法应用下载文件。

基于文件的下载目前仅可用于 SQL Anywhere 远程数据库。

仅下载发布

在大多数情况下，基于文件的下载应使用仅下载发布。仅当您需要使用与执行基于文件的下载所用同样的发布执行上载时，才使用常规发布。

请参见“仅下载发布”一节 《MobiLink - 客户端管理》。

当您使用常规发布时，不能将基于文件的下载作为更新远程数据库的唯一方法。在这种情况下，您仍然需要定期执行完整同步或仅上载同步。推进日志偏移和维护日志文件需要完整同步或仅上载同步，如果不进行这样的操作，日志文件将变得很大，并降低同步速度。从错误中恢复可能也需要完整同步。

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

建立基于文件的下载

以下步骤概述建立基于文件的下载所需完成的任务（假设已建立了 MobiLink 同步）。

在大多数情况下，基于文件的下载应使用仅下载发布。仅当您需要使用与执行基于文件的下载所用同样的发布执行上载时，才使用常规发布。

♦ **建立基于文件的下载概述**

1. 创建一个文件定义数据库。

   请参见“创建文件定义数据库”一节第 277 页。

2. 在统一数据库中，创建具有新脚本版本的脚本。

   请参见“在统一数据库上更改”一节第 277 页。

3. 创建一个下载文件。

   请参见“创建下载文件”一节第 278 页。

4. 应用下载文件。

   请参见“对新的远程数据库进行同步”一节第 278 页。

其它入门资源

- “基于文件的下载示例”一节第 283 页

创建文件定义数据库

要建立基于文件的下载，可创建一个**文件定义数据库**。它是一种 SQL Anywhere 数据库，与远程数据库具有相同的同步表和发布。它可以位于任意位置。此数据库不包含任何数据或状态信息。它无需备份，也无需维护，事实上，您可以根据需要删除它并重新创建。

文件定义数据库必须包含以下内容：

- 与远程数据库相同的发布、在发布中使用的表和列、这些表与列的外键关系和约束以及这些外键关系所需的表。
- MobiLink 用户名，用来标识将要应用下载文件的远程数据库组。您必须在同步脚本中使用此 MobiLink 组用户名，以标识远程数据库组。

在统一数据库上更改

在统一数据库上，为基于文件的下载创建一个新的脚本版本，并在其中实现现有同步系统所需的所有脚本。无需上载脚本。此脚本版本仅用于基于文件的下载。对于此脚本版本，将 MobiLink 用户名用作参数的所有脚本将转而使用引用一组远程数据库的 MobiLink 用户名。此用户名即是在文件定义数据库中定义的用户名。

为每个已定义的脚本版本，实现 begin_publication 脚本。

Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

对于基于时间戳的下载，为每个脚本版本实现 modify_last_download_timestamp 脚本。如何实现此脚本取决于希望在每个下载文件中发送多少数据。例如，一种方法是使用该组中所有用户上次成功下载的最早时间。切记，传递给此脚本的 ml_username 参数实际上是组名。

另请参见

- “脚本版本”一节第 303 页
- “begin_publication 连接事件”一节第 355 页
- “modify_last_download_timestamp 连接事件”一节第 430 页

创建下载文件

下载文件包含要同步的数据。要创建下载文件，请如上所述建立文件定义数据库和统一数据库。使用 -bc 选项运行 dbmlsync 并提供扩展名为 *.df* 的文件名。例如，

```
dbmlsync -c "uid=DBA;pwd=sql;eng=fbdl_eng;dbf=fdef.db" -v+
   -e "sv=filebased" -bc file1.df
```

您也可以选择在创建下载文件时指定以下选项：

- **-be 选项** 使用 -be 会向可在远处数据库通过 sp_hook_dbmlsync_validate_download_file 存储过程访问的下载文件添加字符串。

  请参见“-be 选项”一节 《MobiLink - 客户端管理》
  和“sp_hook_dbmlsync_validate_download_file”一节 《MobiLink - 客户端管理》。

- **-bg 选项** 可以使用 -bg 选项创建一个下载文件，供从未进行过同步的远程数据库使用。

对新的远程数据库进行同步

如果要将下载文件应用到从未使用 MobiLink 同步过的远程数据库，则在应用下载文件前，需要在远程数据库上执行常规同步，或者在创建下载文件时使用 dbmlsync -bg 选项。

对于基于时间戳的同步，进行这两种操作中的任一种都会下载数据的初始快照。对于基于时间戳和基于快照的同步，此步骤都会将世代号设置为 begin_publication 脚本在统一数据库上生成的值。

执行常规同步

通过执行不使用下载文件的同步，可以准备远程数据库，以接收下载文件。

使用 -bg 选项

或者，您可以使用 -bg 选项创建一个下载文件，以用于尚未同步的远程数据库。应用此初始下载文件来准备远程数据库，以便进行基于文件的同步。

- **快照下载** 如果执行快照下载，则初始下载文件只需设置世代号。您可以选择在此文件中包含数据的初始快照，但由于每个快照下载都包含所有数据且并不依赖于先前的下载，所以并不需要这样做。

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

对于快照下载，使用 -bg 选项更直接了当。只要在创建下载文件时在 dbmlsync 命令行上指定 -bg 即可。您可以通过用于后续下载文件的同一脚本版本来创建初始下载文件。

- **基于时间戳的下载** 如果要执行基于时间戳的下载，初始下载必须在远程数据库上设置世代号，并包括数据的快照。利用基于时间戳的下载，每个下载都构建在先前的下载的基础上。每个下载文件都包含一个上次下载时间戳。自文件的上次下载时间戳后统一数据库上更改的所有行都包括在此文件中。要应用文件，远程数据库必须已经接收到文件的上次下载时间戳之前发生的所有更改。这一点可通过以下方法来确认：检查文件的上次下载时间戳是否大于或等于远程数据库的上次下载时间戳（远程数据库从统一数据库接收到所有更改时的时间）。

  远程数据库必须先接收 1900 年 1 月 1 日后到文件上次下载时间戳之前更改的所有数据，然后才可以应用其第一个常规下载文件。通过 -bg 选项创建的初始下载文件必须包含此数据。选择此数据最简单的方法是另外创建一个脚本版本：该脚本版本使用与常规的基于文件的同步脚本版本相同的 download_cursor 脚本版本，但没有 modify_last_download_timestamp 脚本。如果未定义任何 modify_last_download_timestamp 脚本，则基于文件的下载的上次下载时间戳缺省为 1900 年 1 月 1 日。

如果将使用 -bg 选项构建的下载文件应用到已经同步的远程数据库，-bg 选项会使远程数据库上的世代号更新为创建下载文件时统一数据库上的值。世代号是为了确保在恢复丢失或损坏的统一数据库时上载过程执行完毕后，再执行进一步的基于文件的下载，而以上操作会使此目标落空。

请参见“MobiLink 世代号”一节第 281 页。

校验检查

将下载文件应用到远程数据库之前，dbmlsync 会进行几个操作，以确保同步有效。

- dbmlsync 会检查下载文件，确保用于创建它的文件定义数据库：
  - 与远程数据库具有相同的发布
  - 使用的表和列与发布中所用的相同
  - 与这些表和列具有相同的外键关系和约束
- dbmlsync 会检查尚未从远程数据库上载的发布中是否有任何数据。如果有，则将不应用下载文件，因为应用该下载文件会导致未提交的上载数据丢失。
- dbmlsync 会检查下载文件的上次下载时间戳、下一个上次下载时间戳以及创建时间，以确保：
  - 远程数据库上较新的数据不会被下载文件中包含的较旧的数据所覆盖。
  - 如果应用下载文件会使远程数据库丢失统一数据库上发生的一些更改，则不会应用下载文件。如果远程数据库未应用先前的基于文件的下载，则可能发生这种情况。

    请参见“自动校验”一节第 280 页。
- dbmlsync 还可以检查远程数据库中的世代号，确保它与下载文件中的世代号匹配。

  请参见“MobiLink 世代号”一节第 281 页。
- 您可以选择使用 sp_hook_dbmlsync_validate_download_file 存储过程，创建自定义的校验逻辑。

  有关详细信息，请参见“自定义校验”一节第 282 页。

自动校验

应用下载文件前，dbmlsync 会执行特殊的检查，检查上次下载时间戳、下一个上次下载时间戳、下载文件创建时间和事务日志。

上次下载时间戳和下一个上次下载时间戳

每个下载文件都包含要下载的所有更改，这些更改在文件的上次下载时间戳和下一个上次下载时间戳之间发生在统一数据库上。两个时间都是以统一数据库上的时间表示。缺省情况下，文件的上次下载时间为 1900 年 1 月 1 日上午 12:00，文件的下一个上次下载时间戳为下载文件的创建时间。在统一数据库上实施 modify_last_download_timestamp 和 modify_next_last_download_timestamp 脚本时，这些缺省值会被替换。

仅当文件的上次下载时间戳小于或等于远程数据库的上次下载时间戳时，远程站点才可以应用下载文件。这可确保远程数据库永远不会丢失统一数据库上发生的操作。通常，当基于文件的下载因该检查而失败时，远程数据库即已丢失一个或多个下载文件。这种情况可以通过应用丢失的下载文件来更正，也可通过执行完整同步或仅下载同步来更正。

另外，仅当文件的下一个上次下载时间戳大于远程数据库的上次下载时间戳时，远程站点才可以应用下载文件。远程数据库的上次下载时间戳即远程数据库已经接收到所有即将下载的更改的时间（统一数据库上的）。每当远程数据库成功应用下载（常规的或基于文件的）时，远程数据库的上

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

次下载时间都会更新。此检查可确保在已下载更新的数据后不再应用下载文件。发生此问题的常见情形是错误应用下载文件。例如，假设创建了下载文件 *F1.df*，稍后又创建了 *F2.df*。此检查可确保不会在应用 *F2.df* 之后应用 *F1.df*，因为那样会使 *F2.df* 中较新的数据被 *F1.df* 中较旧的数据覆盖。

当基于文件的下载操作因下一个上次下载时间戳的原因而失败时，除了删除该文件外，不需要进行其它任何操作。接收到新的文件后，同步即告成功。

创建时间

下载文件的创建时间表示文件开始创建时统一数据库上的时间。下载文件仅在其创建时间大于远程数据库的上次上载时间时才可以应用。远程数据库的上次上载时间是在统一数据库中远程数据库上次成功上载的提交时间。此检查可确保在下载创建后上载的数据（要比下载的数据新）不会被下载文件中较旧的数据覆盖。

当下载文件因这一检查而遭到拒绝时，不需要执行任何操作。远程站点应可以应用下一个下载文件。

当因为 dbmlsync 向 MobiLink 服务器发送了上载后未得到确认而导致上载失败时，远程数据库的上次上载时间可能不正确。在这种情况下，创建时间检查将无法执行，远程数据库将无法应用下载文件，直到完成常规同步为止。

事务日志

应用下载文件之前，dbmlsync 会扫描远程数据库的事务日志，并构建必须上载的所有更改的列表。仅当下载文件不包含任何会影响具有必须上载的更改的行时，Dbmlsync 才会应用该下载文件。

MobiLink 世代号

世代号提供了一种机制，可强制远程数据库在应用更多下载文件之前上载数据。当由于统一数据库上的问题造成数据丢失而必须从远程数据库恢复丢失的数据时，这特别有用。

在远程数据库上，会为每个预订自动维护一个独立的世代号。在统一数据库上，每个预订的世代号取决于 begin_publication 脚本。每当远程数据库成功执行上载时，它都会以统一数据库中 begin_publication 脚本所设置的值更新远程世代号。

每当创建下载文件时，begin_publication 脚本所设置的世代号都会存储在下载文件中。仅当下载文件中的世代号等于远程数据库中存储的世代号时，远程站点才应用该文件。

注意
每当基于文件的下载的 begin_publication 脚本所产生的世代号发生更改时，远程数据库必须执行成功的上载才能应用新的下载文件。

sp_hook_dbmlsync_validate_download_file 存储过程可用于替换缺省世代号检查。

有关管理 MobiLink 世代号的详细信息，请参见：

- “begin_publication 连接事件”一节第 355 页
- “end_publication 连接事件”一节第 394 页
- “sp_hook_dbmlsync_validate_download_file”一节 《MobiLink - 客户端管理》

自定义校验

您可以创建自定义校验逻辑，以确定某下载文件是否应该应用到远程数据库。为此，您可以利用 sp_hook_dbmlsync_validate_download_file 存储过程。利用此存储过程，您可以拒绝下载文件并替换缺省世代号检查。

您可以使用 dbmlsync -be 选项在文件中嵌入字符串。创建下载文件时，可以对文件定义数据库使用 -be 选项。此字符串将通过 #hook_dict 表传送到 sp_hook_dbmlsync_validate_download_file，并可用于校验逻辑。

有关详细信息，请参见“sp_hook_dbmlsync_validate_download_file”一节 《MobiLink - 客户端管理》。

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

基于文件的下载示例

本节包含两个示例。每个示例都使用只有一个表的统一数据库建立基于文件的下载同步。第一个是简单的快照示例，第二个是稍微复杂一些的基于时间戳的示例。

快照示例

此示例将实现基于文件的下载以便进行快照同步。它将建立基于文件的下载所要求的三个数据库，然后演示如何下载数据。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。

为示例创建数据库

以下命令将创建示例中使用的三个数据库：统一数据库、远程数据库和文件定义数据库。

```
dbinit scons.db
dbinit sremote.db
dbinit sfdef.db
```

以下命令将启动这三个数据库，并创建 MobiLink 连接到统一数据库时使用的数据源名。

```
dbeng11 -n sfdef_eng sfdef.db
dbeng11 -n scons_eng scons.db
dbeng11 -n sremote_eng sremote.db
dbdsn -y -w fbd_demo -c "eng=scons_eng;dbf=scons.db;uid=DBA;
   pwd=sql;astart=off;astop=off"
```

打开 Interactive SQL，连接到 *scons.db*，并运行 MobiLink 安装脚本。例如：

```
read "c:\Program Files\SQL Anywhere 11\MobiLink\setup\syncsa.sql"
```

启动 MobiLink 服务器：

```
start mlsrv11 -v+ -c "dsn=fbd_demo" -zu+ -ot scons.txt
```

建立快照示例统一数据库

在此示例中，统一数据库包含一个名为 T1 的表。连接到统一数据库后，您可以运行以下 SQL 来创建表 T1：

```
CREATE TABLE T1 (
 pk INTEGER PRIMARY KEY,
 c1 INTEGER
);
```

以下代码将创建一个名为 filebased 的脚本版本，并为该脚本版本创建一个下载脚本。

```
CALL ml_add_table_script( 'filebased',
  'T1', 'download_cursor',
  'SELECT pk, c1 FROM T1' );
```

以下代码将创建一个名为 normal 的脚本版本，并为该脚本版本创建上载和下载脚本。

```
CALL ml_add_table_script ( 'normal', 'T1',
  'upload_insert',
  'INSERT INTO T1 VALUES ({ml r.pk}, {ml r.c1})');
```

```
CALL ml_add_table_script( 'normal', 'T1',
  'upload_update',
  'UPDATE T1 SET c1 = {ml r.c1} WHERE pk = {ml r.pk} ' );

CALL ml_add_table_script( 'normal', 'T1',
 'upload_delete',
 'DELETE FROM T1 WHERE pk = {ml r.pk}' );

CALL ml_add_table_script( 'normal', 'T1',
  'download_cursor',
  'SELECT pk, c1 FROM T1' );

COMMIT;
```

以下命令将创建存储过程 begin_pub，并指定 begin_pub 为 "normal" 和 "filebased" 脚本版本的 begin_publication 脚本：

```
CREATE PROCEDURE begin_pub (
       INOUT generation_num integer,
       IN    username       varchar(128),
       IN    pubname        varchar(128) )
BEGIN
   SET generation_num=1;
END;

CALL ml_add_connection_script(
  'filebased',
  'begin_publication',
  '{ call begin_pub(
      {ml s.generation_number},
      {ml s.username},
      {ml s.publication_name} ) }' );

CALL ml_add_connection_script( 'normal',
  'begin_publication',
  '{ call begin_pub(
      {ml s.generation_number},
      {ml s.username},
      {ml s.publication_name} ) }' );
```

创建快照示例远程数据库

在此示例中，远程数据库也包含一个名为 T1 的表。连接到远程数据库并运行下列 SQL 以创建表 T1、一个名为 P1 的发布和一个名为 U1 的用户。该 SQL 语句还会为 U1 创建 P1 的预订。

```
CREATE TABLE T1 (
 pk INTEGER PRIMARY KEY,
 c1 INTEGER
);

CREATE PUBLICATION P1 (
    TABLE T1
);

CREATE SYNCHRONIZATION USER U1;

CREATE SYNCHRONIZATION SUBSCRIPTION
TO P1
FOR U1;
```

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

以下代码将创建 sp_hook_dbmlsync_validate_download_file 挂接，以在远程数据库中实现用户定义的校验逻辑：

```
CREATE PROCEDURE sp_hook_dbmlsync_validate_download_file()
BEGIN
    DECLARE udata varchar(256);
SELECT value
    INTO udata
    FROM #hook_dict
    WHERE name = 'user data';
    IF udata <> 'ok' THEN
    UPDATE #hook_dict
    SET value = 'FALSE'
    WHERE name = 'apply file';
    END IF;
END
```

创建快照示例文件定义数据库

在使用基于文件的下载的 MobiLink 系统中，需要文件定义数据库。此数据库与基于文件的下载所更新的远程数据库具有相同的模式，且不包含任何数据或状态信息。文件定义数据库仅用于定义即将包含在下载文件中的数据的结构。一个文件定义数据库可用于多个远程数据库组，每个组都由其自己的 MobiLink 组用户名定义。

以下代码将为此示例定义文件定义数据库。它将创建一个与远程数据库相同的模式，还将创建：

- 一个名为 P1 的发布，以发布表 T1 的所有行。文件定义数据库和远程数据库必须使用相同的发布名称。
- 一个名为 G1 的 MobiLink 用户。此用户代表即将在基于文件的下载中更新的所有远程数据库。
- 一个发布的预订。

您必须先连接到 sfdef.db 然后才能运行此代码。

```
CREATE TABLE T1 (
 pk INTEGER PRIMARY KEY,
 c1 INTEGER
);

CREATE PUBLICATION P1 (
    TABLE T1
);

CREATE SYNCHRONIZATION USER G1;

CREATE SYNCHRONIZATION SUBSCRIPTION
TO P1
FOR G1;
```

为初始同步做准备

要对新的远程数据库进行准备以便应用下载文件，您需要执行常规同步或使用 dbmlsync -bg 选项创建下载文件。此示例说明如何通过执行常规同步来初始化新的远程数据库。

您可以使用先前创建的名为 normal 的脚本版本对远程数据库进行初始同步：

```
dbmlsync -c "uid=DBA;pwd=sql;eng=sremote_eng;
  dbf=sremote.db" -v+ -e "sv=normal"
```

演示快照示例基于文件的下载

连接到统一数据库，并插入一些由基于文件的下载进行同步的数据（例如以下数据）：

```
INSERT INTO T1 VALUES( 1, 1 );
INSERT INTO T1 VALUES( 2, 4 );
INSERT INTO T1 VALUES( 3, 9 );
COMMIT;
```

以下命令必须在具有文件定义数据库的计算机上运行。此命令执行以下操作：

- dbmlsync -bc 选项将创建下载文件，并将其命名为 *file1.df*。
- -be 选项将把字符串 "OK" 纳入将要由 sp_dbmlsync_validate_download_file 挂接访问的下载文件。

```
dbmlsync -c
   "uid=DBA;pwd=sql;eng=sfdef_eng;dbf=sfdef.db"
   -v+ -e "sv=filebased" -bc file1.df -be ok -ot fdef.txt
```

要应用该下载文件，可在远程数据库上使用 -ba 选项运行 dbmlsync，并提供要应用的下载文件的名称：

```
dbmlsync -c "uid=DBA;pwd=sql;eng=sremote_eng;
   dbf=sremote.db" -v+ -ba file1.df -ot remote.txt
```

现在已将更改应用到远程数据库。打开 Interactive SQL，连接到远程数据库，然后运行以下 SQL 语句，检查远程数据库中是否有这些数据：

```
SELECT * FROM T1
```

清理快照示例

以下命令停止所有三个数据库服务器并消除文件。

```
del file1.df
mlstop -h -w
dbstop -y -c "eng=sfdef_eng; uid=DBA; pwd=sql"
dbstop -y -c "eng=scons_eng; uid=DBA; pwd=sql"
dbstop -y -c "eng=sremote_eng; uid=DBA; pwd=sql"
dberase -y sfdef.db
dberase -y scons.db
dberase -y sremote.db
```

基于时间戳的示例

此示例将实现基于文件的下载以便进行基于时间戳的同步。它将建立三个数据库，然后演示如何基于文件下载数据。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。

为示例创建数据库

以下命令将创建示例中使用的三个数据库：统一数据库、远程数据库和文件定义数据库。

```
dbinit tcons.db
dbinit tremote.db
dbinit tfdef.db
```

以下命令将启动这三个数据库，并创建 MobiLink 连接到统一数据库时使用的数据源名。

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

```
dbeng11 -n tfdef_eng tfdef.db
dbeng11 -n tcons_eng tcons.db
dbeng11 -n tremote_eng tremote.db
dbdsn -y -w tfbd_demo -c "eng=tcons_eng;dbf=tcons.db;uid=DBA;
   pwd=sql;astart=off;astop=off"
```

打开 Interactive SQL，连接到 *tcons.db*，然后运行 MobiLink 安装脚本。例如：

```
read "c:\Program Files\SQL Anywhere 11\MobiLink\setup\syncsa.sql"
```

启动 MobiLink 服务器：

```
start mlsrv11 -v+ -c "dsn=tfbd_demo" -zu+ -ot tcons.txt
```

建立时间戳示例统一数据库

在此示例中，统一数据库包含一个名为 T1 的表。连接到统一数据库后，您可以运行以下代码来创建 T1：

```
CREATE TABLE T1 (
 pk  INTEGER PRIMARY KEY,
 c1  INTEGER,
 last_modified TIMESTAMP DEFAULT TIMESTAMP
);
```

以下代码定义名为 normal 的脚本版本，该脚本版本中包含最少量的脚本。此脚本版本将被用于不使用基于文件的下载的同步。

```
CALL ml_add_table_script( 'normal', 'T1',
   'upload_insert',
   'INSERT INTO T1( pk, c1) VALUES( {ml r.pk}, {ml r.c1} )' );

CALL ml_add_table_script( 'normal', 'T1',
   'upload_update',
   'UPDATE T1 SET c1 = {ml r.c1} WHERE pk = {ml r.pk} ' );

CALL ml_add_table_script( 'normal', 'T1',
   'upload_delete',
   'DELETE FROM T1 WHERE pk = {ml r.pk}' );

CALL ml_add_table_script( 'normal', 'T1',
   'download_cursor',
   'SELECT pk, c1 FROM T1
      WHERE last_modified >= {ml s.last_table_download}' );
```

以下代码将所有预订的世代号设置为 1。使用世代号是个很好的做法，可用于因统一数据库丢失或损坏而需要强制实施上载的情形。

```
CREATE PROCEDURE begin_pub (
       INOUT generation_num integer,
       IN    username       varchar(128),
       IN    pubname        varchar(128) )
BEGIN
 SET generation_num = 1;
END;

CALL ml_add_connection_script( 'normal',
    'begin_publication',
    '{ call begin_pub(
        {ml s.generation_number},
        {ml s.username},
```

```
        {ml s.publication_name},
        {ml s.last_publication_upload},
        {ml s.last_publication_download} ) }' );

COMMIT;
```

以下代码将定义名为 filebased 的脚本版本。此脚本版本将被用于创建基于文件的下载。

```
CALL ml_add_connection_script( 'filebased',
    'begin_publication',
    '{ call begin_pub(
        {ml s.generation_number},
        {ml s.username},
        {ml s.publication_name} ) }' );

CALL ml_add_table_script( 'filebased', 'T1',
    'download_cursor',
    'SELECT pk, c1 FROM T1
        WHERE last_modified >= {ml s.last_table_download}' );
```

以下代码设置上次下载时间，以便将过去五天中发生的所有更改包含在下载文件中。任何缺少过去五天里创建的所有下载文件的远程数据库必须先执行常规同步，然后才能应用其它基于文件的下载。

```
CREATE PROCEDURE ModifyLastDownloadTimestamp(
       INOUT last_download_timestamp TIMESTAMP,
       IN    ml_username             VARCHAR(128) )
BEGIN
    SELECT dateadd( day, -5, CURRENT TIMESTAMP )
    INTO last_download_timestamp;
END;

CALL ml_add_connection_script( 'filebased',
   'modify_last_download_timestamp',
   'CALL ModifyLastDownloadTimestamp(
      {ml s.last_download}, {ml s.username} )' );

COMMIT;
```

创建时间戳示例远程数据库

在此示例中，远程数据库也包含一个名为 T1 的表。连接到远程数据库后，运行下列代码以创建表 T1、一个名为 P1 的发布和一个名为 U1 的用户。该代码还会为 U1 创建 P1 的预订。

```
CREATE TABLE T1 (
 pk INTEGER PRIMARY KEY,
 c1 INTEGER
);

CREATE PUBLICATION P1 (
    TABLE T1
);

CREATE SYNCHRONIZATION USER U1;

CREATE SYNCHRONIZATION SUBSCRIPTION
TO P1
FOR U1;
```

以下代码定义 sp_hook_dbmlsync_validate_download_file 存储过程。此存储过程可防止应用其中未嵌入字符串 "ok" 的下载文件。

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

```
CREATE PROCEDURE sp_hook_dbmlsync_validate_download_file()
BEGIN
    DECLARE udata varchar(256);

    SELECT value
    INTO udata
    FROM #hook_dict
    WHERE name = 'user data';

    IF udata <> 'ok' THEN
      UPDATE #hook_dict
      SET value = 'FALSE'
      WHERE name = 'apply file';
    END IF;
END
```

创建时间戳示例文件定义数据库

以下代码将为此时间戳示例定义文件定义数据库。它将创建一个表、一个发布、一个用户以及该用户的一个发布预订。

```
CREATE TABLE T1 (
 pk INTEGER PRIMARY KEY,
 c1 INTEGER
);

CREATE PUBLICATION P1 (
    TABLE T1
);

CREATE SYNCHRONIZATION USER G1;

CREATE SYNCHRONIZATION SUBSCRIPTION
TO P1
FOR G1;
```

为初始同步做准备

要对新的远程数据库进行准备以便应用下载文件，您需要执行常规同步或使用 dbmlsync -bg 选项创建下载文件。此示例说明如何使用 -bg。

以下代码将为统一数据库定义名为 filebased_init 的脚本版本。此脚本版本有一个 begin_publication 脚本。

```
CALL ml_add_table_script(
  'filebased_init', 'T1', 'download_cursor',
  'SELECT pk, c1 FROM T1' );

CALL ml_add_connection_script(
  'filebased_init',
  'begin_publication',
  '{ call begin_pub(
      {ml s.generation_number},
      {ml s.username},
      {ml s.publication_name} ) }' );

COMMIT;
```

以下两个命令行使用名为 filebased_init 的脚本版本和 -bg 选项，创建并应用初始下载文件。

```
dbmlsync -c "uid=DBA;pwd=sql;eng=tfdef_eng;dbf=tfdef.db"
  -v+ -e "sv=filebased_init" -bc tfile1.df -be ok -bg
```

```
  -ot tfdef1.txt

dbmlsync -c "uid=DBA;pwd=sql;eng=tremote_eng;dbf=tremote.db"
  -v+ -ba tfile1.df -ot tremote.txt
```

演示时间戳示例基于文件的下载

连接到统一数据库，并插入一些由基于文件的下载进行同步的数据（例如以下数据）：

```
INSERT INTO T1(pk, c1) VALUES( 1, 1 );
INSERT INTO T1(pk, c1) VALUES( 2, 4 );
INSERT INTO T1(pk, c1) VALUES( 3, 9 );
commit;
```

以下命令行创建包含新数据的下载文件。

```
dbmlsync -c
   "uid=DBA;pwd=sql;eng=tfdef_eng;dbf=tfdef.db"
   -v+ -e "sv=filebased" -bc tfile2.df -be ok -ot tfdef2.txt
```

以下命令行将下载文件应用到远程数据库。

```
dbmlsync -c "uid=DBA;pwd=sql;eng=tremote_eng;dbf=tremote.db"
   -v+ -ba tfile2.df -ot tfdef3.txt
```

现在已将更改应用到远程数据库。打开 Interactive SQL，连接到远程数据库，然后运行以下 SQL 语句，检查远程数据库中是否有这些数据：

```
SELECT * FROM T1
```

清理时间戳示例

以下命令停止所有三个数据库服务器，然后消除文件。

```
del tfile1.df
mlstop -h -w
dbstop -y -c "eng=tfdef_eng; uid=DBA; pwd=sql"
dbstop -y -c "eng=tcons_eng; uid=DBA; pwd=sql"
dbstop -y -c "eng=tremote_eng; uid=DBA; pwd=sql"
dberase -y tfdef.db
dberase -y tcons.db
dberase -y tremote.db
```

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

MobiLink 事件

本节介绍如何编写 MobiLink 事件的脚本。

编写同步脚本 293
同步事件 319

编写同步脚本

目录

同步脚本介绍 294
脚本和同步过程 297
脚本类型 298
脚本参数 299
脚本版本 303
必需的脚本 305
添加和删除脚本 306
编写用于上载行的脚本 309
编写用于下载行的脚本 312
编写用于处理错误的脚本 317

同步脚本介绍

您可通过编写同步脚本并在统一数据库的 MobiLink 系统表中存储或引用它们来控制同步过程。您可以使用 SQL、Java 或 .NET 编写脚本。

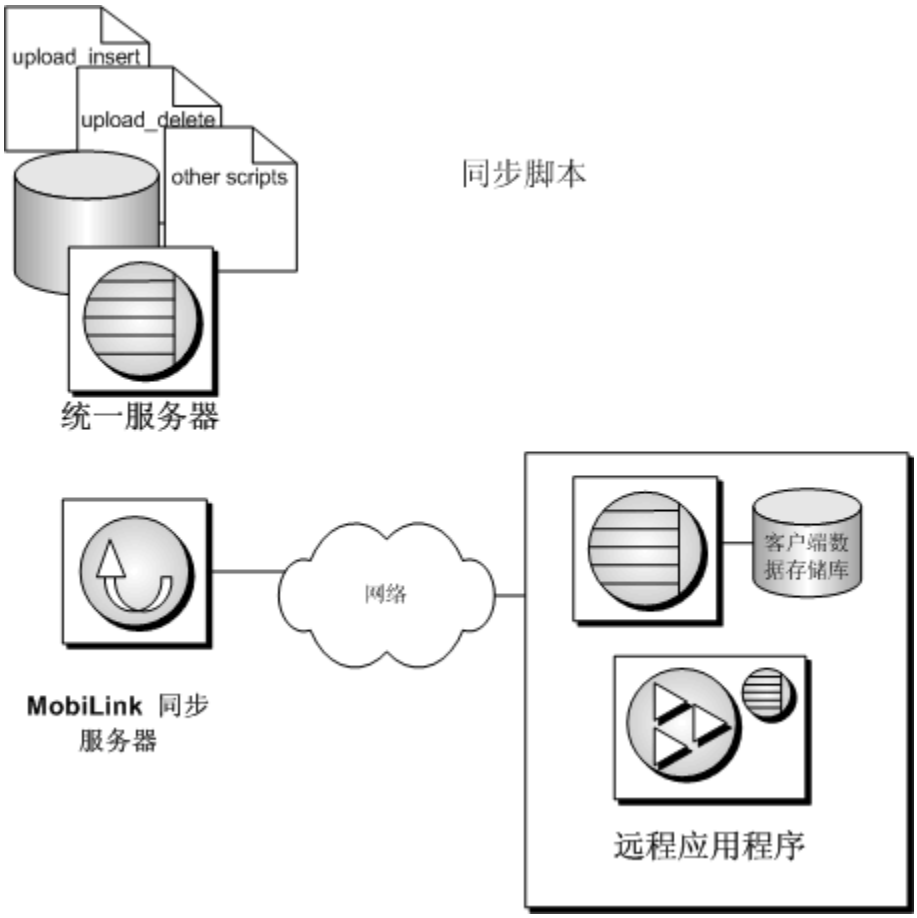
MobiLink 同步逻辑是使用同步脚本指定的。脚本定义：

- 如何将从远程数据库上载的数据应用到统一数据库
- 应从统一数据库下载哪些数据

脚本可以是单个语句，也可以是存储过程调用。您可在统一数据库中存储或引用它们。您可以使用 Sybase Central 或系统过程，将脚本添加到统一数据库中。

> 在 SQL 同步脚本中或从 SQL 同步脚本调用的过程或触发器中，不应有任何隐式或显式的提交或回退。SQL 脚本内的 COMMIT 或 ROLLBACK 语句会改变同步步骤的事务性质。如果您使用这两个语句，则在出现故障时 MobiLink 将无法保证数据的完整性。

在同步过程中，MobiLink 服务器将读取这些脚本，并对统一数据库执行这些脚本。



Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

同步过程由多个步骤组成。每个步骤都由唯一的事件来标识。可以通过编写与这些事件相关联的脚本来控制同步过程。只有在一旦某一特定的事件发生就必须执行某一操作的情况下，您才需要编写相应的脚本。与某个脚本相关联的事件发生时，MobiLink 服务器将执行该脚本。如果您没有为特定的事件定义脚本，MobiLink 服务器就会继续进行下一步骤。

例如，有一个事件是 begin_upload_rows。您可以编写一个脚本并将其与此事件进行关联。MobiLink 服务器将在首次需要此脚本时读取它，并在同步过程的上载阶段予以执行。如果您没有编写任何脚本，MobiLink 服务器将直接执行下一步，即处理上载的行。

某些被称为表脚本的脚本不仅可以与一个事件关联，而且可以与远程数据库中某一特定的表关联。MobiLink 服务器执行某些任务（例如下载行）的方式是按表逐个进行的。您可以将多个脚本与同一个事件关联，而将每个脚本与不同的应用程序表关联。或者，您也可以为某些应用程序表定义多个脚本，而不为另外一些应用程序表定义任何脚本。

有关事件的概述，请参见“同步过程”一节 《MobiLink - 入门》。

有关您可以编写的每个脚本的说明，请参见“同步事件”第 319 页。

您可以使用 SQL、Java 或 .NET 编写脚本。本章适用于所有类型的脚本，但着重讲述如何使用 SQL 编写同步脚本。

有关 SQL、Java 和 .NET 的说明和比较，请参见“用于编写服务器端同步逻辑的选项”一节 《MobiLink - 入门》。

有关使用 .NET 编写脚本的信息，请参见“使用 .NET 编写同步脚本”第 551 页。

有关使用 Java 编写脚本的信息，请参见“使用 Java 语言编写同步脚本”第 493 页。

有关如何实现同步脚本的信息，请参见“同步技术”第 119 页。

简单的同步脚本

MobiLink 提供了许多可以利用的事件，但是您不必为每个事件都提供脚本。在一个简单的同步模型中，您可能只需几个脚本。

通过将表中所有的行下载到每一个远程数据库，可以同步 CustDB 示例应用程序中的 ULProduct 表。在这种情况下，在远程数据库上不允许附加任何数据。您可使用单个脚本来实现这种简单的同步；在这种情况下，只有一个事件与该脚本相关联。

控制在每个同步期间下载的行的 MobiLink 事件称为 download_cursor 事件。游标脚本必须包含 SELECT 语句。MobiLink 服务器将使用这些查询来定义游标。如果使用 download_cursor 脚本，则游标将选择要下载到远程数据库中某个特定表的行。

在 CustDB 示例应用程序中，ULProduct 表有一个单独的 download_cursor 脚本，它包含以下查询：

```
SELECT prod_id, price, prod_name
FROM ULProduct
```

此查询生成一个结果集。组成此结果集的行将被下载到客户端。在这种情况下，该表中的所有行都将被下载。

因为此脚本与 download_cursor 事件和 ULProduct 表都关联（通过存储在统一数据库中），所以 MobiLink 服务器知道将这些行发送到 ULProduct 应用程序表中。通过 Sybase Central 可以进行这些关联。

在本例中，查询将从别名为 ULProduct 的统一表中选择数据。名称可以不匹配。实际上，您可以通过重新编写该查询，将数据从统一数据库中的任何一个表或多个表的任意组合下载到 ULProduct 应用程序表。

可以编写更为复杂的同步脚本。例如，您可以编写一种仅下载最近修改过的行的脚本，也可编写一种可以为每个远程数据库提供不同信息的脚本。

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

脚本和同步过程

每个脚本对应于同步过程中的一个特定事件。仅当某个操作必须发生时才编写脚本。所有不必要的事件都无需定义。

同步过程的两个主要部分是处理上载的信息和准备要下载的行。

MobiLink 服务器将在首次需要脚本时读取并准备脚本，且每个脚本仅读取和准备一次。然后，每当调用事件时，脚本都会被执行。

事件序列

有关完整的 MobiLink 事件序列的信息，请参见“MobiLink 事件概述”一节第 321 页。

有关上载处理的详细信息，请参见“编写用于上载行的脚本”一节第 309 页。

有关下载处理的详细信息，请参见“编写用于下载行的脚本”一节第 312 页。

注意

- MobiLink 技术允许多个客户端同时进行同步操作。在这种情况下，每个客户端都单独与统一数据库进行连接。
- begin_connection 和 end_connection 这两个事件不依赖于任何一个同步过程，因为一个连接可以处理许多同步请求。这些脚本没有参数。这些事件是连接级脚本的示例。
- 有些事件只能在每个同步过程中调用一次，而且只有一个参数。此参数即用户名称，用于唯一标识正在进行同步的 MobiLink 客户端。这些事件也是连接级脚本的示例。
- 对于每一个正在被同步的表，有些事件仅被调用一次。与此类事件相关联的脚本被称为表级别的脚本。这些脚本提供两个参数。第一个参数是在同步函数调用中提供的用户名，第二个参数是正在进行同步的远程数据库中的表的名称。

  尽管每个表都可以拥有自己的表脚本，但您也可以编写供多个表共享的表级别的脚本。
- 有些事件，如 begin_synchronization，在连接级别和表级别都会发生。您可以为这些事件提供连接级别和表级别的脚本。
- COMMIT 语句说明了如何将同步过程分解为不同的事务。
- 错误是在同步过程中的任何时刻都可能会发生的独立事件。错误是通过以下脚本进行处理的。

```
handle_error( error_code, error_message, user_name, table_name )
```

有关参考资料（包括每个脚本及其参数的详细信息），请参见“同步事件”第 319 页。

脚本类型

同步脚本可以应用于整个连接或特定的表。

- **连接级脚本** 此类脚本所执行的操作是连接特定或同步特定的，且不依赖于任何一个远程表。在实现更为复杂的同步模式时，此类脚本可以与其它脚本结合使用。

  请参见“连接脚本”一节第 298 页。

- **表级脚本** 此类脚本执行特定于一个同步和一个特定远程表的操作。在实现更为复杂的同步模式（如冲突解决）时，此类脚本可以与其它脚本结合使用。

  请参见“表脚本”一节第 298 页。

连接脚本

连接级脚本控制与特定表无关联的高级别事件。可以使用这些事件执行在每个同步期间所需的全局任务。

连接脚本可以控制以连接和断开连接为主的操作，也可以控制在发生同步级别的事件时执行的某些操作（例如，开始和结束上载或下载过程）。有些连接脚本有相关的表脚本。无论表是否正在同步，这些连接脚本总会被调用。

只有在发生特定事件时必须执行某一操作的情况下，才会需要编写连接级脚本。您可能只需为少数几个事件创建脚本。对所有事件的缺省操作都是 MobiLink 服务器不执行任何操作。有些简单的同步方案不需要连接脚本。

ml_global 脚本版本

为避免多次定义相同的脚本，您可以定义一次连接级脚本，然后重复使用它们。为此，您可定义一个名为 ml_global 的脚本版本。

请参见“ml_global 脚本版本”一节第 303 页。

表脚本

表脚本允许在发生与特定表的同步相关的特定事件时执行操作，例如开始与结束行的上载、解决冲突和选择要下载的行等。

一个给定表的同步脚本可以指向统一数据库中的任何一个表（或多个表的组合）。利用这一特性，您可以使用存储在一个或多个统一表中的数据填充某一特定的远程表，或者将从一个远程表中上载的数据存储到统一数据库的多个表中。

表名称无需匹配

远程数据库中的表名无需与统一数据库中的表名相匹配。MobiLink 服务器通过在 ml_table 系统表中查找远程表名，确定哪些脚本与表关联。

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

脚本参数

大多数同步脚本可以接收来自 MobiLink 服务器的参数。有关可在每个脚本中使用的参数的详细信息，请参见“同步事件”第 319 页。

可以采用以下两种方式之一在您的 SQL 脚本中指定参数：

- 问号
- 命名的脚本参数

用问号表示的脚本参数

用问号表示参数是一种 ODBC 约定。要在您的 MobiLink SQL 脚本中使用问号，请在脚本中为每个参数放置一个问号。MobiLink 服务器将用一个参数的值替换每个问号。它将按照参数在脚本定义中出现的顺序来替换这些值。

参数必须为“同步事件”第 319 页中指定的顺序。有些参数是可选的。只有在未指定任何后续参数时，参数才是可选的。例如，如果要使用参数 2，则必须使用参数 1。

命名的脚本参数

MobiLink 提供了命名参数，您可以使用这些参数代替脚本中的问号。命名参数具有以下优点：

- 命名参数允许您以任意顺序指定可用参数的任何子集。
- 除了输入/输出参数以外，您可以在一个脚本内多次指定相同的命名参数。
- 如果使用命名参数，则可以在脚本中指定远程 ID。这是在脚本中指定远程 ID 的唯一方法。
- 您可以创建自己的命名参数。请参见“用户定义的命名参数”一节第 300 页。

在一个脚本中不能混合使用命名参数和问号。

MobiLink 命名参数的类型共有四种。要指定命名参数，必须将其类型作为前缀，如下所示：

命名参数的类型	前缀	示例
系统参数。	s.	{ml s.remote_id}
行参数。（列名。如果列包含空格，则用双引号或方括号将其括起来。）	r.	{ml r.cust_id} {ml r."Column name"}
旧的行参数。（仅用于 upload_update 脚本之中，以指定前映像列值。如果列名包含空格，则用双引号或方括号将其括起来。）	o.	{ml o.cust_name} {ml o."Column name"}
验证参数。请参见“验证参数”一节第 301 页。	a.	{ml a.1}

命名参数的类型	前缀	示例
用户定义的参数。请参见“用户定义的命名参数”一节第 300 页。	ui.	{ml ui.varname}

要按名称引用脚本参数，应将该参数括在大括号内，并在前面加上 ml 前缀，如 **{ml** *parameter* **}**。例如，**{ml s.action_code}**。大括号表示是一种 ODBC 约定。

为方便起见，可以在大括号中包含更长的代码，只要这部分代码不包含与 MobiLink 脚本参数名称相同的任何模式名称即可。例如，以下两个 upload_insert 脚本均有效且等效：

```
INSERT INTO t ( id, c0 ) VALUES( {ml r.id}, {ml r.c0} )
```

和

```
{ml  INSERT INTO t ( id, c0 ) VALUES( r.id, r.c0 ) }
```

注意
要在 MobiLink [**创建同步模型向导**] 未在您的远程数据库中生成列时使用命名的行参数，您需要使用 ml_add_column 系统过程在统一数据库中存储列信息。请参见“ml_add_column 系统过程”一节第 626 页。

注释脚本参数

支持以下几种格式的注释：

- 双连字符前缀 (--)
- 双正斜线前缀 (//)
- 块注释 (/* */)

前两种格式可使系统忽略行末之前的脚本文本。第三种格式可使 /* 前缀和 */ 后缀之间的所有脚本文本都被忽略。块注释不能嵌套。

任何其它类型的服务商特定注释都无法识别，因此不应将其用于注释对命名参数的引用。

用户定义的命名参数

您也可以定义自己的参数。这些参数对于不允许使用用户定义变量的 RDBMS 特别有用。

用户定义的参数在首次引用时定义（并设置为空）。它们必须以 ui 和一个句点 (ui.) 开头。用户定义的参数的持续时间与一个同步的时间相同—每个同步启动时该参数都会设置为空。用户定义的参数是输入/输出参数。

对用户定义参数的典型应用是，在不必将状态信息存储到表中的情况下访问状态信息（可能需要复杂连接）。

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

示例

例如，假定您创建一个名为 MyCustomProc 的存储过程，该过程将名为 var1 的变量设置为 custom_value：

```
CREATE PROCEDURE MyCustomProc(
  IN username VARCHAR (128), INOUT var1 VARCHAR (128)
)
begin
 SET var1 = 'custom_value';
end
```

以下 begin_connection 脚本定义用户定义的参数 var1，并将其值设为 custom_value：

```
CALL ml_add_connection_script (
  'version1',
  'begin_synchronization',
  '{call MyCustomProc( {ml s.username}, {ml ui.var1} )}' );
```

以下 begin_upload 脚本引用 var1，其值为 custom_value：

```
CALL ml_add_connection_script (
  'version1',
  'begin_upload',
  'update SomeTable set some_column = 123 where some_other_column = {ml
ui.var1}' );
```

假定您另有一个名为 MyPFDProc 的存储过程，该过程将其第一个参数定义为输入/输出参数。以下 prepare_for_download 脚本将 var1 的值更改为 pfd_value：

```
CALL ml_add_connection_script (
  'version1',
  'prepare_for_download',
  '{call MyPFDProc( {ml ui.var1} )}' );
```

以下 begin_download 脚本引用 var1，其值现在是 pfd_value：

```
CALL ml_add_connection_script (
  'version1',
  'begin_download',
  'insert into SomeTable values( {ml s.username}, {ml ui.var1} )' );
```

验证参数

在 MobiLink 脚本中，验证参数是命名参数，其前缀为字母 a，例如 {ml a.1}。这些参数必须是从 1 开始的数字，且限制在 255 之内。其值从 MobiLink 客户端发出。

在 authenticate_* 脚本中使用时，验证参数会传递验证信息。

可在所有其它事件（begin_connection 和 end_connection 除外）中使用验证参数传递来自 MobiLink 客户端的信息。在通过创建和填充表格完成某操作时，此技术将是一个非常便捷的方法。

在 SQL Anywhere 远程数据库上，利用 dbmlsync -ap 选项传递信息。在 UltraLite 远程数据库上，利用 auth_parms 和 num_auth_parms 传递信息。

另请参见

- “脚本参数”一节第 299 页
- dbmlsync： “-ap 选项”一节 《MobiLink - 客户端管理》
- UltraLite： “Authentication Parameters 同步参数”一节 《UltraLite - 数据库管理和参考》 和 “Number of Authentication Parameters 参数”一节 《UltraLite - 数据库管理和参考》

示例

对于 UltraLite 远程数据库，请使用 ul_synch_info 结构中的 num_auth_parms 和 auth_parms 字段传递参数。num_auth_parms 是参数数目的计数（0 至 255），auth_parms 是指向字符串数组的指针。为防止以纯文本格式查看这些字符串，它们与口令采用相同的方式进行发送。如果 num_auth_parms 为 0，请将 auth_parms 设置为空。以下是在 UltraLite 中传递参数的一个示例：

```
ul_char * Params[ 3 ] = { UL_TEXT( "param1" ),
   UL_TEXT( "param2" ), UL_TEXT( "param3" ) };

...
info.num_auth_parms = 3;
info.auth_parms = Params;
```

对于 SQL Anywhere 远程数据库，请使用 dbmlsync -ap 选项，通过逗号分隔的列表传递验证参数。例如，以下命令行传递三个参数：

```
dbmlsync -ap "param1,param2,param3"
```

在服务器上，按脚本的发送顺序对其进行引用。在此示例中，authenticate_parameters 脚本可以为：

```
CALL my_auth_parm (
  {ml s.authentication_status},
  {ml s.remote_id},
  {ml s.username},
  {ml a.1},
  {ml a.2},
  {ml a.3}
)
```

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

脚本版本

脚本是以组的方式进行组织的，这些组被称为**脚本版本**。通过指定特定的版本，MobiLink 客户端可以选择使用哪一组同步脚本来处理上载及准备下载。

有关如何将脚本版本添加到统一数据库中的信息，请参见“添加脚本版本”一节第 304 页。

脚本版本的应用

通过脚本版本，可以将脚本分成多个组，以便在不同的情况下运行。此项功能提供了一定的灵活性，在以下情况下尤其有用：

- **自定义应用程序** 使用不同的脚本组来处理来自不同类型远程用户的信息。例如，您可以编写不同的脚本组，供管理员在同步他们的数据库时使用，这组脚本与公司中的其他员工使用的脚本组不同。尽管您可以只使用一组脚本来实现相同的功能，但这些脚本将复杂得多。
- **升级应用程序** 当您希望升级一个数据库应用程序时，因为新版本的应用程序处理数据的方式可能不同，所以可能需要新脚本。当远程数据库发生更改时，几乎总需要新脚本。要想同时升级所有用户通常是不可能的。MobiLink 客户端可以请求在同步过程中使用新的脚本组。由于新旧脚本可以在服务器中共存，因而所有用户都可进行同步，而不管他们使用的应用程序的版本如何。
- **维护多个应用程序** 一个 MobiLink 服务器可能需要同步两个截然不同的应用程序。例如，有些员工可能希望使用用于销售的应用程序，而另一些员工则需要用于进行库存控制的应用程序。如果两个应用程序需要不同的数据集，您可以创建两个同步脚本版本，每个脚本版本对应于一个应用程序。
- **设置脚本版本属性** 可以为能通过 .NET 或 Java 同步逻辑中的类引用的脚本版本设置属性。请参见“ml_add_property 系统过程”一节第 637 页。

指定版本名称

脚本版本名称是一个字符串。您可以在将脚本添加到统一数据库时指定此名称。例如，如果您使用 ml_add_connection_script 和 ml_add_table_script 这两个存储过程添加脚本，脚本版本名称是第一个参数。而如果您使用 Sybase Central 添加脚本，则系统会提示您输入脚本版本名。

脚本版本不能使用以下名称：**ml_sis_1** 或 **ml_qa_1**。这些名称由 MobiLink 在内部使用。

> **小心**
> 强烈建议脚本版本名不要以 **ml_** 开头。以 ml_ 开头的脚本版本将保留以供内部使用。

为同步指定一个版本

如果在启动同步时在远程站点上未指定脚本版本，则同步失败。

ml_global 脚本版本

可以创建与其它脚本版本用法不同的脚本版本 **ml_global**。如果创建了脚本版本 ml_global，则对其定义一次之后会在所有同步中自动使用与其相关联的连接脚本。切勿将 ml_global 显式指定为脚本版本。

如果先在 ml_global 脚本版本中定义一个脚本，然后在为同步指定的脚本版本中又为同一事件定义一个脚本，则将使用指定的脚本版本。只有在正在被同步的主脚本版本中未定义脚本时，才会使用 ml_global 脚本版本中的脚本。

ml_global 脚本版本只能包含连接级脚本。该脚本版本不是必需的，并且如果您仅使用一个脚本版本，则其可能没有用。

添加脚本版本

所有脚本都与一个脚本版本相关联。在 Sybase Central [管理] 模式中工作时，必须先将版本名添加到统一数据库中，然后才能添加任何连接脚本。利用系统过程添加脚本时，新的版本名会随脚本自动添加。在 Sybase Central [模型] 模式中，只允许使用一个脚本版本，并且缺省情况下其名称与模型名相同。

请参见“脚本版本”一节第 303 页。

♦ 将脚本版本添加到数据库（Sybase Central 管理模式）

1. 在 Sybase Central 中，选择 [**连接**] » [**使用 MobiLink 11 连接**]，然后连接到统一数据库。
2. 右击 [**版本**] 文件夹，然后选择 [**文件**] » [**新建**] » [**版本**]。
3. 请按照 [**创建脚本版本向导**] 中的说明进行操作。

♦ 从数据库中删除脚本版本（Sybase Central 管理模式）

1. 在 Sybase Central 中，选择 [**连接**] » [**使用 MobiLink 11 连接**]，然后连接到统一数据库。
2. 单击 [**版本**] 文件夹。
3. 在右窗格中，右击版本名称，然后选择 [**删除**]。
4. 单击 [**是**]。

♦ 将脚本版本添加到数据库（系统过程）

- 您可以在添加连接脚本或表脚本的同一操作中添加脚本版本。

  有关详细信息，请参见“用于添加或删除脚本的系统过程”一节第 624 页。

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

必需的脚本

运行 MobiLink 服务器时，有些脚本是必需的。哪些脚本是必需的将由您执行双向、仅上载还是仅下载同步来决定。

对于双向或仅上载同步，MobiLink 至少需要以下表脚本中的一种：

- upload_delete
- upload_insert
- upload_new_row_insert
- upload_old_row_insert
- upload_update
- 或者，如果您正在通过直接行处理来处理上载操作，则 MobiLink 将需要一个用于 handle_UploadData 连接事件的脚本。

对于双向或仅下载同步，MobiLink 要求同步过程中的每个表均有一个下载表脚本（download_cursor 或 download_delete_cursor）。或者，如果您正在通过直接行处理来处理下载操作，则 MobiLink 将要求您指定一个 handle_DownloadData 连接脚本。请注意，此脚本可能是空的，您可以在任意其它事件中进行下载。

缺省情况下，如果必需的脚本缺失，则同步将中止。可以使用 MobiLink 服务器的 -fr 选项覆盖此行为。

请参见“-fr 选项”一节第 66 页。

Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

添加和删除脚本

如果您使用 [**创建同步模型向导**]，则部署模型时脚本会自动添加到统一数据库中。

在 Sybase Central [模型] 模式外部创建同步脚本时，必须将它们添加到统一数据库的 MobiLink 系统表中。如果是 SQL 脚本，则整个脚本均保存在 MobiLink 系统表中。如果是 Java 和 .NET 脚本，则会在系统表中注册方法名。脚本的存储方法与方法名的存储方法类似。

请参见“MobiLink 服务器系统表”第 653 页。

如果使用的是 Sybase Central，则必须先将同步版本添加到数据库中，然后才能添加单个脚本。请参见“添加脚本版本”一节第 304 页。

♦ 添加连接脚本（Sybase Central 管理模式）

1. 选择 [**连接**] » [**使用 MobiLink 11 连接**]，然后连接到统一数据库。
2. 右击 [**连接脚本**]，并选择 [**新建**] » [**连接脚本**]。
3. 请按照 [**创建连接脚本向导**] 中的说明进行操作。

♦ 删除连接脚本（Sybase Central 管理模式）

1. 选择 [**连接**] » [**使用 MobiLink 11 连接**]，然后连接到统一数据库。
2. 展开 [**连接脚本**]。
3. 右击连接脚本，然后选择 [**删除**]。
4. 单击 [**是**]。

♦ 添加表脚本（Sybase Central 管理模式）

1. 选择 [**连接**] » [**使用 MobiLink 11 连接**]，然后连接到统一数据库。
2. 展开 [**同步表**]。
3. 右击表，然后选择 [**新建**] » [**表脚本**]。
4. 按照 [**创建表脚本向导**] 中的说明进行操作。

♦ 删除表脚本（Sybase Central 管理模式）

1. 选择 [**连接**] » [**使用 MobiLink 11 连接**]，然后连接到统一数据库。
2. 展开 [**同步表**]。
3. 展开表。
4. 右击表脚本，然后选择 [**删除**]。
5. 单击 [**是**]。

Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

♦ **添加或删除所有类型的脚本（系统过程）**

- 您可以使用在建立统一数据库时安装的存储过程，将脚本添加到统一数据库中或从统一数据库中删除脚本。

  有关可用于添加或删除脚本的存储过程的说明，请参见：

  - “ml_add_connection_script 系统过程”一节第 627 页
  - “ml_add_table_script 系统过程”一节第 639 页
  - “ml_add_dnet_connection_script 系统过程”一节第 628 页
  - “ml_add_dnet_table_script 系统过程”一节第 629 页
  - “ml_add_java_connection_script 系统过程”一节第 630 页
  - “ml_add_java_table_script 系统过程”一节第 631 页

脚本的直接插入

大多数情况下，建议使用存储过程或 Sybase Central 将脚本插入系统表。不过，有些极少数情况下，可能需要使用 INSERT 语句直接插入脚本。例如，有些 RDBMS 的旧版本可能有长度限制，这样便很难使用存储过程。

有关 MobiLink 系统表的完整说明，请参见“MobiLink 服务器系统表”第 653 页。

在 ml_add_connection_script 和 ml_add_table_script 存储过程的源代码中，可以找到直接插入脚本所需的 INSERT 语句的格式。这些存储过程的源代码位于 MobiLink 安装脚本中。每个受支持的 RDBMS 都有一个不同的安装脚本。所有安装脚本均位于 *install-dir\MobiLink\setup* 中且被称作：

统一数据库	安装文件
Adaptive Server Enterprise	*syncase.sql*
IBM DB2 主机	*syncd2m.sql* 或 *syncd2m_jcl.sql*
IBM DB2 LUW	*syncdb2.sql*
Microsoft SQL Server	*syncmss.sql*
MySQL	*syncmys.sql*
Oracle	*syncora.sql*
SQL Anywhere	*syncsa.sql*

忽略脚本

如果上载流包含一个表的插入、更新或删除数据，但统一数据库中没有 upload_insert、upload_update 和 upload_delete 脚本，或者如果没有该表的下载脚本（download_cursor 和 download_delete_cursor 脚本），则 MobiLink 服务器会执行以下操作之一：

- 如果 MobiLink 服务器不是以 -fr 启动，则可能会报告缺少脚本并中止同步
- 如果 MobiLink 服务器是以 -fr 启动，则将显示消息以提醒用户注意数据不一致的问题

使用 -zwd MobiLink 服务器命令选项可取消警告消息，但是此选项会取消所有同步表的警告消息。

现在，MobiLink 服务器将以不同的方式处理任何包含前缀 --{ml_ignore} 的连接脚本和表脚本。MobiLink 服务器会将这些脚本识别为有意忽略的脚本。更确切地说，如果上载流包含同步表（具有带 --{ml_ignore} 前缀的 upload_insert、upload_update 或 upload_delete 脚本）的插入、更新或删除数据，则无论服务器是否以 -fr 选项启动，MobiLink 服务器都将不会对统一数据库执行这些脚本，而是在不显示任何错误或警告消息的情况下继续同步。

此逻辑也适用于下载。但是，如果上载流包含一个同步表（没有有意忽略的或实际的 upload_delete（upload_insert 或 upload_update 脚本）的删除（插入或更新）数据，则即使没有此表的有意忽略的或实际的 upload_insert 和 upload_update 脚本，MobiLink 服务器也会中止同步或显示警告消息。

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

编写用于上载行的脚本

要告知 MobiLink 服务器如何处理从远程数据库接收的上载流数据，您需要定义上载脚本。您可以编写分别处理在远程数据库中更新、插入或删除的行的脚本。一种简单的实现方式将在统一数据库中执行相应的操作（更新、插入、删除）。

MobiLink 服务器在单个事务中上载数据。有关上载过程的说明，请参见“上载过程中的事件”一节第 326 页。

有关使用 .NET 同步逻辑上载行的方法，请参见“上载行或下载行”一节第 562 页。

注意

- 每个远程表的 begin_upload 和 end_upload 脚本所采用的逻辑独立于所更新的各行。
- 上载是由单个行插入、更新和删除操作组成的。通常使用 upload_insert、upload_update 和 upload_delete 脚本来执行这些操作。
- 为准备 SQL Anywhere 客户端的上载，dbmlsync 实用程序需要访问自上次成功同步以来写入的所有事务日志。请参见“事务日志文件”一节 《MobiLink - 客户端管理》。
- 使用 MobiLink 客户端版本 9.0 或更早版本同步远程数据库，或在 upload_insert、upload_new_row_insert 或 upload_old_row_insert 事件中使用问号而不是命名参数作为占位符时，MobiLink 服务器使用远程数据库中定义的表的列顺序。事件语句中的列顺序必须与远程数据库中定义的列顺序匹配，而在统一数据库中的表和列的名称可以不同于远程数据库中的表和列的名称。

  下面是一个 INSERT 语句，该语句仅当 emp_name 在远程数据库中的定义先于 emp_id 时才使用。

```
INSERT INTO emp (emp_name, emp_id)
VALUES (?, ?);
```

编写 upload_insert 脚本

在处理上载的过程中，MobiLink 服务器使用此事件来处理插入到远程数据库中的行。

下面是一个在 upload_insert 脚本中使用的 INSERT 语句。

```
INSERT INTO emp ( emp_id, emp_name )
VALUES ( { ml r.emp_id }, { ml r.emp_name } );
```

注意

- upload_new_row_insert 和 upload_old_row_insert 事件接受 remote_id 和 user_name 作为附加参数。这些参数必须位于表格的整列列表之前。

另请参见

- “编写用于上载行的脚本”一节第 309 页
- “upload_insert 表事件”一节第 471 页

编写 upload_update 脚本

在处理上载的过程中，MobiLink 服务器使用此事件处理在远程数据库中更新的行。以下 UPDATE 语句介绍了 upload_update 语句的使用方法。

```
UPDATE emp
SET emp_name = {ml r.emp_name}
WHERE emp_id = {ml o.emp_id};
```

注意

- 使用 MobiLink 客户端版本 9.0 或更早版本同步远程数据库，或使用问号而不是命名参数作为占位符时，参数数目可以等于下列值之一：
  - 非主键列数 + 主键列数。
  - 2 x（非主键列数 + 主键列数）。

  列顺序的组成必须先是非主键列，后跟下列项之一：
  - 主键列。
  - 所有列。

另请参见

- “编写用于上载行的脚本”一节第 309 页
- “upload_update 表事件”一节第 488 页

编写 upload_delete 脚本

在处理上载的过程中，MobiLink 服务器使用此事件处理从远程数据库中删除的行。以下语句给出了使用 upload_delete 语句的方法。

```
DELETE FROM emp
WHERE emp_id = {ml r.emp_id};
```

注意

- 使用 MobiLink 客户端版本 9.0 或更早版本同步远程数据库，或使用问号而不是命名参数作为占位符时，参数数目必须等于下列值之一：
  - 主键列数。
  - 所有列数。

另请参见

- “编写用于上载行的脚本”一节第 309 页
- “upload_delete 表事件”一节第 465 页

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

编写 upload_fetch 脚本

upload_fetch 脚本是一个 SELECT 语句，用于在统一数据库表中定义游标。此游标用于比较从远程数据库中接收的已更新行的旧值和位于统一数据库中的值。通过这种方式，upload_fetch 脚本可在处理更新操作时识别冲突。

假定一个同步表如下定义：

```
CREATE TABLE uf_example (
 pk1 integer NOT NULL,
 pk2  integer NOT NULL,
 val  varchar(200),
 PRIMARY KEY( pk1, pk2 ));
```

则此表的一种可能的 upload_fetch 脚本为：

```
SELECT pk1, pk2, val
FROM uf_example
WHERE pk1 = {ml r.pk1} and pk2 = {ml r.pk2}
```

请参见“upload_fetch 表事件”一节第 467 页。

MobiLink 服务器要求 upload_fetch 脚本中的 WHERE 查询子句确定对统一数据库中的具体哪一行进行冲突检查。

编写用于下载行的脚本

有两个脚本可用来在下载事务的过程中处理每个表。它们是 download_cursor 脚本（执行插入和更新）和 download_delete_cursor 脚本（执行删除）。

这些脚本可以是 SELECT 语句，也可以是对返回结果集的过程的调用。MobiLink 服务器会将该脚本的结果集下载到远程数据库。MobiLink 客户端自动基于 download_cursor 脚本结果集插入或更新行，基于 download_delete_cursor 事件删除行。

有关使用存储过程的详细信息，请参见“从存储过程调用中下载结果集”一节第 153 页。

MobiLink 服务器在单个事务中下载数据。有关下载过程的说明，请参见“下载过程中的事件”一节第 329 页。

注意

- 与上载类似，下载与连接事件一同开始和结束。其它事件是表级别的事件。
- 如果将 SendDownloadAck 设置更改为 ON，则服务器行为取决于所使用的下载确认模式。对于阻塞下载确认，如果没有从客户端获得下载确认，则整个下载事务将在统一数据库中回退。对于非阻塞下载确认，下载事务将提交但不执行下载时间戳更新和确认脚本，直到接受确认。

  缺省情况下，SendDownloadAck 设为 OFF。

  请参见“SendDownloadACK (sa) 扩展选项”一节 《MobiLink - 客户端管理》、“发送下载确认同步参数”一节 《UltraLite - 数据库管理和参考》、“-nba 选项”一节第 70 页、“nonblocking_download_ack 连接事件”一节第 438 页和“publication_nonblocking_download_ack 连接事件”一节第 442 页。
- 每个远程表的 begin_download 和 end_download 脚本所采用的逻辑独立于所更新的各行。
- 对于基于时间戳的下载，需指定 last_download_timestamp 参数以确保仅下载自上次同步以来所做的更改。例如，download_cursor 或 download_delete_cursor SQL 脚本可能包括如下行：

  ```
  WHERE Customer.last_modified >= {ml s.last_table_download}
  ```

  请参见“在脚本中使用上次下载时间”一节第 122 页。
- 下载不区分插入操作与更新操作。与 download_cursor 事件关联的脚本是一个 SELECT 语句，该语句定义要下载的行。客户端将检测行是否存在，然后执行适当的插入或更新操作。
- 下载过程结束时，客户端根据需要自动删除行，以避免破坏参照完整性。

小心
不要同步先前部署创建的影子表（例如，不应同步以 *_mod* 或 *_del* 结尾的表）。这些表仅供统一数据库用于跟踪修改或删除的行。

请参见“参照完整性与同步”一节 《MobiLink - 入门》。

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

编写 download_cursor 脚本

可编写 download_cursor 脚本将信息从统一数据库下载到远程数据库。您必须为远程数据库中每个需要下载更改的表编写一个此类型的脚本。还可以使用其它脚本来自定义下载过程，但这不是必需的。

- 每个 download_cursor 脚本必须包含一个 SELECT 语句或对含有 SELECT 语句的过程的调用。MobiLink 服务器将使用此语句在统一数据库中定义游标。
- 该脚本必须选择与远程数据库中相应的表中的列对应的所有列。统一数据库中的列可以与远程数据库中相应的列具有不同的名称，但是列的类型必须相互兼容。

示例

以下脚本可以作为用于保存雇员信息的远程表的 download_cursor 脚本。MobiLink 服务器将使用此 SQL 语句定义下载游标。此脚本将下载有关所有雇员的信息。

```
SELECT emp_id, emp_fname, emp_lname
FROM employee;
```

MobiLink 服务器将特定参数传递给某些脚本。要使用这些参数，可以使用命名参数，也可以在 SQL 语句中包括一个问号。如果采用后一种方法，则在对统一数据库执行该语句之前，MobiLink 服务器将替换该参数的值。以下脚本说明如何使用命名参数：

```
CALL ml_add_table_script(
   'Lab',
   'ULOrder',
   'download_cursor',
   'SELECT o.order_id, o.cust_id, o.prod_id, o.emp_id, o.disc, o.quant,
o.notes, o.status
         FROM ULOrder o
         WHERE o.last_modified >= {ml s.last_table_download}
               AND o.emp_name = {ml s.username}' )
```

注意

- 可以从单个表或多个表的连接中选择行值。
- 脚本自身无需包括远程表名。远程表名称也无需与统一数据库中表的名称相同。远程表名由 ml_table MobiLink 系统表中的一个条目标识。在 Sybase Central 中，远程表与其相应的脚本一同列出。
- 远程表中的行必须包含 emp_id、emp_fname 和 emp_lname 的值。远程列必须按照该固定顺序排列，尽管它们可能具有不同的名称。远程数据库中列的顺序应与参考数据库中列的顺序相同。
- 所有游标脚本都必须按照列在远程数据库中定义的顺序来选择列。如果在统一数据库中列名称或表结构不同，则应按远程数据库或参考数据库（两者等价）中正确的顺序选择列。系统会根据列在 SELECT 语句中的顺序，将列指派给远程数据库。
- 建立 UltraLite 应用程序时，UltraLite 生成器将为 UltraLite 应用程序中的每个表创建一个示例下载脚本。并将这些示例脚本插入到参考数据库中。示例脚本假定统一数据库中包含的表与您的应用程序中的表相同。如果统一数据库在设计上有所不同，则您必须修改示例脚本，但这些脚本仍然为您提供了一个起点。
- download_cursor 脚本必须包含按远程数据库中所定义的顺序排列的所有列。

另请参见

- “download_cursor 表事件”一节第 370 页
- “在远程数据库之间对行进行分区”一节第 127 页
- “编写 download_delete_cursor 脚本”一节第 314 页

编写 download_delete_cursor 脚本

编写 download_delete_cursor 脚本以从远程数据库中删除行。您必须为远程数据库中每个需要在同步过程中删除行的表，编写一个此类型的脚本。

仅在统一数据库中删除行，不能使它们在远程数据库中消失。您需要跟踪已删除行的主键，以便使用 download_delete_cursor 选择这些主键。有两种常用技术可以实现这一点：

- **逻辑删除** 不在统一数据库中物理删除行。而是用一个状态列来跟踪行是否有效。这简化了 download_delete_cursor。但是，可能需要对 download_cursor 和其它应用程序进行修改，以使它们能识别和使用状态行。如果您有一个记录删除时间的上次修改列，而且您还对每个远程数据库的上次下载时间进行跟踪，则在所有远程下载时间都晚于删除时间时，即可物理地删除该行。
- **影子表** 为您想要跟踪其删除操作的每个表创建一个包含两个列的影子表：一个列存放该表的主键，另一列存放时间戳。创建一个只要行被删除就会向影子表中插入主键和时间戳的触发器。然后，您的 download_delete_cursor 就可以从这个影子表中进行选择。与逻辑删除一样，一旦所有远程数据库都下载了相应的数据，您就可以将该行从影子表中删除。

MobiLink 服务器通过从统一数据库选择主键值并将这些值传递到远程数据库，来实现远程数据库中行的删除。如果这些值与远程数据库中的主键值相匹配，则该行将被删除。

- 每个 download_delete_cursor 脚本必须包含一个 SELECT 语句，或者一个对返回结果集的存储过程的调用。MobiLink 服务器将使用此语句在统一数据库中定义游标。
- 此语句必须选择与远程数据库表中的主键列对应的所有列。统一数据库中的列可以与远程数据库中相应的列具有不同的名称，但是列的类型必须相互兼容。
- 必须按照与值对应的列在远程数据库中定义的顺序选择值。此顺序是用于创建表的 CREATE TABLE 语句中的列顺序，而不是列在定义主键的语句中出现的顺序。
- 如果删除父记录，则子记录也会自动删除。

  有关删除子记录的详细信息，请参见“参照完整性与同步”一节 《MobiLink - 入门》 。

尽管每个 download_delete_cursor 脚本必须选择在相应远程表的主键中出现的所有列值，但它也可以选择所有其它列。此功能的目的是为了与旧客户端的兼容。选择更多列会降低效率，因为数据库服务器必须检索更多的数据。如果客户端的设计不是旧式的，MobiLink 服务器会立即放弃多余的值。多余的值只能下载到旧的客户端。

删除表中的所有行

当 MobiLink 检测到 download_delete_cursor 中的某行全部为空时，它将删除远程表中的所有数据。download_delete_cursor 中的空值数可以是表中的主键列数或总列数。

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

例如，以下 download_delete_cursor SQL 脚本删除具有两个主键列的表中的每一行。此示例适用于 SQL Anywhere、Adaptive Server Enterprise 和 Microsoft SQL Server 数据库。

```
SELECT NULL, NULL
```

在 IBM DB2 LUW 和 Oracle 统一数据库中，必须指定一个虚表来选择空列。对于 IBM DB2 LUW 7.1，可以使用以下语法：

```
SELECT NULL, NULL FROM SYSIBM.SYSDUMMY1
```

对于 Oracle 统一数据库，可以使用以下语法：

```
SELECT NULL, NULL FROM DUAL
```

示例

以下示例是用于保存雇员信息的远程表的 download_delete_cursor 脚本。MobiLink 服务器使用此 SQL 语句定义删除游标。此脚本在执行时删除既在统一数据库中又在远程数据库中的所有雇员的信息。

```
SELECT emp_id
FROM employee
```

download_delete_cursor 接受参数 last_download 和 ml_username。以下脚本说明如何使用每个参数来缩小选择范围。

```
SELECT order_id
FROM ULOrder
WHERE last_modified >= {ml s.last_table_download}
 AND status = 'Approved'
 AND user_name = {ml s.username}
```

注意

对于某些统一数据库，您可能需要转换为适当的数据类型。请参见“CAST 函数 [Data type conversion]”一节 《SQL Anywhere 服务器 - SQL 参考》。

这些示例在拥有大量雇员的组织中工作效率可能会很低。您可以只选择远程数据库中可能存在的行，从而提高删除过程的效率。例如，可以仅选择新近被指派为经理的人员，从而限制所选行的数目。另一个策略是允许客户端应用程序删除行本身。此方法只有当规则标识出不需要的行时才可能使用。例如，行可以包含一个指示失效日期的时间戳。在删除行之前，使用 STOP SYNCHRONIZATION DELETE 语句在下一次同步过程中停止这些要上载的删除操作。如果要以正常的方式同步其它删除，请确保之后立即执行 START SYNCHRONIZATION DELETE。

注意

- download_delete_cursor 脚本必须包含按远程数据库中所定义的顺序排列的主键列。

另请参见

您可以使用所有 MobiLink 客户端内置的参照完整性检查来删除行，这是一种高效的方法。请参见“参照完整性与同步”一节 《MobiLink - 入门》。

有关使用 download_delete_cursor 脚本的详细信息，请参见：

- “download_cursor 表事件”一节第 370 页
- “download_delete_cursor 表事件”一节第 373 页
- “处理删除”一节第 147 页
- “暂停删除同步”一节第 147 页
- “STOP SYNCHRONIZATION DELETE 语句 [MobiLink]”一节 《SQL Anywhere 服务器 - SQL 参考》
- “在远程数据库之间对行进行分区”一节第 127 页
- “快照同步”一节第 125 页

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

编写用于处理错误的脚本

当 MobiLink 服务器执行同步脚本时，如果脚本中的一个操作失败，则在该脚本中将发生一个错误。DBMS 将向 MobiLink 服务器返回 SQLCODE，指示该错误的性质。每个统一数据库 DBMS 都有自己的一组 SQLCODE 值。

当有错误发生时，MobiLink 服务器调用 handle_error 事件。您应该提供一个与此事件关联的连接脚本以处理错误。MobiLink 服务器向此脚本传递多个参数以提供错误的性质和上下文信息，并且需要一个输出值以指出应如何响应错误。

错误处理操作

您可能要在错误处理脚本中包含以下操作：

- 在一个单独的表中记录该错误。
- 指示 MobiLink 服务器选择如下操作：忽略错误并继续执行、回退同步过程或者回退同步过程并关闭 MobiLink 服务器。
- 发送电子邮件消息。

有关详细信息，请参见“handle_error 连接事件”一节第 414 页。

报告错误

因为错误可能会中断同步过程的正常进行，所以创建错误日志及其解决办法会比较困难。report_error 脚本提供了完成此任务的方法。只要发生错误，MobiLink 服务器即会执行此脚本。如果定义了 handle_error 脚本，则此脚本将先于报告脚本执行。

report_error 脚本的参数与 handle_error 脚本的参数相同，只是 report_error 脚本无法修改操作代码。由于操作代码的值是由 handle_error 脚本返回的，因此该脚本可用于调试错误处理问题。

此脚本通常包含一个插入语句，该语句将在一个表中记录值，可能还会记录诸如时间或日期之类的其它数据，以供以后引用。为确保数据不会丢失，MobiLink 服务器总是在单独的事务中运行此脚本，并在脚本完成时自动提交更改信息。

请参见“report_error 连接事件”一节第 444 页。

示例

以下 report_error 脚本由单个插入语句组成，它在表中添加脚本参数及当前日期和时间。该脚本并不提交此更改，因为 MobiLink 服务器总是自动完成这一操作。

```
INSERT INTO errors
VALUES(
  CURRENT DATE,
  {ml s.action_code},
  {ml s.error_code},
  {ml s.error_message},
  {ml s.username},
  {ml s.table} );
```

Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

处理单个 SQL 语句的多个错误

ODBC 允许每个 SQL 语句发生多个错误，因而某些 RDBMS 利用了这一特性。例如，Microsoft SQL Server 允许为单个语句报告两个错误。第一个是实际的错误，第二个通常是告知当前语句终止原因的信息性消息。

当单个 SQL 语句导致多个错误时，对于每个错误都调用一次 handle_error 脚本。MobiLink 服务器将使用严重性最高的操作代码（即最大数值）确定需要执行的操作。这同样适用于 handle_error 脚本。

如果 handle_error 脚本自身导致 SQL 错误，将使用缺省动作代码 (3000)。

Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

同步事件

目录

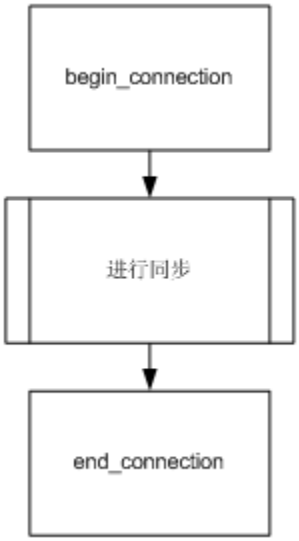
MobiLink 事件概述 321
authenticate_file_transfer 连接事件 331
authenticate_parameters 连接事件 332
authenticate_user 连接事件 335
authenticate_user_hashed 连接事件 339
begin_connection 连接事件 343
begin_connection_autocommit 连接事件 344
begin_download 连接事件 345
begin_download 表事件 347
begin_download_deletes 表事件 349
begin_download_rows 表事件 352
begin_publication 连接事件 355
begin_synchronization 连接事件 358
begin_synchronization 表事件 360
begin_upload 连接事件 362
begin_upload 表事件 364
begin_upload_deletes 表事件 366
begin_upload_rows 表事件 368
download_cursor 表事件 370
download_delete_cursor 表事件 373
download_statistics 连接事件 376
download_statistics 表事件 379
end_connection 连接事件 382
end_download 连接事件 384
end_download 表事件 386
end_download_deletes 表事件 388
end_download_rows 表事件 391
end_publication 连接事件 394
end_synchronization 连接事件 397
end_synchronization 表事件 399
end_upload 连接事件 401

end_upload 表事件 403
end_upload_deletes 表事件 406
end_upload_rows 表事件 408
handle_DownloadData 连接事件 410
handle_error 连接事件 414
handle_odbc_error 连接事件 418
handle_UploadData 连接事件 422
modify_error_message 连接事件 428
modify_last_download_timestamp 连接事件 430
modify_next_last_download_timestamp 连接事件 433
modify_user 连接事件 436
nonblocking_download_ack 连接事件 438
prepare_for_download 连接事件 440
publication_nonblocking_download_ack 连接事件 442
report_error 连接事件 444
report_odbc_error 连接事件 447
resolve_conflict 表事件 450
synchronization_statistics 连接事件 453
synchronization_statistics 表事件 456
time_statistics 连接事件 459
time_statistics 表事件 462
upload_delete 表事件 465
upload_fetch 表事件 467
upload_fetch_column_conflict 表事件 469
upload_insert 表事件 471
upload_new_row_insert 表事件 473
upload_old_row_insert 表事件 476
upload_statistics 连接事件 479
upload_statistics 表事件 483
upload_update 表事件 488

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

MobiLink 事件概述

当有同步请求发出且 MobiLink 服务器确定必须建立新连接时，将触发 begin_connection 事件并启动同步。



随着同步的进行，连接被放入连接池中，MobiLink 再次等待同步请求。在一个连接被最终从连接池中删除之前，将触发 end_connection 事件。但是如果接收到另一个对同一脚本版本的同步请求，MobiLink 将在同一连接上处理下一个同步请求。有几个事件会对当前同步产生影响。

事务

在每个同步中，可能发生以下事务。每个事务都是可选的。

- 验证
- 开始同步
- 上载

  可以使用 dbmlsync -tu 选项指定多个上载事务。
- 准备下载
- 下载
- 结束同步
- 非阻塞下载确认

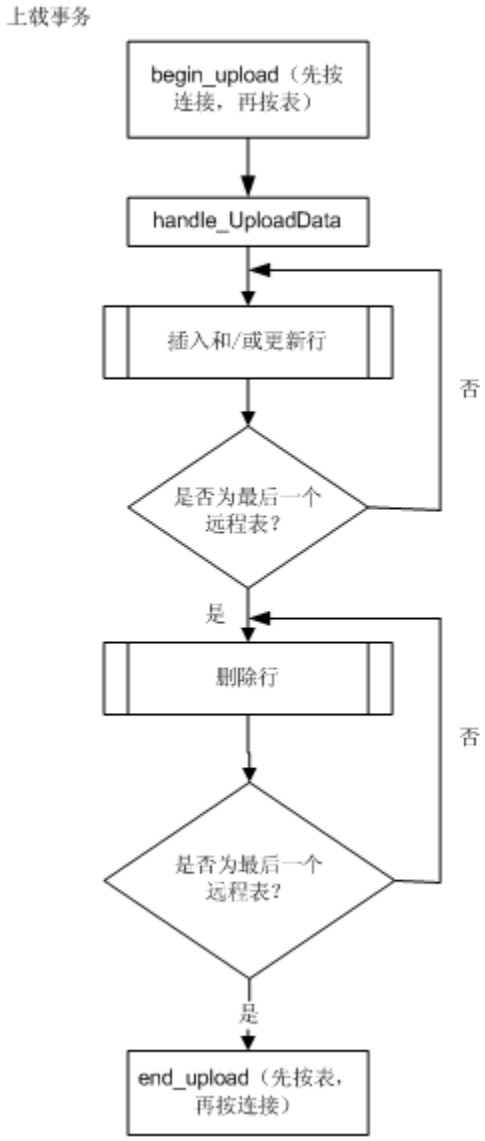
此外，还可以有两个连接事务。一个开始连接事务，建立连接后即发生；以及一个结束连接事务，关闭连接时发生。

同步的主要阶段是上载和下载事务。上载和下载事务中包含的事件将在下面介绍。

上载事务

上载事务应用从远程数据库上载的更改。

begin_upload 事件标志着上载事务的开始。上载事务是一个由两个部分组成的过程。首先，上载所有远程表中的插入和更新操作，然后上载所有远程表中的删除操作。



end_upload 事件标志着上载事务的结束。

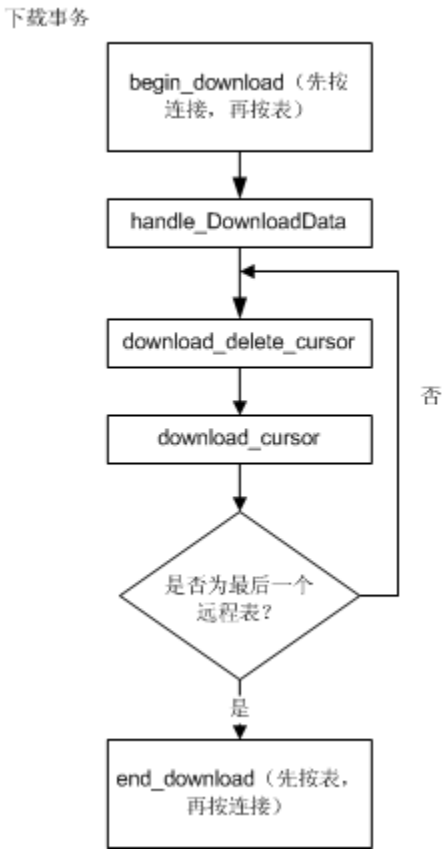
请参见“编写用于上载行的脚本”一节第 309 页。

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

下载事务

下载事务从统一数据库中读取行。它开始于 begin_download 事件。

下载事务是一个由两个部分组成的过程。对于每个表，首先下载删除操作，然后下载更新/插入行 (upsert)。end_download 事件结束下载事务。



请参见“编写用于下载行的脚本”一节第 312 页。

非阻塞下载确认事务

非阻塞下载确认事务仅在 MobiLink 处于非阻塞下载确认模式且收到下载确认时执行。此事务具有两个目的。脚本 publication_nonblocking_download_ack 和 nonblocking_download_ack 都在此事务中运行；它们有助于下载状态跟踪。其次，MobiLink 系统表中的下载时间戳在此事务期间更新。

请注意，此事务不是在与目标同步的其它事件相同的数据库连接上执行的。这意味着在此事务中没有连接级变量可以引用。

伪代码事件概述

以下伪代码概述了调用各种事件以及调用同名脚本的序列。此 MobiLink 事件模型的表示形式假定存在一个没有出现错误的完整同步（而不是仅上载同步或仅下载同步）。

注意

- 在大多数情况下，如果没有为给定事件定义脚本，则缺省操作是不执行任何操作。
- begin_connection 和 end_connection 事件是**连接解别的事件**。它们独立于任何一个同步并且没有参数。
- 对于正在执行同步的每一个表，有些事件将在每个同步过程中被调用一次。与此类事件相关联的脚本被称为**表级别脚本**。

  尽管每个表都可以拥有自己的表脚本，但您也可以编写供多个表共享的表级别的脚本。
- 有些事件，如 begin_synchronization，在连接级别和表级别都会发生。您可以为这些事件提供连接级别和表级别的脚本。
- COMMIT 语句说明了如何将同步过程分解为不同的事务。
- 数据库错误可能会在同步过程中的任何时刻发生。可使用 handle_error 或 handle_odbc_error 脚本处理数据库错误。

在 SQL 同步脚本中或从 SQL 同步脚本调用的过程或触发器中，不应有任何隐式或显式的提交或回退。SQL 脚本内的 COMMIT 或 ROLLBACK 语句会改变同步步骤的事务性质。如果您使用这两个语句，则在出现故障时 MobiLink 将无法保证数据的完整性。

完整的 MobiLink 事件模型

```
-------------------------------------------------------
MobiLink complete event model.

Legend:
- // This is a comment.
- <name>
    The pseudo code for <name> is listed separately
    in a later section, under a banner:
        ------------------------
        name
        ------------------------
- VariableName <- value
    Assign the given value to the given variable name.
    Variable names are in mixed case.
- event_name
    If you have defined a script for the given event name,
    it is invoked.
-------------------------------------------------------

CONNECT to consolidated database
begin_connection_autocommit
begin_connection
COMMIT
for each synchronization request with
     the same script version {
```

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

```
  <synchronize>
}
end_connection
COMMIT
DISCONNECT from consolidated database

-------------------------------------------------------
synchronize
-------------------------------------------------------

<authenticate>
<begin_synchronization>
<upload>
<prepare_for_download>
<download>
<end_synchronization>

-------------------------------------------------------
authenticate
-------------------------------------------------------

Status <- 1000
UseDefaultAuthentication <- TRUE
if( authenticate_user script is defined ) {
  UseDefaultAuthentication <- FALSE
  TempStatus <- authenticate_user
  if( TempStatus > Status ) {
    Status <- TempStatus
  }
}
if( authenticate_user_hashed script is defined ) {
  UseDefaultAuthentication <- FALSE
  TempStatus <- authenticate_user_hashed
  if( TempStatus > Status ) {
    Status <- TempStatus
  }
}
  if( authenticate_parameters script is defined )
 {
    TempStatus <- authenticate_parameters
    if( TempStatus > Status ) {
      Status <- TempStatus
  }
if( UseDefaultAuthentication ) {
  if( the user exists in the ml_user table ) {
    if( ml_user.hashed_password column is not NULL ) {
      if( password matches ml_user.hashed_password ) {
        Status <- 1000
      } else {
        Status <- 4000
      }
    } else {
      Status <- 1000
    }
  } else if( -zu+ was on the command line ) {
    Status <- 1000
  } else {
    Status <- 4000
  }
}
if( Status >= 3000 ) {
  // Abort the synchronization.
} else {
  // UserName defaults to MobiLink user name
```

```
  // sent from the remote.
  if( modify_user script is defined ) {
    UserName <- modify_user
    // The new value of UserName is later passed to
    // all scripts that expect the MobiLink user name.
  }
}
COMMIT

------------------------------------------------------
begin_synchronization
------------------------------------------------------

begin_synchronization   // Connection event.
for each table being synchronized {
    begin_synchronization    // Call the table level script.
}
for each publication being synchronized {
  begin_publication
}
COMMIT

------------------------------------------------------
end_synchronization
------------------------------------------------------

for each publication being synchronized {
  if( begin_publication script was called ) {
    end_publication
  }
}
for each table being synchronized {
  if( begin_synchronization table script was called ) {
    end_synchronization // Table event.
  }
}
if( begin_synchronization connection script was called ) {
  end_synchronization     // Connection event.
}
for each table being synchronized {
synchronization_statistics // Table event.
}
synchronization_statistics // Connection event.
for each table being synchronized {
  time_statistics // Table event.
}
time_statistics // Connection event.

COMMIT
```

有关上载处理的详细信息，请参见“上载过程中的事件”一节第 326 页。

有关下载处理的详细信息，请参见“下载过程中的事件”一节第 329 页。

上载过程中的事件

下面的伪代码说明了如何调用上载事件和上载脚本。

这些事件在完整事件模型中的上载位置发生。请参见“MobiLink 事件概述”一节第 321 页。

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

上载概述

```
-------------------------------------------------------
upload
-------------------------------------------------------

begin_upload // Connection event
for each table being synchronized {
  begin_upload // Table event
}
  handle_UploadData
  for each table being synchronized {
    begin_upload_rows
    for each uploaded INSERT or UPDATE for this table {
      if( INSERT ) {
        <upload_inserted_row>
      }
      if( UPDATE ) {
        <upload_updated_row>
      }
    }
    end_upload_rows
  }
  for each table being synchronized IN REVERSE ORDER {
    begin_upload_deletes
    for each uploaded DELETE for this table {
      <upload_deleted_row>
    }
    end_upload_deletes
  }

For each table being synchronized {
  if( begin_upload table script is called ) {
    end_upload // Table event
  }
}
if( begin_upload connection script was called ) {
  end_upload // Connection event

  for each table being synchronized {
    upload_statistics  // Table event.
  }
    upload_statistics  // Connection event.

  COMMIT
```

上载插入

```
-------------------------------------------------------
<upload_inserted_row>
-------------------------------------------------------
// NOTES:
// - Only table scripts for the current table are involved.

  ConflictsAreExpected <- (
       upload_new_row_insert script is defined
    or upload_old_row_insert script is defined
    or resolve_conflict script is defined )
  if( upload_insert script is defined ) {
    upload_insert
  } else if( ConflictsAreExpected
      and upload_update script is not defined
      and upload_insert script is not defined
      and upload_delete script is not defined ) {
```

```
        // Forced conflict.
        upload_new_row_insert
        resolve_conflict
    } else {
        // Ignore the insert.
    }
```

上载更新

```
-------------------------------------------------------
upload_updated_row
-------------------------------------------------------
// NOTES:
// - Only table scripts for the current table are involved.
// - Both the old (original) and new rows are uploaded for
//   each update.

  ConflictsAreExpected <- (
       upload_new_row_insert script is defined
    or upload_old_row_insert script is defined
    or resolve_conflict script is defined )
  Conflicted <- FALSE
  if( upload_update script is defined ) {
    if( ConflictsAreExpected
      and upload_fetch script is defined ) {
      FETCH using upload_fetch INTO current_row
      if( current_row <> old row ) {
        Conflicted <- TRUE
      }
    }
    if( not Conflicted ) {
      upload_update
    }
  } else if( upload_update script is not defined
      and upload_insert script is not defined
      and upload_delete script is not defined ) {
      // Forced conflict.
      Conflicted <- TRUE
  }
  if( ConflictsAreExpected and Conflicted ) {
    upload_old_row_insert
    upload_new_row_insert
    resolve_conflict
  }
```

上载删除

```
-------------------------------------------------------
upload_deleted_row
-------------------------------------------------------
// NOTES:
// - Only table scripts for the current table are involved.

  ConflictsAreExpected <- (
       upload_new_row_insert script is defined
    or upload_old_row_insert script is defined
    or resolve_conflict script is defined )
  if( upload_delete is defined ) {
    upload_delete
  } else if( ConflictsAreExpected
    and upload_update script is not defined
    and upload_insert script is not defined
    and upload_delete script is not defined ) {
    // Forced conflict.
```

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

```
    upload_old_row_insert
    resolve_conflict
  } else {
    // Ignore this delete.
  }
```

下载过程中的事件

下面的伪代码概述了调用下载事件以及调用同名脚本的序列。

这些事件在“MobiLink 事件概述”一节第 321 页中提供的完整事件模型中的下载位置发生。

```
-------------------------------------------------------
prepare_for_download
-------------------------------------------------------

modify_last_download_timestamp
fetch the next download timestamp from consolidated
prepare_for_download
if( modify_last_download_timestamp script is defined
    or prepare_for_download script is defined ) {
    COMMIT
}

-------------------------------------------------------
download
-------------------------------------------------------

begin_download // Connection event.
for each table being synchronized {
   begin_download // Table event.
}
   handle_DownloadData
   for each table being synchronized {
     begin_download_deletes
     for each row in download_delete_cursor {
       if( all primary key columns are NULL ) {
         send TRUNCATE to remote
       } else {
         send DELETE to remote
       }
     }
     end_download_deletes
     begin_download_rows
     for each row in download_cursor {
       send INSERT ON EXISTING UPDATE to remote
     }
     end_download_rows
   }
   modify_next_last_download_timestamp
   for each table being synchronized {
     if( begin_download table script is called ) {
        end_download // Table event
     }
}
if( begin_download connect script is called ) {
   end_download // Connection event
}
   for each table being synchronized {
     download_statistics   // Table event.
   }
```

```
        download_statistics    // Connection event.
    COMMIT
```

注意

- 如果希望得到确认，并且没有从客户端获得下载确认，则整个下载事务将在统一数据库中回退。

  有关 SQL Anywhere 远程数据库的信息，请参见“SendDownloadACK (sa) 扩展选项”一节《MobiLink - 客户端管理》。有关 UltraLite 远程数据库的信息，请参见“发送下载确认同步参数”一节 《UltraLite - 数据库管理和参考》。

- 下载流不区分插入操作与更新操作。与 download_cursor 事件关联的脚本是一个 SELECT 语句，该语句定义要下载的行。客户端将检测行是否存在，然后执行适当的插入或更新操作。

- 下载过程结束时，客户端将自动删除破坏参照完整性的行。

  请参见“参照完整性与同步”一节 《MobiLink - 入门》。

Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

authenticate_file_transfer 连接事件

使用 mlfiletransfer 实用程序或 MLFileTransfer 方法实现文件传输的自定义验证。

参数

在下表中，说明部分提供 SQL 数据类型。如果您使用 Java 或 .NET 编写脚本，则应该使用相应的数据类型。请参见“SQL-Java 数据类型”一节第 498 页和“SQL-.NET 数据类型”一节第 556 页。

在 SQL 脚本中，可以使用名称或问号指定事件参数，但不能在一个脚本中混合使用名称和问号。如果使用问号，则参数必须按照如下所示的顺序并且仅当没有指定任何后继参数时才是可选的（例如，如果您想使用参数 2，则必须使用参数 1）。如果使用命名参数，则可以按照任何顺序指定任意参数子集。

SQL 脚本的参数名称	说明	顺序
s.file_authentication_code	INTEGER。必需。这是一个 INOUT 参数。它指示验证是否完全成功。 如果此值为 1000-1999，则允许文件传输。如果此值为 2000-2999，则不允许文件传输。	1
s.filename	VARCHAR(128)。此可选的参数是要进行验证的所传输文件的名称。不要包括路径。该文件必须位于使用 mlsrv11 -ftr 选项指定的根传输目录中，或在一个自动创建的子目录中。	2
s.remote_id	VARCHAR(128)。MobiLink 远程 ID。只有在使用命名参数时才能引用远程 ID。	不适用
s.username	VARCHAR(128)。MobiLink 用户名。	3

注释

在允许使用 mlfiletransfer 实用程序或 MLFileTransfer 方法进行任何文件传输之前，MobiLink 服务器执行此事件。它在用户使用常规验证进行验证后执行。如果未定义此脚本，则允许文件传输。

只有 UltraLite 客户端才能使用 MLFileTransfer 方法。

另请参见

- “添加和删除脚本”一节第 306 页
- “-ftr 选项”一节第 67 页
- “MobiLink 文件传输实用程序 (mlfiletransfer)”一节 《MobiLink - 客户端管理》
- UltraLite： “使用 MobiLink 文件传输”一节 《UltraLite - 数据库管理和参考》
- “在脚本中使用远程 ID 和 MobiLink 用户名”一节 《MobiLink - 客户端管理》

Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

authenticate_parameters 连接事件

接收从远程数据库发送来的值，这些值使您能在用户 ID 和口令之外另外获得一层验证。这些值还可以用于任意地自定义每个同步。

参数

在下表中，说明部分提供 SQL 数据类型。如果您使用 Java 或 .NET 编写脚本，则应该使用相应的数据类型。请参见“SQL-Java 数据类型”一节第 498 页和“SQL-.NET 数据类型”一节第 556 页。

在 SQL 脚本中，可以使用名称或问号指定事件参数，但不能在一个脚本中混合使用名称和问号。如果使用问号，则参数必须按照如下所示的顺序并且仅当没有指定任何后继参数时才是可选的（例如，如果您想使用参数 2，则必须使用参数 1）。如果使用命名参数，则可以按照任何顺序指定任意参数子集。

SQL 脚本的参数名称	说明	顺序
s.authentication_status	INTEGER。这是一个 INOUT 参数。	1
s.remote_id	VARCHAR(128)。MobiLink 远程 ID。只有在使用命名参数时才能引用远程 ID。	不适用
s.username	VARCHAR(128)。MobiLink 用户名。	2
a.*N*（一个或多个）	VARCHAR(128)。例如，已命名的参数可以是 `a.1 a.2`。	3...

参数说明

- **authentication_status** authentication_status 参数是必需的。它指示验证是否完全成功，可设置为以下值之一：

返回值	authentication_status	说明
V <= 1999	1000	验证成功。
1999 < V <= 2999	2000	验证成功，但口令即将失效。
2999 < V <= 3999	3000	验证失败，因为口令已失效。
3999 < V <= 4999	4000	验证失败。
4999 < V <= 5999	5000	验证失败，因为用户已经在进行同步。
5999 < V	4000	如果返回值大于 5999，则 MobiLink 将其解释为返回值 4000（验证失败）。

- **username** 此参数为 MobiLink 用户名。VARCHAR(128)。
- **remote_ID** MobiLink 远程 ID。只有在使用已命名的参数时才能引用远程 ID。

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

请参见“在脚本中使用远程 ID 和 MobiLink 用户名”一节 《MobiLink - 客户端管理》。

- **remote_parameters** 远程参数的数目必须与所需数目相匹配，否则会产生错误。如果从客户端发送参数但没有对应于此事件的任何脚本，也将发生错误。

注释

从 SQL Anywhere 和 UltraLite 客户端都可以发送字符串（或字符串形式的参数）。这使您能在用户 ID 和口令之外另外获得一层验证。这还意味着，您可以在验证过程中根据参数值自定义同步，并在预同步阶段完成此项工作。

MobiLink 服务器在每个同步开始时执行此事件。它与 authenticate_user 事件在同一个事务中执行。

您可以使用此事件并用自定义机制代替内置的 MobiLink 验证机制。您可能希望调用 DBMS 的验证机制，或者实现 MobiLink 的内置机制中没有的功能。

如果调用了 authenticate_user 或 authenticate_user_hashed 脚本且脚本返回一个错误，则不调用此事件。

authenticate_parameters 事件的 SQL 脚本必须作为存储过程实现。

另请参见

- “脚本参数”一节第 299 页
- “添加和删除脚本”一节第 306 页
- “验证参数”一节第 301 页
- “MobiLink 用户” 《MobiLink - 客户端管理》
- “在脚本中使用远程 ID 和 MobiLink 用户名”一节 《MobiLink - 客户端管理》
- “自定义用户验证”一节 《MobiLink - 客户端管理》
- “authenticate_user 连接事件”一节第 335 页
- “authenticate_user_hashed 连接事件”一节第 339 页
- “begin_synchronization 连接事件”一节第 358 页
- dbmlsync： “-ap 选项”一节 《MobiLink - 客户端管理》
- UltraLite： “Authentication Parameters 同步参数”一节 《UltraLite - 数据库管理和参考》 和 “Number of Authentication Parameters 参数”一节 《UltraLite - 数据库管理和参考》

示例

对于 UltraLite 远程数据库，请使用 ul_synch_info 结构中的 num_auth_parms 和 auth_parms 字段传递参数。num_auth_parms 是参数数目的计数（0 至 255），auth_parms 是指向字符串数组的指针。为防止以纯文本格式查看这些字符串，它们与口令采用相同的方式进行发送。如果 num_auth_parms 为 0，请将 auth_parms 设置为空。以下是在 UltraLite 中传递参数的一个示例：

```
ul_char * Params[ 3 ] = { UL_TEXT( "param1" ),
   UL_TEXT( "param2" ), UL_TEXT( "param3" ) };

...
info.num_auth_parms = 3;
info.auth_parms = Params;
```

对于 SQL Anywhere 远程数据库，请使用 dbmlsync -ap 选项，通过逗号分隔的列表传递参数。例如，以下命令行传递三个参数：

```
dbmlsync -ap "param1,param2,param3"
```

在此示例中，authenticate_parameters 脚本可以为：

```
CALL my_auth_parm (
  {ml s.authentication_status},
  {ml s.remote_id},
  {ml s.username},
  {ml a.1},
  {ml a.2},
  {ml a.3}
)
```

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

authenticate_user 连接事件

实现自定义用户验证。

参数

在下表中，说明部分列出 SQL 数据类型。如果您使用 Java 或 .NET 编写脚本，则应该使用相应的数据类型。请参见“SQL-Java 数据类型”一节第 498 页和“SQL-.NET 数据类型”一节第 556 页。

在 SQL 脚本中，可以使用名称或问号指定事件参数，但不能在一个脚本中混合使用名称和问号。如果使用问号，则参数必须按照如下所示的顺序并且仅当没有指定任何后继参数时才是可选的（例如，如果您想使用参数 2，则必须使用参数 1）。如果使用命名参数，则可以按照任何顺序指定任意参数子集。

SQL 脚本的参数名称	说明	顺序
s.authentication_status	INTEGER。这是一个 INOUT 参数。	1
s.remote_id	VARCHAR(128)。MobiLink 远程 ID。只有在使用命名参数时才能引用远程 ID。	不适用
s.username	VARCHAR(128)。MobiLink 用户名。	2
s.password	VARCHAR(128)。用于验证目的的口令。如果用户不提供口令，则值为空。	3
s.new_password	VARCHAR(128)。如果它正被用于重置用户口令，则为新口令。如果用户不更改口令，则此值为空。	4

缺省操作

使用 MobiLink 内置用户验证机制。

注释

MobiLink 服务器在每个同步开始时执行此事件。该事件在 begin_synchronization 事务之前的事务中执行。

您可以使用此事件并用自定义机制代替内置的 MobiLink 验证机制。您可能希望调用 DBMS 的验证机制，或者实现 MobiLink 的内置机制中没有的功能，如口令失效日期或最小口令长度等。

authenticate_user 事件使用如下参数：

- **authentication_status** authentication_status 参数是必需的。它指示验证是否完全成功，可设置为以下值之一：

返回值	authentication_status	说明
V <= 1999	1000	验证成功。

返回值	authentication_status	说明
1999 < V <= 2999	2000	验证成功：口令即将失效。
2999 < V <= 3999	3000	验证失败：口令已失效。
3999 < V <= 4999	4000	验证失败。
4999 < V <= 5999	5000	验证失败，因为用户已经在进行同步。
5999 < V	4000	如果返回值大于 5999，则 MobiLink 将其解释为返回值 4000。

- **username** 此可选参数为 MobiLink 用户名。

  请参见“在脚本中使用远程 ID 和 MobiLink 用户名”一节 《MobiLink - 客户端管理》。
- **remote_id** MobiLink 远程 ID。只有在使用已命名的参数时才能引用远程 ID。
- **password** 可选参数，指示用于进行验证的口令。如果用户不提供口令，则为空。
- **new_password** 此可选参数指示新口令。如果用户不更改口令，则为空。

authenticate_user 事件的 SQL 脚本必须作为存储过程实现。

如果定义了两个验证脚本，并且两个脚本返回不同的 authentication_status 代码，则会使用其中较大的值。

authenticate_user 脚本在事务中与所有验证脚本一起执行。始终提交此事务。

可为 authenticate_user 事件使用一些预定义的脚本，以简化使用 LDAP、IMAP 和 POP3 服务器时的验证。

请参见“向外部服务器验证”一节 《MobiLink - 客户端管理》。

另请参见

- “脚本参数”一节第 299 页
- “添加和删除脚本”一节第 306 页
- “MobiLink 用户” 《MobiLink - 客户端管理》
- “在脚本中使用远程 ID 和 MobiLink 用户名”一节 《MobiLink - 客户端管理》
- “自定义用户验证”一节 《MobiLink - 客户端管理》
- “向外部服务器验证”一节 《MobiLink - 客户端管理》
- “authenticate_user_hashed 连接事件”一节第 339 页
- “authenticate_parameters 连接事件”一节第 332 页
- “modify_user 连接事件”一节第 436 页
- “begin_synchronization 连接事件”一节第 358 页

SQL 示例

典型的 authenticate_user 脚本是对某一存储过程的调用。该调用中参数的顺序必须与上面的顺序匹配。以下示例使用 ml_add_connection_script 将事件指派给名为 my_auth 的存储过程。

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

```
CALL ml_add_connection_script(
   'ver1', 'authenticate_user', 'call my_auth ( {ml s.username} )'
)
```

以下 SQL Anywhere 存储过程只使用用户名进行验证—不进行口令检查。该过程只检查所提供的用户名是否为 ULEmployee 表中列出的一个员工 ID。

```
CREATE PROCEDURE my_auth( in @user_name varchar(128) )
BEGIN
  IF EXISTS
  ( SELECT * FROM ulemployee
    WHERE emp_id = @user_name )
  THEN
    MESSAGE 'OK' type info to client;
    RETURN 1000;
  ELSE
    MESSAGE 'Not OK' type info to client;
    RETURN 4000;
  END IF
END
```

Java 示例

以下对 MobiLink 系统过程的调用在同步脚本版本 ver1 时将名为 authenticateUser 的 Java 方法注册为 authenticate_user 事件的脚本。此语法用于 SQL Anywhere 统一数据库。

```
CALL ml_add_java_connection_script(
   'ver1', 'authenticate_user',
   'ExamplePackage.ExampleClass.authenticateUser'
)
```

以下是 Java 方法 authenticateUser 示例。它调用检查用户口令（并在需要时更改用户口令）的 Java 方法。

```
public String authenticateUser(
  ianywhere.ml.script.InOutInteger authStatus,
  String user,
  String pwd,
  String newPwd )
  throws java.sql.sqlException {
  // A real authenticate_user handler would
  // handle more authentication code states.
  _curUser = user;
  if( checkPwd( user, pwd ) ) {
    // Authentication successful.
    if( newPwd != null ) {
      // Password is being changed.
      if( changePwd( user, pwd, newPwd ) ) {
        // Authentication OK and password change OK.
        // Use custom code.
        authStatus.setValue( 1001 );
      } else {
        // Authentication OK but password
        // change failed. Use custom code.
        java.lang.System.err.println( "user: "
         + user + " pwd change failed!" );
        authStatus.setValue( 1002 );
      }
    } else {
      authStatus.setValue( 1000 );
    }
  } else {
```

```
      // Authentication failed.
      authStatus.setValue( 4000 );
    }
    return ( null );
  }
```

.NET 示例

以下对 MobiLink 系统过程的调用在同步脚本版本 ver1 时将名为 AuthUser 的 .NET 方法注册为 authenticate_user 连接事件的脚本。此语法用于 SQL Anywhere 统一数据库。

```
CALL ml_add_dnet_connection_script(
   'ver1', 'authenticate_user',
   'TestScripts.Test.AuthUser'
)
```

以下是 .NET 方法 AuthUser 示例。它调用检查用户口令（并在需要时更改用户口令）的 .NET 方法。

```
public string AuthUser(
  ref int authStatus,
  string user,
  string pwd,
  string newPwd ) {
  // A real authenticate_user handler would
  // handle more authentication code states.
  _curUser = user;
  if( CheckPwd( user, pwd ) ) {
    // Authentication successful.
    if( newPwd != null ) {
      // Password is being changed.
      if( ChangePwd( user, pwd, newPwd ) ) {
        // Authentication OK and password change OK.
        // Use custom code.
        authStatus = 1001;
      } else {
        // Authentication OK but password
        // change failed. Use custom code.
        System.Console.WriteLine( "user: "
         + user + " pwd change failed!" );
        authStatus = 1002;
      }
    } else {
      authStatus = 1000 ;
    }
  } else {
    // Authentication failed.
    authStatus = 4000;
  }
  return ( null );
}
```

有关使用 .NET 中 C# 编写的 authenticate_user 脚本的更详细示例，请参见“.NET 同步示例”一节 第 565 页。

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

authenticate_user_hashed 连接事件

实现自定义用户验证机制。

参数

在下表中，说明部分提供 SQL 数据类型。如果您使用 Java 或 .NET 编写脚本，则应该使用相应的数据类型。请参见“SQL-Java 数据类型”一节第 498 页和“SQL-.NET 数据类型”一节第 556 页。

在 SQL 脚本中，可以使用名称或问号指定事件参数，但不能在一个脚本中混合使用名称和问号。如果使用问号，则参数必须按照如下所示的顺序并且仅当没有指定任何后继参数时才是可选的（例如，如果您想使用参数 2，则必须使用参数 1）。如果使用命名参数，则可以按照任何顺序指定任意参数子集。

SQL 脚本的参数名称	说明	顺序
s.authentication_status	INTEGER。这是一个 INOUT 参数。	1
s.remote_id	VARCHAR(128)。MobiLink 远程 ID。只有在使用命名参数时才能引用远程 ID。	不适用
s.username	VARCHAR(128)。MobiLink 用户名。	2
s.hashed_password	BINARY(20)。如果用户不提供口令，则值为空。	3
s.hashed_new_password	BINARY(20)。如果此事件不是正用于更改用户口令，则此值为空。	4

缺省操作

使用 MobiLink 内置用户验证机制。

注释

该事件与 authenticate_user 相同，但是其口令是以散列形式传递的，与存储在 ml_user.hashed_password 列中口令的形式相同。以散列形式传递口令可以增强安全性。

将使用单向散列。单向散列接受口令并将其转换为一个字节序列，该序列（实质上）对每个可能的口令是唯一的。单向散列允许进行口令验证，而不必将实际的口令存储于统一数据库中。

在一个用户的验证序列过程中，可以调用此脚本多次。

如果同时定义了 authenticate_user 和 authenticate_user_hashed，并且两个脚本返回不同的 authentication_status 代码，则会使用其中较大的值。

另请参见

- “脚本参数”一节第 299 页
- “添加和删除脚本”一节第 306 页
- “MobiLink 用户” 《MobiLink - 客户端管理》
- “在脚本中使用远程 ID 和 MobiLink 用户名”一节 《MobiLink - 客户端管理》
- “自定义用户验证”一节 《MobiLink - 客户端管理》
- “authenticate_user 连接事件”一节第 335 页
- “authenticate_parameters 连接事件”一节第 332 页

SQL 示例

典型的 authenticate_user_hashed 脚本是对某个存储过程的调用。该调用中参数的顺序必须与上面的顺序匹配。以下示例调用 ml_add_connection_script，将事件指派给名为 my_auth 的存储过程。

```
CALL ml_add_connection_script(
   'ver1', 'authenticate_user_hashed',
   'call my_auth (
      {ml s.authentication_status},
      {ml s.username},
      {ml s.hashed_password})'
)
```

以下 SQL Anywhere 存储过程使用用户名和口令进行验证。该过程只检查所提供的用户名是否为 ULEmployee 表中列出的一个员工 ID。该过程假定 Employee 表中有一个名为 hashed_pwd 的 binary(20) 列。

```
CREATE PROCEDURE my_auth(
  inout @authentication_status integer,
  in @user_name varchar(128),
  in @hpwd binary(20) )
BEGIN
  IF EXISTS
  ( SELECT * FROM ulemployee
    WHERE emp_id = @user_name
      and hashed_pwd = @hpwd )
  THEN
    message 'OK' type info to client;
    RETURN 1000;
  ELSE
    message 'Not OK' type info to client;
    RETURN 4000;
  END IF
END
```

Java 示例

以下对 MobiLink 系统过程的调用在同步脚本版本 ver1 时将名为 authUserHashed 的 Java 方法注册为 authenticate_user_hashed 事件的脚本。

```
CALL ml_add_java_connection_script(
   'ver1', 'authenticate_user_hashed',
   'ExamplePackage.ExampleClass.authUserHashed')
```

以下是 Java 方法 authUserHashed 示例。它调用检查用户口令（并在需要时更改用户口令）的 Java 方法。

```
public String authUserHashed(
  ianywhere.ml.script.InOutInteger authStatus,
```

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

```
  String user,
  byte pwd[],
  byte newPwd[] )
  throws java.sql.SQLException {
  // A real authenticate_user_hashed handler
  // would handle more auth code states.
  _curUser = user;
  if( checkPwdHashed( user, pwd ) ) {
    // Authorization successful.
    if( newPwd != null ) {
      // Password is being changed.
      if( changePwdHashed( user, pwd, newPwd ) ) {
        // Authorization OK and password change OK.
        // Use custom code.
        authStatus.setValue( 1001 );
      } else {
        // Auth OK but password change failed.
        // Use custom code
        java.lang.System.err.println( "user: " + user
          + " pwd change failed!" );
        authStatus.setValue( 1002 );
      }
      } else {
        authStatus.setValue( 1000 );
      }
    } else {
      // Authorization failed.
      authStatus.setValue( 4000 );
    }
  return ( null );
}
```

.NET 示例

以下对 MobiLink 系统过程的调用在同步脚本版本 ver1 时将名为 AuthUserHashed 的 .NET 方法注册为 authenticate_user_hashed 连接事件的脚本。此语法用于 SQL Anywhere 统一数据库。

```
CALL ml_add_dnet_connection_script(
   'ver1',
   'authenticate_user_hashed',
   'TestScripts.Test.AuthUserHashed'
)
```

以下是 .NET 方法 AuthUserHashed 示例。

```
public string AuthUserHashed(
  ref int authStatus,
  string user,
  byte[] pwd,
  byte[] newPwd ) {
  // A real authenticate_user_hashed handler
  // would handle more auth code states.
  _curUser = user;
  if( CheckPwdHashed( user, pwd ) ) {
    // Authorization successful.
    if( newPwd != null ) {
      // Password is being changed.
      if( ChangePwdHashed( user, pwd, newPwd ) ) {
      // Authorization OK and password change OK.
      // Use custom code.
      authStatus = 1001;
    } else {
      // Auth OK but password change failed.
```

```
            // Use custom code
            System.Console.WriteLine( "user: " + user
              + " pwd change failed!" );
            authStatus = 1002;
          }
            } else {
              authStatus = 1000;
            }
          } else {
            // Authorization failed.
            authStatus = 4000;
          }
          return ( null );
      }
```

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

begin_connection 连接事件

在 MobiLink 服务器连接到统一数据库服务器时调用。

参数

无。

缺省操作

无。

注释

MobiLink 同步过程在收到同步请求时打开连接。当应用程序创建或重新创建与 MobiLink 服务器的连接时，MobiLink 服务器将会临时分配一个与数据库服务器的连接，该连接在整个同步过程中持续。如果 MobiLink 服务器正在使用池中的一个连接，则可能不会调用此事件。

注意
Java 或 .NET 中通常不使用此脚本，这是因为您将在此类实例中使用成员变量而不是数据库变量，并在构造函数中准备成员。

另请参见

- “添加和删除脚本”一节第 306 页
- “end_connection 连接事件”一节第 382 页
- “-cn 选项”一节第 52 页
- “-w 选项”一节第 100 页

SQL 示例

以下的 SQL 脚本在 SQL Anywhere 统一数据库中运行。该脚本创建了两个变量，一个代表 last_download 时间戳，另一个代表雇员 ID。

```
CALL ml_add_connection_script(
    'custdb',
    'begin_connection',
    'create variable @LastDownload timestamp;
    create variable @EmployeeID integer;')
```

begin_connection_autocommit 连接事件

打开自动提交。

参数

无。

缺省操作

关闭自动提交。

注释

当 MobiLink 服务器连接到统一数据库时，它会关闭自动提交，以便在发生错误时能够回退上载和下载。

但是，如果使用的是 Adaptive Server Enterprise 统一数据库，则必须在打开自动提交的情况下才能执行 DDL 功能，如创建临时表。如果使用的是 Adaptive Server Enterprise 统一数据库，可在 begin_connection_autocommit 事件中运行 DDL 命令。事件完成时，将关闭自动提交。

必须编写 Begin_connection_autocommit 脚本，以便能够重复使用它们。原因是，如果发生错误或死锁，MobiLink 服务器必须能够重新尝试该脚本（因为无法回退它）。

只有为事件定义了脚本后，此事件才会执行。

另请参见

- “添加和删除脚本”一节第 306 页

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

begin_download 连接事件

紧接在 MobiLink 服务器开始对下载进行准备之前处理任何语句。

参数

在下表中，说明部分提供 SQL 数据类型。如果您使用 Java 或 .NET 编写脚本，则应该使用相应的数据类型。请参见“SQL-Java 数据类型”一节第 498 页和“SQL-.NET 数据类型”一节第 556 页。

在 SQL 脚本中，可以使用名称或问号指定事件参数，但不能在一个脚本中混合使用名称和问号。如果使用问号，则参数必须按照如下所示的顺序并且仅当没有指定任何后继参数时才是可选的（例如，如果您想使用参数 2，则必须使用参数 1）。如果使用命名参数，则可以按照任何顺序指定任意参数子集。

SQL 脚本的参数名称	说明	顺序
s.last_download	TIMESTAMP。任何已同步表的上一次下载时间。	1
s.remote_id	VARCHAR(128)。MobiLink 远程 ID。只有在使用命名参数时才能引用远程 ID。	不适用
s.username	VARCHAR(128)。MobiLink 用户名。	2

缺省操作

无。

注释

MobiLink 服务器执行该事件是对下载的信息进行处理的第一步。下载信息是在单独的一个事务中进行处理的。执行此事件是此事务中的第一步操作。

另请参见

- “脚本参数”一节第 299 页
- “添加和删除脚本”一节第 306 页
- “end_download 连接事件”一节第 384 页
- “在脚本中使用远程 ID 和 MobiLink 用户名”一节 《MobiLink - 客户端管理》
- “在脚本中使用上次下载时间”一节第 122 页

SQL 示例

下面的示例调用 ml_add_connection_script，将事件指派给名为 SetDownloadParameters 的存储过程。

```
CALL ml_add_connection_script (
   'Lab',
   'begin_download',
   'CALL SetDownloadParameters( {ml s.last_table_download}, {ml
s.username} )' )
```

Java 示例

以下对 MobiLink 系统过程的调用在同步脚本版本 ver1 时将名为 beginDownloadConnection 的 Java 方法注册为 begin_download 连接事件的脚本。

```
CALL ml_add_java_connection_script(
   'example_ver',
   'begin_download',
   'ExamplePackage.ExampleClass.beginDownloadConnection' )
```

以下是 Java 方法 beginDownloadConnection 示例。它调用一个 Java 方法 (prepDeleteTables)，该方法使用先前设置的 JDBC 同步准备删除表。

```
public String beginDownloadConnection(
  Timestamp ts,
  String user )
  throws java.sql.SQLException {
  prepDeleteTables ( _syncConn, ts, user );
  return ( null );
}
```

.NET 示例

以下对 MobiLink 系统过程的调用在同步脚本版本 ver1 时将名为 BeginDownload 的 .NET 方法注册为 begin_download 连接事件的脚本。

```
CALL ml_add_dnet_connection_script(
  'ver1',
  'begin_download',
  'TestScripts.Test.BeginDownload'
)
```

以下是 .NET 方法 BeginDownload 示例。它调用一个 .NET 方法 (prepDeleteTables)，该方法使用先前设置的 JDBC 同步准备删除表。

```
public string BeginDownload(
  DateTime timestamp,
  string user ) {
  prepDeleteTables ( _syncConn, ts, user );
  return ( null );
}
```

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

begin_download 表事件

紧接在准备下载插入、更新和删除操作之前处理与某一特定的表有关的语句。

参数

在下表中，说明部分提供 SQL 数据类型。如果您使用 Java 或 .NET 编写脚本，则应该使用相应的数据类型。请参见“SQL-Java 数据类型”一节第 498 页和“SQL-.NET 数据类型”一节第 556 页。

在 SQL 脚本中，可以使用名称或问号指定事件参数，但不能在一个脚本中混合使用名称和问号。如果使用问号，则参数必须按照如下所示的顺序并且仅当没有指定任何后继参数时才是可选的（例如，如果您想使用参数 2，则必须使用参数 1）。如果使用命名参数，则可以按照任何顺序指定任意参数子集。

SQL 脚本的参数名称	说明	顺序
s.last_table_download	TIMESTAMP。表的上一次下载时间。	1
s.remote_id	VARCHAR(128)。MobiLink 远程 ID。只有在使用命名参数时才能引用远程 ID。	不适用
s.username	VARCHAR(128)。MobiLink 用户名。	2
s.table	VARCHAR(128)。表名。	3

缺省操作

无。

注释

MobiLink 服务器执行此事件是为特定的表准备下载信息的第一步。下载信息是在单独的一个事务中进行准备的。执行此事件是此事务中的第一步表特定的操作。

您可以为远程数据库的每个表编写一个 begin_download 脚本。只有在对表进行同步时才会调用相应的脚本。

另请参见

- “脚本参数”一节第 299 页
- “添加和删除脚本”一节第 306 页
- “end_download 表事件”一节第 386 页
- “在脚本中使用远程 ID 和 MobiLink 用户名”一节 《MobiLink - 客户端管理》
- “在脚本中使用上次下载时间”一节第 122 页

SQL 示例

以下对 MobiLink 系统过程 ml_add_table_script 的调用将调用 BeginTableDownload 过程。此语法用于 SQL Anywhere 11 统一数据库。

```
CALL ml_add_table_script(
   'version1',
```

```
    'Leads',
    'begin_download',
    'CALL BeginTableDownLoad(
      {ml s.last_table_download},
      {ml s.username},
      {ml s.table} )' );
```

以下 SQL 语句创建 BeginTableDownload 过程。

```
CREATE PROCEDURE BeginTableDownload(
   LastDownload timestamp,
   MLUser varchar(128),
   TableName varchar(128) )
BEGIN
  EXECUTE IMMEDIATE 'update ' || TableName ||
' set last_download_check = CURRENT TIMESTAMP
  WHERE Owner = ' ||MLUser;
END
```

Java 示例

以下对 MobiLink 系统过程的调用在同步脚本版本 ver1 时将名为 beginDownloadTable 的 Java 方法注册为 begin_download 表事件的脚本。

```
CALL ml_add_java_table_script(
   'ver1',
   'table1',
   'begin_download',
   'ExamplePackage.ExampleClass.beginDownloadTable' )
```

以下是 Java 方法 beginDownloadTable 示例。它保存当前表的名称以用于以后的方法调用。

```
public String beginDownloadTable(
  Timestamp ts,
  String user,
  String table ) {
  _curTable = table;
  return ( null );
}
```

.NET 示例

以下对 MobiLink 系统过程的调用在同步脚本版本 ver1 和表 table1 时将名为 BeginTableDownload 的 .NET 方法注册为 begin_download 表事件的脚本。

```
CALL ml_add_dnet_table_script(
  'ver1', 'table1', 'begin_download',
  'TestScripts.Test.BeginTableDownload'
)
```

以下是 .NET 方法 BeginDownload 示例。它保存当前表的名称以用于以后的方法调用。

```
public string BeginTableDownload(
  DateTime timestamp,
  string user,
  string table ) {
  _curTable = table;
  return ( null );
}
```

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

begin_download_deletes 表事件

紧接在读取将要从远程数据库的指定表中删除的行的列表之前，处理与该表有关的语句。

参数

在下表中，说明部分提供 SQL 数据类型。如果您使用 Java 或 .NET 编写脚本，则应该使用相应的数据类型。请参见“SQL-Java 数据类型”一节第 498 页和“SQL-.NET 数据类型”一节第 556 页。

在 SQL 脚本中，可以使用名称或问号指定事件参数，但不能在一个脚本中混合使用名称和问号。如果使用问号，则参数必须按照如下所示的顺序并且仅当没有指定任何后继参数时才是可选的（例如，如果您想使用参数 2，则必须使用参数 1）。如果使用命名参数，则可以按照任何顺序指定任意参数子集。

SQL 脚本的参数名称	说明	顺序
s.last_table_download	TIMESTAMP。表的上一次下载时间。	1
s.remote_id	VARCHAR(128)。MobiLink 远程 ID。只有在使用命名参数时才能引用远程 ID。	不适用
s.username	VARCHAR(128)。MobiLink 用户名。	2
s.table	VARCHAR(128)。表名。	3

缺省操作

无。

注释

此事件将在读取将要从远程数据库的指定表中删除行的列表之前执行。

您可以为远程数据库的每个表编写一个 begin_download_deletes 脚本。

另请参见

- “脚本参数”一节第 299 页
- “添加和删除脚本”一节第 306 页
- “begin_download_rows 表事件”一节第 352 页
- “end_download_rows 表事件”一节第 391 页
- “在脚本中使用远程 ID 和 MobiLink 用户名”一节 《MobiLink - 客户端管理》
- “在脚本中使用上次下载时间”一节第 122 页

SQL 示例

为了减少远程存储的数据量，您可以使用此事件标记在执行 download_delete_cursor 时将要删除的数据。下面的示例标记删除超过 10 周的来自远程设备的销售线索。该示例可以在 SQL Anywhere 11 数据库上使用。

以下对 MobiLink 系统过程的调用在同步脚本版本 ver1 时将 BeginDownloadDeletes 存储过程指派给 begin_download_deletes 事件。

Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

```
CALL ml_add_table_script (
  'ver1',
  'Leads',
  'begin_download_deletes',
  'CALL BeginDownloadDeletes (
    {ml s.last_table_download},
    {ml s.username},
    {ml s.table})' );
```

以下 SQL 语句创建 BeginDownloadDeletes 存储过程。

```
CREATE PROCEDURE BeginDownloadDeletes(
  LastDownload timestamp,
  MLUser varchar(128),
  TableName varchar(128) )
BEGIN
  execute immediate 'update ' || TableName ||
  ' set delete_flag = 1 where
  days(creation_time, CURRENT DATE) > 70 and Owner = '
  || MLUser;
END;
```

Java 示例

以下对 MobiLink 系统过程的调用在同步脚本版本 ver1 时将名为 beginDownloadDeletes 的 Java 方法注册为 begin_download_deletes 表事件的脚本。

```
CALL ml_add_java_table_script(
   'ver1',
   'table1',
   'begin_download_deletes',
   'ExamplePackage.ExampleClass.beginDownloadDeletes' )
```

示例 Java 方法 beginDownloadDeletes 保存当前表的名称以用于以后的方法调用。

```
public String beginDownloadDeletes (
  Timestamp ts,
  String user,
  String table ) {
  _curTable = table;
  return ( null );
}
```

.NET 示例

以下对 MobiLink 系统过程的调用在同步脚本版本 ver1 和表 table1 时将名为 BeginDownloadDeletes 的 .NET 方法注册为 begin_download_deletes 表事件的脚本。

```
CALL ml_add_dnet_table_script (
  'ver1', 'table1', 'begin_download_deletes',
  'TestScripts.Test.BeginDownloadDeletes'
)
```

示例 .NET 方法 BeginDownloadDeletes 保存当前表的名称以用于以后的方法调用。

```
public string BeginDownloadDeletes(
  DateTime timestamp,
  string user,
  string table ) {
  _curTable = table;
```

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

```
    return ( null );
}
```

begin_download_rows 表事件

紧接在读取将要在远程数据库的指定表中插入或更新行的列表之前，处理与该特定表有关的语句。

参数

在下表中，说明部分提供 SQL 数据类型。如果您使用 Java 或 .NET 编写脚本，则应该使用相应的数据类型。请参见“SQL-Java 数据类型”一节第 498 页和“SQL-.NET 数据类型”一节第 556 页。

在 SQL 脚本中，可以使用名称或问号指定事件参数，但不能在一个脚本中混合使用名称和问号。如果使用问号，则参数必须按照如下所示的顺序并且仅当没有指定任何后继参数时才是可选的（例如，如果您想使用参数 2，则必须使用参数 1）。如果使用命名参数，则可以按照任何顺序指定任意参数子集。

SQL 脚本的参数名称	说明	顺序
s.last_table_download	TIMESTAMP。表的上一次下载时间。	1
s.remote_id	VARCHAR(128)。MobiLink 远程 ID。只有在使用命名参数时才能引用远程 ID。	不适用
s.username	VARCHAR(128)。MobiLink 用户名。	2
s.table	VARCHAR(128)。表名。	3

缺省操作

无。

注释

此事件将在读取将要在远程数据库的指定表中插入或更新行的流之前执行。

您可以为远程数据库的每个表编写一个 begin_download_rows 脚本。

另请参见

- “脚本参数”一节第 299 页
- “添加和删除脚本”一节第 306 页
- “begin_download_deletes 表事件”一节第 349 页
- “end_download_deletes 表事件”一节第 388 页
- “在脚本中使用远程 ID 和 MobiLink 用户名”一节 《MobiLink - 客户端管理》
- “在脚本中使用上次下载时间”一节第 122 页

SQL 示例

您可以使用 begin_download_rows 表事件标记不再想为此表下载的行。以下示例存档超过七天的销售资料。

以下对 MobiLink 系统过程的调用为 begin_download_rows 事件注册 BeginDownloadRows 存储过程。

```
CALL ml_add_table_script(
  'version1',
```

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

```
  'Leads',
  'begin_download_rows',
  'CALL BeginDownloadRows (
    {ml s.last_table_download},
    {ml s.username},
    {ml s.table})' ); )
```

以下 SQL 语句创建 BeginDowloadRows 存储过程。

```
CREATE PROCEDURE BeginDownloadRows (
  LastDownload timestamp, MLUser varchar(128),
  TableName varchar(128) )
BEGIN
  execute immediate 'update ' || TableName ||
  ' set download_flag = 0 where
  days(creation_time, CURRENT DATE) > 7 and Owner = '
  || MLUser;
END;
```

Java 示例

以下对 MobiLink 系统过程的调用在同步脚本版本 ver1 时将名为 beginDownloadRows 的 Java 方法注册为 begin_download_rows 表事件的脚本。此语法用于 SQL Anywhere 统一数据库。

```
CALL ml_add_java_table_script(
   'ver1',
   'table1',
   'begin_download_rows',
   'ExamplePackage.ExampleClass.beginDownloadRows' )
```

以下是 Java 方法 beginDownloadRows 示例。它使用表和用户生成 UPDATE 语句，以便 MobiLink 执行。

```
public String beginDownloadRows(
  Timestamp ts,
  String user,
  String table ) {
  return( "update " + table + " set download_flag = 0 "
   + " where days(creation_time, CURRENT DATE) > 7 " +
   " and Owner = '" + user + "'" );
}
```

.NET 示例

以下对 MobiLink 系统过程的调用在同步脚本版本 ver1 和表 table1 时将名为 BeginDownloadRows 的 .NET 方法注册为 begin_download_rows 表事件的脚本。

```
CALL ml_add_dnet_table_script(
  'ver1', 'table1', 'begin_download_rows',
  'TestScripts.Test.BeginDownloadRows'
)
```

以下是 .NET 方法 BeginDownloadRows 示例。它使用表和用户生成 UPDATE 语句，以便 MobiLink 执行。

```
public string BeginDownloadRows(
  DateTime timestamp,
  string user,
  string table ) {
  return( "update " + table + " set download_flag = 0 "
   + " where days(creation_time, CURRENT DATE) > 7 " +
```

```
            " and Owner = '" + user + "'" );
        }
```

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

begin_publication 连接事件

提供有关正被同步的发布的有用信息。还可以使用此脚本来管理基于文件的下载的世代号。

参数

在下表中，说明部分提供 SQL 数据类型。如果您使用 Java 或 .NET 编写脚本，则应该使用相应的数据类型。请参见“SQL-Java 数据类型”一节第 498 页和“SQL-.NET 数据类型”一节第 556 页。

SQL 脚本的参数名称	说明	顺序
s.generation_number	INTEGER。这是一个 INOUT 参数。如果部署不使用基于文件的下载，则可以忽略此参数。缺省值为 1。	1
s.remote_id	VARCHAR(128)。MobiLink 远程 ID。只有在使用命名参数时才能引用远程 ID。	不适用
s.username	VARCHAR(128)。MobiLink 用户名。	2
s.publication_name	VARCHAR(128)。发布的名称。	3
s.last_publication_upload	TIMESTAMP。此发布上次成功上载的时间。	4
s.last_publication_download	TIMESTAMP。发布的上一次下载时间。	5
s.subscription_id	VARCHAR(128)。预订 ID。	6

缺省操作

缺省世代号是 1。如果未为此事件定义脚本，则发送到远程的世代号始终为 1。

注释

此事件使您可以根据当前正在同步的发布来设计同步逻辑。此事件与 begin_synchronization 事件在相同的事务中并在 begin_synchronization 事件之后被调用。对于当前正在同步的每个发布，将调用此事件一次。

此事件的一个潜在用途是根据所用的发布控制下载内容。例如，假设有一个表，它同时属于优先级发布 (PriorityPub) 和所有表的发布 (AllTablesPub)。begin_publication 事件的脚本可将这些发布名存储在 Java 类或 SQL 变量或包中。然后，根据正在同步的发布是 PriorityPub 还是 AllTablesPub，下载脚本可以具有不同的行为方式。

如果一个 UltraLite 远程数据库在与 UL_SYNC_ALL 进行同步，则将使用 'unknown' 这一名称调用此事件一次。

世代号

generation_number 参数专用于基于文件的下载。世代号的输出值从 begin_synchronization 脚本传递到 end_synchronization 脚本。generation_number 的意义取决于是否在使用当前同步来创建一个下载文件，或者当前同步是否具有上载。

在基于文件的下载中，世代号用于强制在下载之前进行上载。该编号存储在下载文件中。在包含上载的同步过程中，将为每个发布预订输出一个世代号。这些世代号在上载确认中被发送给远程数据库，并存储于 SYSSYNC.generation_number 中。

另请参见

- “脚本参数”一节第 299 页
- “添加和删除脚本”一节第 306 页
- “end_publication 连接事件”一节第 394 页
- “MobiLink 基于文件的下载”第 275 页
- “MobiLink 世代号”一节第 281 页
- “在脚本中使用远程 ID 和 MobiLink 用户名”一节 《MobiLink - 客户端管理》
- “在脚本中使用上次下载时间”一节第 122 页

SQL 示例

您可能要为每个正被同步的发布记录该信息。以下示例调用 ml_add_connection_script，将事件指派给名为 RecordPubSync 的存储过程。

```
CALL ml_add_connection_script(
 'version1',
 'begin_publication',
 '{CALL RecordPubSync(
    {ml s.generation_number},
    {ml s.username},
    {ml s.publication_name},
    {ml s.last_publication_upload},
    {ml s.last_publication_download},
    {ml s.subscription_id} )}' );
```

Java 示例

以下对 MobiLink 系统过程的调用在同步脚本版本 ver1 时将名为 beginPublication 的 Java 方法注册为 begin_publication 连接事件的脚本。

```
CALL ml_add_java_connection_script(
 'ver1',
 'begin_publication',
 'ExamplePackage.ExampleClass.beginPublication' )
```

以下是 Java 方法 beginPublication 示例。它保存每个发布的名称供以后使用。

```
public String beginPublication(
  ianywhere.ml.script.InOutInteger generation_number,
  String user,
  String pub_name,
  Timestamp last_publication_upload,
  Timestamp last_download ) {
  _publicationNames[ _numPublications++ ] = pub_name;
  return ( null );
}
```

.NET 示例

以下对 MobiLink 系统过程的调用在同步脚本版本 ver1 时将名为 BeginPub 的 .NET 方法注册为 begin_publication 连接事件的脚本。

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

```
CALL ml_add_dnet_connection_script(
 'ver1',
 'begin_publication',
 'TestScripts.Test.BeginPub'
)
```

以下是 .NET 方法 BeginPub 示例。它保存每个发布的名称供以后使用。

```
public string BeginPub(
  ref int generation_number,
  string user,
  string pub_name,
  DateTime last_publication_upload,
  DateTime last_download ) {
  _publicationNames[ _numPublications++ ] = pub_name;
  return ( null );
}
```

begin_synchronization 连接事件

在应用程序连接到 MobiLink 服务器以便进行同步过程准备时处理任何语句。

参数

在下表中，说明部分提供 SQL 数据类型。如果您使用 Java 或 .NET 编写脚本，则应该使用相应的数据类型。请参见“SQL-Java 数据类型”一节第 498 页和“SQL-.NET 数据类型”一节第 556 页。

SQL 脚本的参数名称	说明	顺序
s.remote_id	VARCHAR(128)。MobiLink 远程 ID。只有在使用命名参数时才能引用远程 ID。	不适用
s.username	VARCHAR(128)。MobiLink 用户名。	1

缺省操作

无。

注释

在准备进行同步的应用程序与 MobiLink 服务器建立连接后，MobiLink 服务器立即执行此事件。该事件在上载事务之前的一个单独事务中执行。

begin_synchronization 脚本对于维护统计信息很有用。这是因为即使存在错误或冲突，也会调用 end_synchronization 脚本，从而使回退上载事务时，统计信息之类的内容可得到维护。

另请参见

- “脚本参数”一节第 299 页
- “添加和删除脚本”一节第 306 页
- “end_synchronization 连接事件”一节第 397 页
- “begin_synchronization 表事件”一节第 360 页
- “在脚本中使用远程 ID 和 MobiLink 用户名”一节 《MobiLink - 客户端管理》

SQL 示例

如果要在后面的脚本中多次引用用户名值，则您最好将该值存储在临时表或变量中。

```
CALL ml_add_connection_script (
  'version1',
  'begin_synchronization',
  'set @EmployeeID = {ml s.username}' );
```

Java 示例

以下对 MobiLink 系统过程的调用在同步脚本版本 ver1 时将名为 beginSynchronizationConnection 的 Java 方法注册为 begin_synchronization 连接事件的脚本。

```
CALL ml_add_java_connection_script(
 'ver1',
```

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

```
  'begin_synchronization',
  'ExamplePackage.ExampleClass.beginSynchronizationConnection'
)
```

以下是 Java 方法 beginSynchronizationConnection 示例。它保存同步用户的名称供以后使用。

```
public String beginSynchronizationConnection(
  String user ) {
  _curUser = user;
  return ( null );
}
```

.NET 示例

以下对 MobiLink 系统过程的调用在同步脚本版本 ver1 时将名为 BeginSync 的 .NET 方法注册为 begin_synchronization 连接事件的脚本。

```
CALL ml_add_dnet_connection_script( 'ver1',
  'begin_synchronization',
  'TestScripts.Test.BeginSync'
)
```

以下是 .NET 方法 BeginSync 示例。它保存同步用户的名称供以后使用。

```
public string BeginSync(
  string user ) {
  _curUser = user;
  return ( null );
}
```

begin_synchronization 表事件

在应用程序连接到 MobiLink 服务器以便进行同步过程准备时，处理与特定的表有关的语句。

参数

在下表中，说明部分提供 SQL 数据类型。如果您使用 Java 或 .NET 编写脚本，则应该使用相应的数据类型。请参见“SQL-Java 数据类型”一节第 498 页和“SQL-.NET 数据类型”一节第 556 页。

在 SQL 脚本中，可以使用名称或问号指定事件参数，但不能在一个脚本中混合使用名称和问号。如果使用问号，则参数必须按照如下所示的顺序并且仅当没有指定任何后继参数时才是可选的（例如，如果您想使用参数 2，则必须使用参数 1）。如果使用命名参数，则可以按照任何顺序指定任意参数子集。

SQL 脚本的参数名称	说明	顺序
s.remote_id	VARCHAR(128)。MobiLink 远程 ID。只有在使用命名参数时才能引用远程 ID。	不适用
s.username	VARCHAR(128)。MobiLink 用户名。	1
s.table	VARCHAR(128)。表名。	2

缺省操作

无。

注释

MobiLink 服务器在准备同步的应用程序创建完与 MobiLink 服务器的连接以及 begin_synchronization 连接级别事件之后执行此事件。

您可以为远程数据库的每个表编写一个 begin_synchronization 脚本。只有在对表进行同步时才会调用相应的事件。

另请参见

- “脚本参数”一节第 299 页
- “添加和删除脚本”一节第 306 页
- “end_synchronization 表事件”一节第 399 页
- “begin_synchronization 连接事件”一节第 358 页
- “在脚本中使用远程 ID 和 MobiLink 用户名”一节 《MobiLink - 客户端管理》

SQL 示例

begin_synchronization 表事件用于设置特定表的同步。以下 SQL 脚本在同步过程中注册一个脚本，该脚本创建一个用于存储行的临时表。此语法用于 SQL Anywhere 统一数据库。

```
CALL ml_add_table_script(
 'ver1',
 'sales_order',
 'begin_synchronization',
 'CREATE TABLE #sales_order (
```

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

```
    id          integer NOT NULL default autoincrement,
    cust_id     integer NOT NULL,
    order_date      date NOT NULL,
    fin_code_id char(2) NULL,
    region      char(7) NULL,
    sales_rep       integer NOT NULL,
    PRIMARY KEY (id),
)' )
```

Java 示例

以下对 MobiLink 系统过程的调用在同步脚本版本 ver1 时将名为 beginSynchronizationTable 的 Java 方法注册为 begin_synchronization 表事件的脚本。

```
CALL ml_add_java_table_script(
 'ver1',
 'table1',
 'begin_synchronization',
 'ExamplePackage.ExampleClass.beginSynchronizationTable' )
```

以下是 Java 方法 beginSynchronizationTable 示例。它将当前表名添加到此实例中包含的表名列表中。

```
public String beginSynchronizationTable(
  String user,
  String table ) {
  _tableList.add( table );
  return ( null );
}
```

.NET 示例

以下对 MobiLink 系统过程的调用在同步脚本版本 ver1 和表 table1 时将名为 BeginTableSync 的 .NET 方法注册为 begin_synchronization 表事件的脚本。

```
CALL ml_add_dnet_table_script (
  'ver1',
  'table1',
  'begin_synchronization',
  'TestScripts.Test.BeginTableSync' )
```

以下是 .NET 方法 BeginTableSync 示例。它将当前表名添加到此实例中包含的表名列表中。

```
public string BeginTableSync(
  string user,
  string table ) {
  _tableList.Add( table );
  return ( null );
}
```

begin_upload 连接事件

紧接在 MobiLink 服务器开始处理上载的插入、更新和删除操作的流之前处理任何语句。

参数

在下表中，说明部分提供 SQL 数据类型。如果您使用 Java 或 .NET 编写脚本，则应该使用相应的数据类型。请参见“SQL-Java 数据类型”一节第 498 页和“SQL-.NET 数据类型”一节第 556 页。

SQL 脚本的参数名称	说明	顺序
s.remote_id	VARCHAR(128)。MobiLink 远程 ID。只有在使用命名参数时才能引用远程 ID。	不适用
s.username	VARCHAR(128)。MobiLink 用户名。	1

缺省操作

无。

注释

MobiLink 服务器执行此事件是对上载的信息进行处理的第一步。上载信息在单独的事务中进行处理。执行此事件是此事务中的第一步操作。

另请参见

- “脚本参数”一节第 299 页
- “添加和删除脚本”一节第 306 页
- “end_upload 连接事件”一节第 401 页
- “begin_upload 表事件”一节第 364 页
- “在脚本中使用远程 ID 和 MobiLink 用户名”一节 《MobiLink - 客户端管理》

SQL 示例

begin_upload 连接事件用于执行在上载行前所需执行的任何步骤。以下 SQL 脚本创建一个临时表，用于存储旧行值和新行值以处理 sales_order 表中的冲突。此示例在 SQL Anywhere 统一数据库中运行。

```
CALL ml_add_connection_script(
 'version1',
 'begin_upload',
 'CREATE TABLE #sales_order_conflicts (
    id          integer NOT NULL default autoincrement,
    cust_id     integer NOT NULL,
    order_date      date NOT NULL,
    fin_code_id char(2) NULL,
    region      char(7) NULL,
    sales_rep       integer NOT NULL,
    new_value       char(1) NOT NULL,
    PRIMARY KEY (id) )' )
```

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

Java 示例

以下对 MobiLink 系统过程的调用在同步脚本版本 ver1 时将名为 beginUploadConnection 的 Java 方法注册为 begin_upload 连接事件的脚本。

```
CALL ml_add_java_connection_script(
 'ver1',
 'begin_upload',
 'ExamplePackage.ExampleClass.beginUploadConnection ' )
```

以下是 Java 方法 beginUploadConnection 示例。它将消息输出到 MobiLink 消息日志。（请注意：将消息输出到 MobiLink 消息日志在开发时可能会有帮助，但会降低生产服务器的性能。）

```
public String beginUploadConnection( String user ) {
  java.lang.System.out.println(
    "Starting upload for user: " + user );
  return ( null );
}
```

.NET 示例

以下对 MobiLink 系统过程的调用在同步脚本版本 ver1 时将名为 BeginUpload 的 .NET 方法注册为 begin_upload 连接事件的脚本。

```
CALL ml_add_dnet_connection_script(
  'ver1',
  'begin_upload',
  'TestScripts.Test.BeginUpload'
)
```

以下 C# 示例保存当前用户名供以后的事件使用。

```
public string BeginUpload( string curUser ) {
  user = curUser;
  return ( null );
}
```

begin_upload 表事件

紧接在 MobiLink 服务器开始处理上载的插入、更新和删除操作的流之前，处理与特定的表有关的语句。

参数

在下表中，说明部分提供 SQL 数据类型。如果您使用 Java 或 .NET 编写脚本，则应该使用相应的数据类型。请参见“SQL-Java 数据类型”一节第 498 页和“SQL-.NET 数据类型”一节第 556 页。

在 SQL 脚本中，可以使用名称或问号指定事件参数，但不能在一个脚本中混合使用名称和问号。如果使用问号，则参数必须按照如下所示的顺序并且仅当没有指定任何后继参数时才是可选的（例如，如果您想使用参数 2，则必须使用参数 1）。如果使用命名参数，则可以按照任何顺序指定任意参数子集。

SQL 脚本的参数名称	说明	顺序
s.remote_id	VARCHAR(128)。MobiLink 远程 ID。只有在使用命名参数时才能引用远程 ID。	不适用
s.username	VARCHAR(128)。MobiLink 用户名。	1
s.table	VARCHAR(128)。表名。	2

缺省操作

无。

注释

MobiLink 服务器执行此事件是对上载的信息进行处理的第一步。上载信息在单独的事务中进行处理。执行此事件是此事务中的第一步表特定的操作。

您可以为远程数据库的每个表编写一个 begin_upload 脚本。只有在对表实际进行同步时才会调用相应的脚本。

另请参见

- “脚本参数”一节第 299 页
- “添加和删除脚本”一节第 306 页
- “end_upload 表事件”一节第 403 页
- “begin_upload 连接事件”一节第 362 页
- “在脚本中使用远程 ID 和 MobiLink 用户名”一节 《MobiLink - 客户端管理》

SQL 示例

在从远程数据库上载行时，您可能希望将所做的更改放入一个中间表中并亲自手工处理这些更改。您可以在此事件中填入一个全局临时表。

```
CALL ml_add_table_script(
 'version1',
```

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

```
 'Leads',
 'begin_upload',
 'INSERT INTO T_Leads
  SELECT * FROM Leads
  WHERE Owner = @EmployeeID' )
```

Java 示例

以下对 MobiLink 系统过程的调用在同步脚本版本 ver1 时将名为 beginUploadTable 的 Java 方法注册为 begin_upload 表事件的脚本。

```
CALL ml_add_java_table_script(
  'ver1',
  'table1',
  'begin_upload',
  'ExamplePackage.ExampleClass.beginUploadTable'
 )
```

以下是 Java 方法 beginUploadTable 示例。它将消息输出到 MobiLink 消息日志。（请注意：将消息输出到 MobiLink 消息日志在开发时可能会有帮助，但会降低生产服务器的性能。）

```
public String beginUploadTable(
  String user,
  String table ) {
  java.lang.System.out.println("Beginning to process upload for: " + table);
  return ( null );
}
```

.NET 示例

以下对 MobiLink 系统过程的调用在同步脚本版本 ver1 和表 table1 时将名为 BeginTableUpload 的 .NET 方法注册为 begin_upload 表事件的脚本。

```
CALL ml_add_dnet_table_script(
  'ver1',
  'table1',
  'begin_upload',
  'TestScripts.Test.BeginTableUpload'
)
```

以下是 .NET 方法 BeginTableUpload 示例。它将消息输出到 MobiLink 消息日志。（请注意：将消息输出到 MobiLink 消息日志在开发时可能会有帮助，但会降低生产服务器的性能。）

```
public string BeginTableUpload(
  string user,
  string table ) {
  System.Console.WriteLine("Beginning to process upload for: " + table);
  return ( null );
}
```

begin_upload_deletes 表事件

紧接在上载已从远程数据库的特定表中删除的行之前，处理与该特定表有关的语句。

参数

在下表中，说明部分提供 SQL 数据类型。如果您使用 Java 或 .NET 编写脚本，则应该使用相应的数据类型。请参见“SQL-Java 数据类型”一节第 498 页和“SQL-.NET 数据类型”一节第 556 页。

在 SQL 脚本中，可以使用名称或问号指定事件参数，但不能在一个脚本中混合使用名称和问号。如果使用问号，则参数必须按照如下所示的顺序并且仅当没有指定任何后继参数时才是可选的（例如，如果您想使用参数 2，则必须使用参数 1）。如果使用命名参数，则可以按照任何顺序指定任意参数子集。

SQL 脚本的参数名称	说明	顺序
s.remote_id	VARCHAR(128)。MobiLink 远程 ID。只有在使用命名参数时才能引用远程 ID。	不适用
s.username	VARCHAR(128)。MobiLink 用户名。	1
s.table	VARCHAR(128)。表名。	2

缺省操作

无。

注释

此事件紧接在应用客户端表（由第二个参数指定）中的行删除操作所带来的更改之前发生。

您可以为远程数据库的每个表编写一个 begin_upload_deletes 脚本。只有在对表实际进行同步时才会调用相应的脚本。

另请参见

- “脚本参数”一节第 299 页
- “添加和删除脚本”一节第 306 页
- “end_upload_deletes 表事件”一节第 406 页
- “在脚本中使用远程 ID 和 MobiLink 用户名”一节 《MobiLink - 客户端管理》

SQL 示例

begin_upload_deletes 连接事件用于执行在为特定表上载插入和更新操作之后、上载删除操作之前所需执行的任何步骤。以下 SQL 脚本创建一个临时表，用于在上载过程中临时存储删除操作。此语法用于 SQL Anywhere 统一数据库。

```
CALL ml_add_table_script(
 'ver1',
 'sales_order',
 'begin_upload_deletes',
 'CREATE TABLE #sales_order_deletes (
```

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

```
    id          integer NOT NULL default autoincrement,
    cust_id     integer NOT NULL,
    order_date  date NOT NULL,
    fin_code_id char(2) NULL,
    region      char(7) NULL,
    sales_rep   integer NOT NULL,
    PRIMARY KEY (id) )' )
```

Java 示例

以下对 MobiLink 系统过程的调用在同步脚本版本 ver1 时将名为 beginUploadDeletes 的 Java 方法注册为 begin_upload_deletes 表事件的脚本。

```
CALL ml_add_java_table_script(
  'ver1',
  'table1',
  'begin_upload_deletes',
  'ExamplePackage.ExampleClass.beginUploadDeletes' )
```

以下是 Java 方法 beginUploadDeletes 示例。它将消息输出到 MobiLink 消息日志。（请注意：将消息输出到 MobiLink 消息日志在开发时可能会有帮助，但会降低生产服务器的性能。）

```
public String beginUploadDeletes(
  String user,
  String table )
  throws java.sql.SQLException {
  java.lang.System.out.println( "Starting upload
    deletes for table: " + table );
  return ( null );
}
```

.NET 示例

以下对 MobiLink 系统过程的调用在同步脚本版本 ver1 和表 table1 时将名为 BeginUploadDeletes 的 .NET 方法注册为 begin_upload_deletes 表事件的脚本。

```
CALL ml_add_dnet_table_script(
  'ver1',
  'table1',
  'begin_upload_deletes',
  'TestScripts.Test.BeginUploadDeletes'
)
```

以下是 .NET 方法 BeginUploadDeletes 示例。它将消息输出到 MobiLink 消息日志。（请注意：将消息输出到 MobiLink 消息日志在开发时可能会有帮助，但会降低生产服务器的性能。）

```
public string BeginUploadDeletes(
  string user,
  string table ) {
  System.Console.WriteLine(
   "Starting upload deletes for table: " + table );
  return ( null );
}
```

begin_upload_rows 表事件

紧接在上载远程数据库的特定表中插入和更新的行之前，处理与该特定表有关的语句。

参数

在下表中，说明部分提供 SQL 数据类型。如果您使用 Java 或 .NET 编写脚本，则应该使用相应的数据类型。请参见“SQL-Java 数据类型”一节第 498 页和“SQL-.NET 数据类型”一节第 556 页。

在 SQL 脚本中，可以使用名称或问号指定事件参数，但不能在一个脚本中混合使用名称和问号。如果使用问号，则参数必须按照如下所示的顺序并且仅当没有指定任何后继参数时才是可选的（例如，如果您想使用参数 2，则必须使用参数 1）。如果使用命名参数，则可以按照任何顺序指定任意参数子集。

SQL 脚本的参数名称	说明	顺序
s.remote_id	VARCHAR(128)。MobiLink 远程 ID。只有在使用命名参数时才能引用远程 ID。	不适用
s.username	VARCHAR(128)。MobiLink 用户名。	1
s.table	VARCHAR(128)。表名。	2

缺省操作

无。

注释

此事件紧接在应用客户端表（由第二个参数指定）中的插入和删除操作所带来的更改之前发生。

您可以为远程数据库的每个表编写一个 begin_upload_rows 脚本。只有在对表实际进行同步时才会调用相应的脚本。

另请参见

- “脚本参数”一节第 299 页
- “添加和删除脚本”一节第 306 页
- “end_upload_rows 表事件”一节第 408 页
- “在脚本中使用远程 ID 和 MobiLink 用户名”一节 《MobiLink - 客户端管理》

SQL 示例

begin_upload_rows 连接事件用于执行在为特定表上载插入操作和更新操作之前所需执行的任何步骤。下面的脚本调用一个存储过程，该过程为对 Inventory 表的插入和更新操作准备统一数据库：

```
CALL ml_add_table_script(
 'MyCorp 1.0',
 'Inventory',
 'begin_upload_rows',
 'CALL PrepareForUpserts()' )
```

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

Java 示例

以下对 MobiLink 系统过程的调用在同步脚本版本 ver1 时将名为 beginUploadRows 的 Java 方法注册为 begin_upload_rows 表事件的脚本。

```
CALL ml_add_java_table_script(
  'ver1',
  'table1',
  'begin_upload_rows',
  'ExamplePackage.ExampleClass.beginUploadRows' )
```

以下是 Java 方法 beginUploadRows 示例。它将消息输出到 MobiLink 消息日志。（请注意：将消息输出到 MobiLink 消息日志在开发时可能会有帮助，但会降低生产服务器的性能。）

```
public String beginUploadRows(
  String user,
  String table )
  throws java.sql.SQLException {
  java.lang.System.out.println(
   "Starting upload rows for table: " +
   table + " and user: " + user );
  return ( null );
}
```

.NET 示例

以下对 MobiLink 系统过程的调用在同步脚本版本 ver1 和表 table1 时将名为 BeginUploadRows 的 .NET 方法注册为 begin_upload_rows 表事件的脚本。

```
CALL ml_add_dnet_table_script(
  'ver1',
  'table1',
  'begin_upload_rows',
  'TestScripts.Test.BeginUploadRows'
)
```

以下是 .NET 方法 BeginUploadRows 示例。它将消息输出到 MobiLink 消息日志。（请注意：将消息输出到 MobiLink 消息日志在开发时可能会有帮助，但会降低生产服务器的性能。）

```
public string BeginUploadRows(
  string user,
  string table ) {
  System.Console.WriteLine(
   "Starting upload rows for table: " +
   table + " and user: " + user );
  return ( null );
}
```

download_cursor 表事件

定义游标以选择远程数据库中要下载并插入或更新的行。

参数

在下表中，说明部分提供 SQL 数据类型。如果您使用 Java 或 .NET 编写脚本，则应该使用相应的数据类型。请参见“SQL-Java 数据类型”一节第 498 页和“SQL-.NET 数据类型”一节第 556 页。

在 SQL 脚本中，可以使用名称或问号指定事件参数，但不能在一个脚本中混合使用名称和问号。如果使用问号，则参数必须按照如下所示的顺序并且仅当没有指定任何后继参数时才是可选的（例如，如果您想使用参数 2，则必须使用参数 1）。如果使用命名参数，则可以按照任何顺序指定任意参数子集。

SQL 脚本的参数名称	说明	顺序
s.last_table_download	TIMESTAMP。表的上一次下载时间。	1
s.remote_id	VARCHAR(128)。MobiLink 远程 ID。只有在使用命名参数时才能引用远程 ID。	不适用
s.username	VARCHAR(128)。MobiLink 用户名。	2

缺省操作

无。

注释

MobiLink 服务器打开一个只读游标，并使用此游标读取要下载到远程数据库中的行列表。此脚本应该包含适当的 SELECT 语句。

您可以为远程数据库的每个表编写一个 download_cursor 脚本。

为了优化 UltraLite 客户端同步过程中下载阶段的性能，当主键值的范围超出了设备上的当前行时，您应按照主键对下载游标中的行进行排序。例如，下载大型参考表时将从这种优化中受益。

每个 download_cursor 脚本必须包含一个 SELECT 语句或对含有 SELECT 语句的过程的调用。MobiLink 服务器将使用此语句在统一数据库中定义游标。

该脚本必须选择与远程数据库中相应的表中的列对应的所有列。统一数据库中的列可以与远程数据库中相应的列具有不同的名称，但是列的类型必须相互兼容。

必须按照远程数据库中对应列的定义顺序来选择列。

注意，download_cursor 允许级联删除。因此，您可以从数据库中删除记录。

为了避免下载不必要的行，您应该在 download_cursor 脚本的 WHERE 子句中包括以下行：

```
AND last_table_download > '1900/1/1'
```

对于 Java 和 .NET 应用程序，此脚本必须返回有效的 SQL。

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

如果您因为正在进行影响下载性能的大量更新而考虑在 download_cursor 脚本中使用 READPAST 表提示，请考虑为下载改用快照隔离。READPAST 表提示如果在 download_cursor 脚本中使用可能会引起问题。使用基于时间戳的下载时，READPAST 提示会造成行遗漏丢失，还可能造成某行永远不会下载到远程数据库。例如：

- 将某行添加到统一数据库并提交。该行有一个时间为昨天的 last_modified 列。
- 同一行被更新但不提交。
- last_download 时间为上周的远程数据库进行同步。
- 如果 download_cursor 脚本试图使用 READPAST 选择行，则会跳过该行。
- 更新该行的事务回退。远程数据库的下一个上次下载时间提前为今天。

从此时开始，除非进行更新，否则该行永远不会被下载。可能的解决方案是实现 modify_next_last_download_timestamp 脚本并将上次下载时间设置为最早打开的事务的开始时间。

另请参见

- “脚本参数”一节第 299 页
- “添加和删除脚本”一节第 306 页
- “编写用于下载行的脚本”一节第 312 页
- “编写 download_cursor 脚本”一节第 313 页
- “在远程数据库之间对行进行分区”一节第 127 页
- “download_delete_cursor 表事件”一节第 373 页
- “在脚本中使用远程 ID 和 MobiLink 用户名”一节 《MobiLink - 客户端管理》
- “在脚本中使用上次下载时间”一节第 122 页
- 在 “FROM 子句”一节 《SQL Anywhere 服务器 - SQL 参考》 中 "对 MobiLink 同步使用 READPAST"

SQL 示例

以下示例来自于 Oracle 安装，但语句对于所有支持的数据库都有效。该示例将下载自从用户上次下载数据以来更改过且与 emp_name 列中的用户名相匹配的所有行。

```
CALL ml_add_table_script(
  'Lab',
  'ULOrder',
  'download_cursor',
  'SELECT order_id,
    cust_id,
    prod_id,
    emp_id,
    disc,
    quant,
    notes,
    status
   FROM ULOrder
   WHERE last_modified >= {ml s.last_table_download}
    AND emp_name = {ml s.username}' )
```

Java 示例

以下对 MobiLink 系统过程的调用在同步脚本版本 ver1 时将名为 downloadCursor 的 Java 方法注册为 download_cursor 表事件的脚本。

```
CALL ml_add_java_table_script(
   'ver1',
   'ULCustomer',
   'download_cursor',
   'ExamplePackage.ExampleClass.downloadCursor ' )
```

以下是 Java 方法 downloadCursor 示例。它返回一条 SQL 语句，用于下载 last_modified 列大于或等于上次下载时间的行。

```
public String downloadCursor(
  java.sql.Timestamp ts,
  String user )  {
  return( "SELECT cust_id, cust_name FROM ULCustomer
             WHERE last_modified >= ' "
   + ts + " ' ");
}
```

.NET 示例

以下对 MobiLink 系统过程的调用在同步脚本版本 ver1 和表 table1 时将名为 DownloadCursor 的 .NET 方法注册为 download_cursor 表事件的脚本。

```
CALL ml_add_dnet_table_script(
 'ver1',
 'table1',
 'download_cursor',
 'TestScripts.Test.DownloadCursor'
)
```

以下是 .NET 方法 DownloadCursor 示例。它用名为 *rows.txt* 的文件的内容填充临时表。然后它返回一个游标，该游标使 MobiLink 将临时表中的行发送到远程数据库。此语法对于 SQL Anywhere 统一数据库有效。

```
public string DownloadCursor(
  DateTime ts,
  string user ) {
  DBCommand stmt   = curConn.CreateCommand();
  StreamReader input  = new StreamReader( "rows.txt" );
  string sql    = input.ReadLine();
  stmt.CommandText = "DELETE FROM dnet_dl_temp";
  stmt.ExecuteNonQuery();
  while( sql != null ){
    stmt.CommandText = "INSERT INTO dnet_dl_temp VALUES " + sql;
    stmt.ExecuteNonQuery();
    sql = input.ReadLine();
  }
  return( "SELECT * FROM dnet_dl_temp" );
}
```

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

download_delete_cursor 表事件

定义游标以选择要从远程数据库中删除的行。

参数

在下表中，说明部分提供 SQL 数据类型。如果您使用 Java 或 .NET 编写脚本，则应该使用相应的数据类型。请参见“SQL-Java 数据类型”一节第 498 页和“SQL-.NET 数据类型”一节第 556 页。

在 SQL 脚本中，可以使用名称或问号指定事件参数，但不能在一个脚本中混合使用名称和问号。如果使用问号，则参数必须按照如下所示的顺序并且仅当没有指定任何后继参数时才是可选的（例如，如果您想使用参数 2，则必须使用参数 1）。如果使用命名参数，则可以按照任何顺序指定任意参数子集。

SQL 脚本的参数名称	说明	顺序
s.last_table_download	TIMESTAMP。表的上一次下载时间。	1
s.remote_id	VARCHAR(128)。MobiLink 远程 ID。只有在使用命名参数时才能引用远程 ID。	不适用
s.username	VARCHAR(128)。MobiLink 用户名。	2

缺省操作

无。

注释

MobiLink 服务器打开一个只读游标，并使用此游标读取将要在远程数据库中下载然后插入或更新的行的列表。此脚本必须包含一个 SELECT 语句，该语句必须能够返回将要从远程数据库的表中删除的行的主键值。

您可以为远程数据库的每个表编写一个 download_delete_cursor 脚本。

如果在 download_delete_cursor 中某个表的一个或多个行的主键列为空，则 MobiLink 将通知远程数据库删除该表中的所有数据。请参见“删除表中的所有行”一节第 314 页。

注意，统一数据库中删除的行不出现在 download_delete_cursor 事件定义的结果集中，因此不会从远程数据库中自动删除。一个用于标识将要从远程数据库中删除的行的方法是向统一数据库表添加一列，用以将行标识为非活动。

为了避免下载不必要的行，您应该在 download_delete_cursor 脚本的 WHERE 子句中包括以下行：

```
AND last_modified > '1900/1/1'
```

对于 Java 和 .NET 应用程序，此脚本必须返回有效的 SQL。

在 download_delete_cursor 中使用 READPAST 表提示可能会有问题。有关详细信息，请参见 download_cursor 事件。

Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

另请参见

- “脚本参数”一节第 299 页
- “添加和删除脚本”一节第 306 页
- “download_cursor 表事件”一节第 370 页
- “编写用于下载行的脚本”一节第 312 页
- “在远程数据库之间对行进行分区”一节第 127 页
- “编写 download_delete_cursor 脚本”一节第 314 页
- “在脚本中使用远程 ID 和 MobiLink 用户名”一节 《MobiLink - 客户端管理》
- “在脚本中使用上次下载时间”一节第 122 页
- 在 “FROM 子句”一节 《SQL Anywhere 服务器 - SQL 参考》 中 "对 MobiLink 同步使用 READPAST"

SQL 示例

此示例来自 Contact 示例，并可在 *Samples\MobiLink\Contact\build_consol.sql* 中找到。它从远程数据库中删除满足下列条件的所有客户：自从上次该用户下载数据后已更改 (`Customer.last_modified >= {ml s.last_table_download}`)，并且

- 不属于正在同步的用户 (`SalesRep.username != {ml s.username}`)，或者
- 在统一数据库中被标记为非活动状态 (`Customer.active = 0`)。

```
CALL ml_add_table_script(
 'ver1',
 'table1',
 'download_delete_cursor',
 'SELECT cust_id FROM Customer key join SalesRep
  WHERE Customer.last_modified >= {ml s.last_table_download} AND
   ( SalesRep.username != {ml s.username} OR Customer.active = 0 )')
```

Java 示例

以下对 MobiLink 系统过程的调用在同步脚本版本 ver1 时将名为 downloadDeleteCursor 的 Java 方法注册为 download_delete_cursor 事件的脚本。

```
CALL ml_add_java_table_script(
   'ver1',
   'table1',
   'download_delete_cursor',
   'ExamplePackage.ExampleClass.downloadDeleteCursor' )
```

以下是 Java 方法 downloadDeleteCursor 示例。它调用的 Java 方法可以生成下载删除游标的 SQL。

```
public String downloadDeleteCursor(
  Timestamp ts,
  String user ) {
  return( getDownloadCursor( _curUser, _curTable ) );
}
```

.NET 示例

以下对 MobiLink 系统过程的调用在同步脚本版本 ver1 和表 table1 时将名为 DownloadDeleteCursor 的 .NET 方法注册为 download_delete_cursor 表事件的脚本。

```
CALL ml_add_dnet_table_script(
  'ver1',
```

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

```
  'table1',
  'download_delete_cursor',
  'TestScripts.Test.DownloadDeleteCursor'
)
```

以下是 .NET 方法 DownloadDeleteCursor 示例。它调用的 .NET 方法可以生成下载删除游标的 SQL。

```
public string DownloadDeleteCursor(
  DateTime timestamp,
  string user ) {
  return( GetDownloadCursor( _curUser, _curTable ) );
}
```

download_statistics 连接事件

跟踪有关下载操作的同步统计信息。

参数

在下表中，说明部分提供 SQL 数据类型。如果您使用 Java 或 .NET 编写脚本，则应该使用相应的数据类型。请参见“SQL-Java 数据类型”一节第 498 页和“SQL-.NET 数据类型”一节第 556 页。

在 SQL 脚本中，可以使用名称或问号指定事件参数，但不能在一个脚本中混合使用名称和问号。如果使用问号，则参数必须按照如下所示的顺序并且仅当没有指定任何后继参数时才是可选的（例如，如果您想使用参数 2，则必须使用参数 1）。如果使用命名参数，则可以按照任何顺序指定任意参数子集。

SQL 脚本的参数名称	说明	顺序
s.remote_id	VARCHAR(128)。MobiLink 远程 ID。只有在使用命名参数时才能引用远程 ID。	不适用
s.username	VARCHAR(128)。在 SYNCHRONIZATION USER 定义中指定的 MobiLink 用户名。	1
s.warnings	INTEGER。发出的警告的数目。	2
s.errors	INTEGER。发生的错误的数目，包括已处理的错误。	3
s.fetched_rows	INTEGER。download_cursor 脚本读取的行数。	4
s.deleted_rows	INTEGER。download_delete_cursor 脚本读取的行数。	5
s.filtered_rows	INTEGER。从 deleted_rows 参数实际发送到远程数据库的行数。它反映了已上载的值的下载过滤。	6
s.bytes	INTEGER。作为下载发送到远程数据库的字节数。	7

缺省操作

无。

注释

download_statistics 事件可用于为所有用户收集关于下载的统计信息。download_statistics 连接脚本紧接在下载事务结束时执行提交操作之前被调用。

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

注意
并不是所有警告和错误都被记录下来，是否记录取决于命令行，因此实际警告和错误的数目将大于被记录下来的警告和错误的数目。

另请参见

- “脚本参数”一节第 299 页
- “添加和删除脚本”一节第 306 页
- “download_statistics 表事件”一节第 379 页
- “upload_statistics 连接事件”一节第 479 页
- “upload_statistics 表事件”一节第 483 页
- “synchronization_statistics 连接事件”一节第 453 页
- “synchronization_statistics 表事件”一节第 456 页
- “time_statistics 连接事件”一节第 459 页
- “time_statistics 表事件”一节第 462 页
- “MobiLink 监控器”第 167 页
- “在脚本中使用远程 ID 和 MobiLink 用户名”一节 《MobiLink - 客户端管理》

SQL 示例

以下示例将同步统计插入一个名为 download_audit 的表中。

```
CALL ml_add_connection_script(
 'ver1',
 'download_statistics',
 'INSERT INTO download_audit(
   user_name,
   warnings,
   errors,
   deleted_rows,
   fetched_rows,
   download_rows,
   bytes )
  VALUES (
   {ml s.username},
   {ml s.warnings},
   {ml s.errors},
   {ml s.fetched_rows},
   {ml s.deleted_rows},
   {ml s.filtered_rows},
   {ml s.bytes})')
```

在审计表中插入重要的统计信息之后，您将可以使用这些统计信息监控同步过程并在条件允许时进行优化。

Java 示例

以下对 MobiLink 系统过程的调用在同步脚本版本 ver1 时将名为 downloadStatisticsConnection 的 Java 方法注册为 download_statistics 事件的脚本。

```
CALL ml_add_java_connection_script(
  'ver1',
  'download_statistics',
  'ExamplePackage.ExampleClass.downloadStatisticsConnection' )
```

以下是 Java 方法 downloadStatisticsConnection 示例。它将读取的行数输出到 MobiLink 消息日志。（请注意：将读取的行数输出到 MobiLink 消息日志在开发时可能会有帮助，但会降低生产服务器的性能。）

```
public String downloadStatisticsConnection(
  String user,
  int warnings,
  int errors,
  int fetchedRows,
  int deletedRows,
  int bytes ) {
  java.lang.System.out.println(
   "download connection stats fetchedRows: "
   + fetchedRows );
  return ( null );
}
```

.NET 示例

以下对 MobiLink 系统过程的调用在同步脚本版本 ver1 时将名为 DownloadStats 的 .NET 方法注册为 download_statistics 连接事件的脚本。

```
CALL ml_add_dnet_connection_script(
  'ver1',
  'download_statistics',
  'TestScripts.Test.DownloadStats'
)
```

以下是 .NET 方法 DownloadStats 示例。它将读取的行数输出到 MobiLink 消息日志。（请注意：将读取的行数输出到 MobiLink 消息日志在开发时可能会有帮助，但会降低生产服务器的性能。）

```
public string DownloadStats(
  string user,
  int warnings,
  int errors,
  int deletedRows,
  int fetchedRows,
  int downloadRows,
  int bytes ) {
  System.Console.WriteLine(
   "download connection stats fetchedRows: "
   + fetchedRows );
  return ( null );
}
```

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

download_statistics 表事件

按照表跟踪有关下载操作的同步统计信息。

参数

在下表中，说明部分提供 SQL 数据类型。如果您使用 Java 或 .NET 编写脚本，则应该使用相应的数据类型。请参见“SQL-Java 数据类型”一节第 498 页和“SQL-.NET 数据类型”一节第 556 页。

在 SQL 脚本中，可以使用名称或问号指定事件参数，但不能在一个脚本中混合使用名称和问号。如果使用问号，则参数必须按照如下所示的顺序并且仅当没有指定任何后继参数时才是可选的（例如，如果您想使用参数 2，则必须使用参数 1）。如果使用命名参数，则可以按照任何顺序指定任意参数子集。

SQL 脚本的参数名称	说明	顺序
s.remote_id	VARCHAR(128)。MobiLink 远程 ID。只有在使用命名参数时才能引用远程 ID。	不适用
s.username	VARCHAR(128)。在 SYNCHRONIZATION USER 定义中指定的 MobiLink 用户名。	1
s.table	VARCHAR(128)。表名。	2
s.warnings	INTEGER。发出的警告的数目。	3
s.errors	INTEGER。发生的错误的数目，包括已处理的错误。	4
s.fetched_rows	INTEGER。download_cursor 脚本读取的行数。	5
s.deleted_rows	INTEGER。download_delete_cursor 脚本读取的行数。	6
s.filtered_rows	INTEGER。从 (6) 实际发送到远程数据库的行数。它反映了已上载的值的下载过滤。	7
s.bytes	INTEGER。作为下载发送到远程数据库的字节数。	8

缺省操作

无。

注释

download_statistics 事件可用于为所有用户和表收集关于下载（当它们应用于该表时）的统计信息。download_statistics 表脚本在下载事务结束时执行提交操作之前被调用。

Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

另请参见

- “脚本参数”一节第 299 页
- “添加和删除脚本”一节第 306 页
- “download_statistics 连接事件”一节第 376 页
- “upload_statistics 连接事件”一节第 479 页
- “upload_statistics 表事件”一节第 483 页
- “synchronization_statistics 连接事件”一节第 453 页
- “synchronization_statistics 表事件”一节第 456 页
- “time_statistics 连接事件”一节第 459 页
- “time_statistics 表事件”一节第 462 页
- “MobiLink 监控器”第 167 页
- “在脚本中使用远程 ID 和 MobiLink 用户名”一节 《MobiLink - 客户端管理》

SQL 示例

以下示例将同步统计插入一个名为 download_audit 的表中。在审计表中插入重要的统计信息之后，您将可以使用这些统计信息监控同步过程并在条件允许时进行优化。

```
CALL ml_add_table_script(
 'ver1',
 'table1',
 'download_statistics',
 'INSERT INTO download_audit (
   user_name,
   table, warnings,
   errors,
   deleted_rows,
   fetched_rows,
   download_rows,
   bytes)
  VALUES (
   {ml s.username},
   {ml s.table},
   {ml s.warnings},
   {ml s.errors},
   {ml s.fetched_rows},
   {ml s.deleted_rows},
   {ml s.filtered_rows},
   {ml s.bytes})')
```

Java 示例

以下对 MobiLink 系统过程的调用在同步脚本版本 ver1 时将名为 downloadStatisticsTable 的 Java 方法注册为 download_statistics 表事件的脚本。

```
CALL ml_add_java_table_script(
  'ver1',
  'table1',
  'download_statistics',
  'ExamplePackage.ExampleClass.downloadStatisticsTable' )
```

以下是 Java 方法 downloadStatisticsTable 示例。它将此表的某些统计信息输出到 MobiLink 消息日志。（请注意：将表的统计信息输出到 MobiLink 消息日志在开发时可能会有帮助，但会降低生产服务器的性能。）

```
public String downloadStatisticsTable(
  String user,
```

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

```
  String table,
  int warnings,
  int errors,
  int fetchedRows,
  int deletedRows,
  int bytes ) {
  java.lang.System.out.println( "download table stats "
   + "table: " + table + "bytes: " + bytes );
  return ( null );
}
```

.NET 示例

以下对 MobiLink 系统过程的调用在同步脚本版本 ver1 和表 table1 时将名为 DownloadTableStats 的 .NET 方法注册为 download_statistics 表事件的脚本。

```
CALL ml_add_dnet_table_script(
 'ver1',
 'table1',
 'download_statistics',
 'TestScripts.Test.DownloadTableStats'
)
```

以下是 .NET 方法 DownloadTableStats 示例。它将此表的某些统计信息输出到 MobiLink 消息日志。（请注意：将表的统计信息输出到 MobiLink 消息日志在开发时可能会有帮助，但会降低生产服务器的性能。）

```
public string DownloadTableStats(
  string user,
  string table,
  int warnings,
  int errors,
  int deletedRows,
  int fetchedRows,
  int downloadRows,
  int bytes ) {
  System.Console.WriteLine( "download table stats "
   + "table: " + table + "bytes: " + bytes );
  return ( null );
}
```

end_connection 连接事件

紧接在 MobiLink 服务器关闭与统一数据库服务器的连接之前，在准备关机或从连接池中删除连接时处理任何语句。

参数

无。

缺省操作

无。

注释

您可以使用 end_connection 脚本紧接在 MobiLink 服务器与统一数据库服务器之间的连接关闭之前执行所选的操作。

该脚本通常用于完成 begin_connection 脚本所启动的所有操作并释放其占用的所有资源。

另请参见

- “begin_connection 连接事件”一节第 343 页
- “添加和删除脚本”一节第 306 页

SQL 示例

以下 SQL 脚本删除由 begin_connection 脚本创建的临时表。此语法用于 SQL Anywhere 统一数据库。严格地讲，并不需要显式删除此表，因为 SQL Anywhere 在连接被取消时自动执行此操作。是否需要显式删除临时表取决于统一数据库的类型。

```
CALL ml_add_connection_script(
 'version 1.0',
 'end_connection',
 'DROP TABLE #sync_info' )
```

Java 示例

以下对 MobiLink 系统过程的调用在同步脚本版本 ver1 时将名为 endConnection 的 Java 方法注册为 end_connection 事件的脚本。

```
CALL ml_add_java_connection_script(
   'ver1',
   'end_connection',
   'ExamplePackage.ExampleClass.endConnection' )
```

以下是 Java 方法 endConnection 示例。它将消息输出到 MobiLink 消息日志。（请注意：将消息输出到 MobiLink 消息日志在开发时可能会有帮助，但会降低生产服务器的性能。）

```
public String endConnection() {
  java.lang.System.out.println( "Ending connection." );
  return ( null );
}
```

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

.NET 示例

以下对 MobiLink 系统过程的调用在同步脚本版本 ver1 时将名为 EndConnection 的 .NET 方法注册为 end_connection 连接事件的脚本。

```
CALL ml_add_dnet_connection_script(
  'ver1',
  'end_connection',
  'TestScripts.Test.EndConnection'
)
```

以下是 .NET 方法 EndConnection 示例。它将消息输出到 MobiLink 消息日志。（请注意：将消息输出到 MobiLink 消息日志在开发时可能会有帮助，但会降低生产服务器的性能。）

```
public string EndConnection() {
  System.Console.WriteLine( "Ending connection." );
  return ( null );
}
```

end_download 连接事件

在 MobiLink 服务器完成下载数据的准备工作之后立即处理任何语句。

参数

在下表中，说明部分提供 SQL 数据类型。如果您使用 Java 或 .NET 编写脚本，则应该使用相应的数据类型。请参见“SQL-Java 数据类型”一节第 498 页和“SQL-.NET 数据类型”一节第 556 页。

在 SQL 脚本中，可以使用名称或问号指定事件参数，但不能在一个脚本中混合使用名称和问号。如果使用问号，则参数必须按照如下所示的顺序并且仅当没有指定任何后继参数时才是可选的（例如，如果您想使用参数 2，则必须使用参数 1）。如果使用命名参数，则可以按照任何顺序指定任意参数子集。

SQL 脚本的参数名称	说明	顺序
s.last_download	TIMESTAMP。任何已同步表的上次下载时间。	1
s.remote_id	VARCHAR(128)。MobiLink 远程 ID。只有在使用命名参数时才能引用远程 ID。	不适用
s.username	VARCHAR(128)。MobiLink 用户名。	2

缺省操作

无。

注释

在完成所有行的下载后，MobiLink 服务器将执行此脚本。如果使用阻塞下载确认，则在收到接收确认之后执行此脚本。下载信息是在单独的一个事务中进行处理的。执行此脚本是此事务中的最后一步非统计操作。

另请参见

- “脚本参数”一节第 299 页
- “添加和删除脚本”一节第 306 页
- “begin_download 连接事件”一节第 345 页
- “在脚本中使用远程 ID 和 MobiLink 用户名”一节 《MobiLink - 客户端管理》
- “在脚本中使用上次下载时间”一节第 122 页

SQL 示例

以下示例说明了 end_download 连接脚本的一种可能的用法。ULEmpCust 表有一个操作列。以下脚本使用此列中的值删除远程数据库中的记录。

```
CALL ml_add_connection_script(
 'ver1',
 'end_download',
```

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

```
 'DELETE FROM ULEmpCust ec
  WHERE ec.emp_id = {ml s.username} AND action = ''D''')
```

Java 示例

以下对 MobiLink 系统过程的调用在同步脚本版本 ver1 时将名为 endDownloadConnection 的 Java 方法注册为 end_download 连接事件的脚本。

```
CALL ml_add_java_connection_script(
  'ver1',
  'end_download',
  'ExamplePackage.ExampleClass.endDownloadConnection' )
```

以下是 Java 方法 endDownloadConnection 示例。ULEmpCust 表有一个操作列。以下脚本使用此列中的值删除远程数据库中的记录。它还使用当前 MobiLink 连接（以前保存的）来在下载结束之前执行更新。此 SQL 语法用于 SQL Anywhere 统一数据库。

```
public String endDownloadConnection(
  Timestamp ts,
  String user )
  throws java.sql.SQLException {
  String del_sql = "DELETE FROM ULEmpCust ec " +
    "WHERE ec.emp_id = '" + user + "' " +
    "AND action = 'D' ";
  execUpdate( _syncConn, del_sql );
  return ( null );
}
```

.NET 示例

以下对 MobiLink 系统过程的调用在同步脚本版本 ver1 时将名为 EndDownload 的 .NET 方法注册为 end_download 连接事件的脚本。

```
CALL ml_add_dnet_connection_script(
 'ver1',
 'end_download',
 'TestScripts.Test.EndDownload' )
```

以下是 .NET 方法 EndDownload 示例。ULEmpCust 表有一个操作列。以下脚本使用此列中的值删除远程数据库中的记录。它还使用当前 MobiLink 连接（以前保存的）来在下载结束之前执行更新。此 SQL 语法用于 SQL Anywhere 统一数据库。

```
public string EndDownload(
  DateTime timestamp,
  string user ) {
  string del_sql = "DELETE FROM ULEmpCust ec " +
   "WHERE ec.emp_id = '" + user + "' " +
   "AND action = 'D' ";
   execUpdate( _syncConn, del_sql );
  return ( null );
}
```

end_download 表事件

在 MobiLink 服务器完成下载的插入、更新和删除操作的流的准备工作后立即处理与特定的表有关的语句。

参数

在下表中，说明部分提供 SQL 数据类型。如果您使用 Java 或 .NET 编写脚本，则应该使用相应的数据类型。请参见“SQL-Java 数据类型”一节第 498 页和“SQL-.NET 数据类型”一节第 556 页。

在 SQL 脚本中，可以使用名称或问号指定事件参数，但不能在一个脚本中混合使用名称和问号。如果使用问号，则参数必须按照如下所示的顺序并且仅当没有指定任何后继参数时才是可选的（例如，如果您想使用参数 2，则必须使用参数 1）。如果使用命名参数，则可以按照任何顺序指定任意参数子集。

SQL 脚本的参数名称	说明	顺序
s.last_table_download	TIMESTAMP。表的上一次下载时间。	1
s.remote_id	VARCHAR(128)。MobiLink 远程 ID。只有在使用命名参数时才能引用远程 ID。	不适用
s.username	VARCHAR(128)。MobiLink 用户名。	2
s.table	VARCHAR(128)。表名。	3

缺省操作

无。

注释

在完成所有行的下载并已收到接收确认之后，MobiLink 服务器将执行此脚本。下载信息是在单独的一个事务中进行准备的。执行此脚本是此事务中的最后一步表特定的非统计操作。

您可以为远程数据库的每个表编写一个 end_download 脚本。

另请参见

- “脚本参数”一节第 299 页
- “添加和删除脚本”一节第 306 页
- “begin_download 表事件”一节第 347 页
- “end_download 连接事件”一节第 384 页
- “在脚本中使用远程 ID 和 MobiLink 用户名”一节 《MobiLink - 客户端管理》
- “在脚本中使用上次下载时间”一节第 122 页

SQL 示例

end_download 表事件用于执行在下载特定表后所需执行的任何步骤。以下 SQL Anywhere SQL 脚本删除由 prepare_for_download 脚本创建的临时表，该临时表用以存放 sales_summary 表的下载行。

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

```
CALL ml_add_table_script(
 'MyCorp 1.0',
 'sales_summary',
 'end_download',
 'DROP TABLE #sales_summary_download' )
```

Java 示例

以下对 MobiLink 系统过程的调用在同步脚本版本 ver1 时将名为 endDownloadTable 的 Java 方法注册为 end_download 表事件的脚本。

```
CALL ml_add_java_table_script (
  'ver1',
  'table1',
  'end_download',
  'ExamplePackage.ExampleClass.endDownloadTable' )
```

以下是 Java 方法 endDownloadTable 示例。它将重置当前表成员变量。

```
public String endDownloadTable(
  Timestamp ts,
  String user,
  String table ) {
  _curTable = null;
  return ( null );
}
```

.NET 示例

以下对 MobiLink 系统过程的调用在同步脚本版本 ver1 和表 table1 时将名为 EndTableDownload 的 .NET 方法注册为 end_download 表事件的脚本。

```
CALL ml_add_dnet_table_script(
  'ver1',
  'table1',
  'end_download',
  'TestScripts.Test.EndTableDownload'
)
```

以下是 .NET 方法 EndTableDownload 示例。它将重置当前表成员变量。

```
public string EndTableDownload
  DateTime timestamp,
  string user,
  string table ) {
  _curTable = null;
  return ( null );
}
```

end_download_deletes 表事件

在准备好将要从远程数据库的指定表中删除的行的列表后立即处理有关该特定表的语句。

参数

在下表中，说明部分提供 SQL 数据类型。如果您使用 Java 或 .NET 编写脚本，则应该使用相应的数据类型。请参见“SQL-Java 数据类型”一节第 498 页和“SQL-.NET 数据类型”一节第 556 页。

在 SQL 脚本中，可以使用名称或问号指定事件参数，但不能在一个脚本中混合使用名称和问号。如果使用问号，则参数必须按照如下所示的顺序并且仅当没有指定任何后继参数时才是可选的（例如，如果您想使用参数 2，则必须使用参数 1）。如果使用命名参数，则可以按照任何顺序指定任意参数子集。

SQL 脚本的参数名称	说明	顺序
s.last_table_download	TIMESTAMP。表的上一次下载时间。	1
s.remote_id	VARCHAR(128)。MobiLink 远程 ID。只有在使用命名参数时才能引用远程 ID。	不适用
s.username	VARCHAR(128)。MobiLink 用户名。	2
s.table	VARCHAR(128)。表名。	3

缺省操作

无。

注释

在准备好将要从远程数据库的指定的表中删除的行的列表后立即执行此脚本。

您可以为远程数据库的每个表编写一个 end_download_deletes 脚本。

另请参见

- “脚本参数”一节第 299 页
- “添加和删除脚本”一节第 306 页
- “begin_download_deletes 表事件”一节第 349 页
- “end_download 连接事件”一节第 384 页
- “begin_download_rows 表事件”一节第 352 页
- “end_download_rows 表事件”一节第 391 页
- “download_delete_cursor 表事件”一节第 373 页
- “在脚本中使用远程 ID 和 MobiLink 用户名”一节 《MobiLink - 客户端管理》
- “在脚本中使用上次下载时间”一节第 122 页

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

SQL 示例

您可能要在远程数据库上将某一行标记为已删除。以下脚本更新统一数据库中名为 OnRemote 的列。

> **注意**
> UPDATE 上的 WHERE 子句与用于 download_delete_cursor 事件脚本的 WHERE 子句匹配。

```
CALL ml_add_table_script(
 'version1',
 'Leads',
 'end_download_deletes',
 'UPDATE Leads SET OnRemote = 0
  WHERE LastModified >= {ml s.last_table_download}
   AND Owner = {ml s.username} AND DeleteFlag=1');
```

Java 示例

以下对 MobiLink 系统过程的调用在同步脚本版本 ver1 时将名为 endDownloadDeletes 的 Java 方法注册为 end_download_deletes 表事件的脚本。

```
CALL ml_add_java_table_script(
  'ver1',
  'table1',
  'end_download_deletes',
  'ExamplePackage.ExampleClass.endDownloadDeletes' )
```

您可能要在远程数据库上将某一行标记为已删除。以下是 Java 方法 endDownloadDeletes 示例。它更新统一数据库中名为 OnRemote 的列，以指示该记录不再位于远程数据库上。

> **注意**
> UPDATE 上的 WHERE 子句与用于 download_delete_cursor 事件脚本的 WHERE 子句匹配。

```
public String endDownloadDeletes(
  Timestamp ts,
  String user,
  String table ) {
  return( "UPDATE Leads SET OnRemote = 0
     WHERE LastModified >= {ml s.last_table_download}
     AND Owner = {ml s.username} AND DeleteFlag=1" );
}
```

.NET 示例

以下对 MobiLink 系统过程的调用在同步脚本版本 ver1 和表 table1 时将名为 EndDownloadDeletes 的 .NET 方法注册为 end_download_deletes 表事件的脚本。

```
CALL ml_add_dnet_table_script(
  'ver1',
  'table1',
  'end_download_deletes',
  'TestScripts.Test.EndDownloadDeletes'
)
```

您可能要在远程数据库上将某一行标记为已删除。以下是 .NET 方法 EndDownloadDeletes 示例。它更新统一数据库中名为 OnRemote 的列，以指示该记录不再位于远程数据库上。UPDATE 上的 WHERE 子句与用于 download_delete_cursor 事件脚本的 WHERE 子句匹配。

```
public string EndDownloadDeletes(
  DateTime timestamp,
  string user,
  string table) {
  return( "UPDATE Leads SET OnRemote = 0
     WHERE LastModified >= {ml s.last_table_download}
     AND Owner = {ml s.username} AND DeleteFlag=1" );
}
```

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

end_download_rows 表事件

在准备好将要在远程数据库的特定表中插入或更新的行的列表后立即处理有关该特定表的语句。

参数

在下表中，说明部分提供 SQL 数据类型。如果您使用 Java 或 .NET 编写脚本，则应该使用相应的数据类型。请参见“SQL-Java 数据类型”一节第 498 页和“SQL-.NET 数据类型”一节第 556 页。

在 SQL 脚本中，可以使用名称或问号指定事件参数，但不能在一个脚本中混合使用名称和问号。如果使用问号，则参数必须按照如下所示的顺序并且仅当没有指定任何后继参数时才是可选的（例如，如果您想使用参数 2，则必须使用参数 1）。如果使用命名参数，则可以按照任何顺序指定任意参数子集。

SQL 脚本的参数名称	说明	顺序
s.last_table_download	TIMESTAMP。表的上一次下载时间。	1
s.remote_id	VARCHAR(128)。MobiLink 远程 ID。只有在使用命名参数时才能引用远程 ID。	不适用
s.username	VARCHAR(128)。MobiLink 用户名。	2
s.table	VARCHAR(128)。表名。	3

缺省操作

无。

注释

在准备好将要在远程数据库的指定表中插入或更新的行的流后立即执行此脚本。

您可以为远程数据库的每个表编写一个 end_download_rows 脚本。

另请参见

- “脚本参数”一节第 299 页
- “添加和删除脚本”一节第 306 页
- “begin_download_rows 表事件”一节第 352 页
- “end_download 连接事件”一节第 384 页
- “end_download_deletes 表事件”一节第 388 页
- “begin_download_deletes 表事件”一节第 349 页
- “在脚本中使用远程 ID 和 MobiLink 用户名”一节 《MobiLink - 客户端管理》
- “在脚本中使用上次下载时间”一节第 122 页

SQL 示例

您可能希望将一行标记为已成功下载到远程数据库。以下脚本更新统一数据库中名为 OnRemote 的列。

注意
UPDATE 上的 WHERE 子句与用于 download_delete_cursor 事件脚本的 WHERE 子句匹配。

```
CALL ml_add_table_script(
   'version1',
   'Leads',
   'end_download_rows',
   'UPDATE Leads SET OnRemote = 1
    WHERE LastModified >= {ml s.last_table_download}
     AND Owner = {ml s.username}
     AND DownloadFlag=1' );
```

Java 示例

以下对 MobiLink 系统过程的调用在同步脚本版本 ver1 时将名为 endDownloadRows 的 Java 方法注册为 end_download_rows 表事件的脚本。

```
CALL ml_add_java_table_script(
   'ver1',
   'table1',
   'end_download_rows',
   'ExamplePackage.ExampleClass.endDownloadRows' )
```

以下是 Java 方法 endDownloadRows 示例。它将消息输出到 MobiLink 消息日志。（请注意：将消息输出到 MobiLink 消息日志在开发时可能会有帮助，但会降低生产服务器的性能。）

```
public String endDownloadRows(
  Timestamp ts,
  String user,
  String table ) {
  java.lang.System.out.println(
   "Done downloading inserts and updates for table "
   + table );
  return ( null );
}
```

.NET 示例

以下对 MobiLink 系统过程的调用在同步脚本版本 ver1 和表 table1 时将名为 EndDownloadRows 的 .NET 方法注册为 end_download_rows 表事件的脚本。

```
CALL ml_add_dnet_table_script(
 'ver1',
 'table1',
 'end_download_rows',
 'TestScripts.Test.EndDownloadRows'
)
```

以下是 .NET 方法 EndDownloadRows 示例。它将消息输出到 MobiLink 消息日志。（请注意：将消息输出到 MobiLink 消息日志在开发时可能会有帮助，但会降低生产服务器的性能。）

```
public string EndDownloadRows(
  DateTime timestamp,
  string user,
  string table ) {
  System.Console.WriteLine(
   "Done downloading inserts and updates for table "
   + table );
```

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

```
    return ( null );
}
```

end_publication 连接事件

提供有关正被同步的发布的有用信息。

参数

在下表中，说明部分提供 SQL 数据类型。如果您使用 Java 或 .NET 编写脚本，则应该使用相应的数据类型。请参见“SQL-Java 数据类型”一节第 498 页和“SQL-.NET 数据类型”一节第 556 页。

在 SQL 脚本中，可以使用名称或问号指定事件参数，但不能在一个脚本中混合使用名称和问号。如果使用问号，则参数必须按照如下所示的顺序并且仅当没有指定任何后继参数时才是可选的（例如，如果您想使用参数 2，则必须使用参数 1）。如果使用命名参数，则可以按照任何顺序指定任意参数子集。

SQL 脚本的参数名称	说明	顺序
s.generation_number	INTEGER。如果部署不使用基于文件的下载，则可以忽略此参数。缺省值是 1。	1
s.remote_id	VARCHAR(128)。MobiLink 远程 ID。只有在使用已命名的参数时才能引用远程 ID。	不适用
s.username	VARCHAR(128)。MobiLink 用户名。	2
s.publication_name	VARCHAR(128)	3
s.last_publication_upload	TIMESTAMP。此发布上次成功上载的时间。	4
s.last_publication_download	TIMESTAMP。此发布的上一次下载时间。	5
s.subscription_id	VARCHAR(128)。预订 ID。	6

缺省操作

无。

注释

此事件使您可以根据当前正在同步的发布来设计同步逻辑。此事件与 end_synchronization 事件在相同的事务中并在 end_synchronization 事件之前被调用。对于当前正在同步的每个发布，将调用此事件一次。

如果当前同步成功应用了上载，则 last_upload 参数将包含应用此最新上载的时间。如果使用阻塞下载确认且当前同步获得成功下载确认，则 last_download 时间包含生成此最新下载的时间。该值与作为上次下载时间传递给下载脚本的值相同。

如果一个 UltraLite 远程数据库在与 UL_SYNC_ALL 进行同步，则将使用 'unknown' 这一名称调用此事件一次。

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

世代号

generation_number 参数专用于基于文件的下载。

世代号的输出值从 begin_publication 脚本传递到 end_publication 脚本。generation_number 的意义取决于是否在使用当前同步来创建一个下载文件，或者当前同步是否具有上载。

在基于文件的下载中，世代号用于强制在下载之前进行上载。该编号存储在下载文件中。

另请参见

- “脚本参数”一节第 299 页
- “添加和删除脚本”一节第 306 页
- “begin_publication 连接事件”一节第 355 页
- “MobiLink 基于文件的下载”第 275 页
- “在脚本中使用远程 ID 和 MobiLink 用户名”一节 《MobiLink - 客户端管理》
- “在脚本中使用上次下载时间”一节第 122 页

SQL 示例

您可能要为每个正被同步的发布记录该信息。以下示例调用 ml_add_connection_script，将事件指派给名为 RecordPubEndSync 的存储过程。

```
CALL ml_add_connection_script(
   'version1',
   'end_publication',
   'CALL RecordPubEndSync(
    {ml s.generation_number},
    {ml s.username},
    {ml s.publication_name},
    {ml s.last_publication_upload},
    {ml s.last_publication_download} )' );
```

Java 示例

以下对 MobiLink 系统过程的调用在同步脚本版本 ver1 时将名为 endPublication 的 Java 方法注册为 end_publication 连接事件的脚本。

```
CALL ml_add_java_connection_script(
   'ver1',
   'end_publication',
   'ExamplePackage.ExampleClass.endPublication' )
```

以下是 Java 方法 endPublication 示例。它将向 MobiLink 消息日志输出一条消息。（请注意：将消息输出到 MobiLink 消息日志在开发时可能会有帮助，但会降低生产服务器的性能。）

```
public String endPublication(
  ianywhere.ml.script.InOutInteger generation_number,
  String user,
  String pub_name,
  Timestamp last_publication_upload,
  Timestamp last_publication_download ) {
  java.lang.System.out.println(
   "Finished synchronizing publication " + pub_name );
  return ( null );
}
```

.NET 示例

以下对 MobiLink 系统过程的调用在同步脚本版本 ver1 时将名为 EndPub 的 .NET 方法注册为 end_publication 连接事件的脚本。

```
CALL ml_add_dnet_connection_script(
  'ver1',
  'end_publication',
  'TestScripts.Test.EndPub'
)
```

以下是 .NET 方法 endPub 示例。它将向 MobiLink 消息日志输出一条消息。（请注意：将消息输出到 MobiLink 消息日志在开发时可能会有帮助，但会降低生产服务器的性能。）

```
public string EndPub(
  ref int generation_number,
  string user,
  string pub_name,
  DateTime last_publication_upload,
  DateTime last_publication_download ) {
  System.Console.Write(
   "Finished synchronizing publication " + pub_name );
  return ( null );
}
```


Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

end_synchronization 连接事件

在应用程序完成同步过程以后断开与 MobiLink 服务器的连接时处理任何语句。

参数

在下表中，说明部分提供 SQL 数据类型。如果您使用 Java 或 .NET 编写脚本，则应该使用相应的数据类型。请参见“SQL-Java 数据类型”一节第 498 页和“SQL-.NET 数据类型”一节第 556 页。

SQL 脚本的参数名称	说明	顺序
s.remote_id	VARCHAR(128)。MobiLink 远程 ID。只有在使用命名参数时才能引用远程 ID。	不适用
s.username	VARCHAR(128)。MobiLink 用户名。	1
s.synchronization_ok	INTEGER。对于成功的同步，此值为 1；对于不成功的同步，此值为 0。	2

缺省操作

无。

注释

同步完成后，MobiLink 服务器将执行此脚本。如果使用阻塞下载确认，则在 MobiLink 客户端已将下载接收确认返回之后执行此脚本。

此脚本将在下载事务后的一个单独事务中执行。

end_synchronization 脚本对于维护统计信息很有用。这是因为如果调用 begin_synchronization 脚本，则即使存在错误或冲突，也会调用 end_synchronization 脚本，从而使回退上载事务时，统计信息可得到维护。

另请参见

- “脚本参数”一节第 299 页
- “添加和删除脚本”一节第 306 页
- “begin_synchronization 连接事件”一节第 358 页
- “begin_synchronization 表事件”一节第 360 页
- “end_synchronization 表事件”一节第 399 页
- “在脚本中使用远程 ID 和 MobiLink 用户名”一节 《MobiLink - 客户端管理》

SQL 示例

以下 SQL 脚本调用一个系统过程，该过程记录同步尝试的结束时间及其成功或失败状态。此语法用于 SQL Anywhere 统一数据库。

```
CALL ml_add_connection_script(
 'ver1',
 'end_synchronization',
 'CALL RecordEndOfSyncAttempt(
```

```
    {ml s.username},
    {ml s.synchronization_ok} )' )
```

Java 示例

以下对 MobiLink 系统过程的调用在同步脚本版本 ver1 时将名为 endSynchronizationConnection 的 Java 方法注册为 end_synchronization 事件的脚本。

```
CALL ml_add_java_connection_script(
  'ver1',
  'end_synchronization',
  'ExamplePackage.ExampleClass.endSynchronizationConnection'
)
```

以下是 Java 方法 endSynchronizationConnection 示例。它使用 JDBC 连接执行更新。此语法用于 SQL Anywhere 统一数据库。

```
public String endSynchronizationConnection(
  String user )
  throws java.sql.SQLException {
  execUpdate( _syncConn,
   "UPDATE sync_count set count = count + 1 where user_id = '"
   + user + "' " );
  return ( null );
}
```

.NET 示例

以下对 MobiLink 系统过程的调用在同步脚本版本 ver1 时将名为 EndSync 的 .NET 方法注册为 end_synchronization 连接事件的脚本。

```
CALL ml_add_dnet_connection_script(
  'ver1',
  'end_synchronization',
  'TestScripts.Test.EndSync'
)
```

以下是 .NET 方法 EndSync 示例。它更新表 sync_count。此语法用于 SQL Anywhere 统一数据库。

```
public string EndSync(
  string user ) {
  return(
   "UPDATE sync_count set count = count + 1 where user_id = '"
   + user + "' ");
  return ( null );
}
```

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

end_synchronization 表事件

在应用程序完成同步过程以后断开与 MobiLink 服务器的连接时处理与特定的表有关的语句。

参数

在下表中，说明部分提供 SQL 数据类型。如果您使用 Java 或 .NET 编写脚本，则应该使用相应的数据类型。请参见“SQL-Java 数据类型”一节第 498 页和“SQL-.NET 数据类型”一节第 556 页。

在 SQL 脚本中，可以使用名称或问号指定事件参数，但不能在一个脚本中混合使用名称和问号。如果使用问号，则参数必须按照如下所示的顺序并且仅当没有指定任何后继参数时才是可选的（例如，如果您想使用参数 2，则必须使用参数 1）。如果使用命名参数，则可以按照任何顺序指定任意参数子集。

SQL 脚本的参数名称	说明	顺序
s.remote_id	VARCHAR(128)。MobiLink 远程 ID。只有在使用命名参数时才能引用远程 ID。	不适用
s.username	VARCHAR(128)。MobiLink 用户名。	1
s.table	VARCHAR(128)。表名。	2
s.synchronization_ok	INTEGER。对于成功的同步，此值为 1；对于不成功的同步，此值为 0。	3

缺省操作

无。

注释

在应用程序完成同步过程并即将与 MobiLink 服务器断开连接时，MobiLink 服务器将在执行同名的连接级别脚本之前执行此脚本。

您可以为远程数据库的每个表编写一个 end_synchronization 脚本。

另请参见

- “脚本参数”一节第 299 页
- “添加和删除脚本”一节第 306 页
- “begin_synchronization 表事件”一节第 360 页
- “end_synchronization 连接事件”一节第 397 页
- “end_synchronization 表事件”一节第 399 页
- “在脚本中使用远程 ID 和 MobiLink 用户名”一节 《MobiLink - 客户端管理》

SQL 示例

以下 SQL Anywhere SQL 脚本删除由 begin_synchronization 脚本创建的临时表。

Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

```
CALL ml_add_table_script(
 'ver1',
 'sales_order',
 'end_synchronization',
 'DROP TABLE #sales_order' )
```

Java 示例

以下对 MobiLink 系统过程的调用在同步脚本版本 ver1 时将名为 endSynchronizationTable 的 Java 方法注册为 end_synchronization 表事件的脚本。

```
CALL ml_add_java_table_script(
   'ver1',
   'table1',
   'end_synchronization',
   'ExamplePackage.ExampleClass.endSynchronizationTable' )
```

以下是 Java 方法 endSynchronizationTable 示例。它返回一条 SQL 语句，此语句删除由 begin_synchronization 脚本创建的临时表。

```
public String endSynchronizationTable(
  String user,
  String table ) {
  return( "DROP TABLE #sales_order" );
}
```

.NET 示例

以下对 MobiLink 系统过程的调用在同步脚本版本 ver1 和表 table1 时将名为 EndTableSync 的 .NET 方法注册为 end_synchronization 表事件的脚本。

```
CALL ml_add_dnet_table_script(
 'ver1',
 'table1',
 'end_synchronization',
 'TestScripts.Test.EndTableSync'
)
```

以下是 .NET 方法 EndTableSync 示例。它返回一条 SQL 语句，此语句删除由 begin_synchronization 脚本创建的临时表。

```
public string EndTableSync(
  string user,
  string table ) {
  return( "DROP TABLE #sales_order" );
}
```

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

end_upload 连接事件

在 MobiLink 服务器处理完上载的插入、更新和删除操作后立即处理任何语句。

参数

在下表中，说明部分提供 SQL 数据类型。如果您使用 Java 或 .NET 编写脚本，则应该使用相应的数据类型。请参见“SQL-Java 数据类型”一节第 498 页和“SQL-.NET 数据类型”一节第 556 页。

SQL 脚本的参数名称	说明	顺序
s.remote_id	VARCHAR(128)。MobiLink 远程 ID。只有在使用命名参数时才能引用远程 ID。	不适用
s.username	VARCHAR(128)。MobiLink 用户名。	1

缺省操作

无。

注释

MobiLink 服务器执行此脚本是对上载的信息进行处理的最后一步。上载信息在单独的事务中进行处理。此脚本的执行是执行统计脚本之前此事务中的最后一步操作。

另请参见

- “脚本参数”一节第 299 页
- “添加和删除脚本”一节第 306 页
- “begin_upload 连接事件”一节第 362 页
- “end_upload 表事件”一节第 403 页
- “在脚本中使用远程 ID 和 MobiLink 用户名”一节 《MobiLink - 客户端管理》

SQL 示例

以下 SQL Anywhere SQL 脚本调用 EndUpload 存储过程。

```
CALL ml_add_connection_script(
 'ver1',
 'sales_order',
 'end_upload',
 'CALL EndUpload({ml s.username});' )
```

Java 示例

以下对 MobiLink 系统过程的调用在同步脚本版本 ver1 时将名为 endUploadConnection 的 Java 方法注册为 end_upload 连接事件的脚本。

```
CALL ml_add_java_connection_script(
   'ver1',
   'end_upload',
   'ExamplePackage.ExampleClass.endUploadConnection' )
```

Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

以下是 Java 方法 endUploadConnection 示例。它调用一个方法以在数据库上执行操作。

```
public String endUploadConnection( String user ) {
  // Clean up new and old tables.
  Iterator    two_iter = _tables_with_ops.iterator();
  while( two_iter.hasNext() ) {
    TableInfo cur_table = (TableInfo)two_iter.next();
    dumpTableOps( _sync_conn, cur_table );
  }
  _tables_with_ops.clear();
  return ( null );
}
```

.NET 示例

以下对 MobiLink 系统过程的调用在同步脚本版本 ver1 时将名为 EndUpload 的 .NET 方法注册为 end_upload 连接事件的脚本。

```
CALL ml_add_dnet_connection_script(
  'ver1',
  'end_upload',
  'TestScripts.Test.EndUpload'
)
```

以下是 .NET 方法 EndUpload 示例。该方法返回调用 EndUpload 存储过程的 SQL 语句。

```
public string EndUpload( string user ) {
  return ( "CALL EndUpload({ml s.username});" );
}
```

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

end_upload 表事件

在 MobiLink 服务器处理完上载的插入、更新和删除操作的流之后立即处理与特定表有关的语句。

参数

在下表中，说明部分提供 SQL 数据类型。如果您使用 Java 或 .NET 编写脚本，则应该使用相应的数据类型。请参见“SQL-Java 数据类型”一节第 498 页和“SQL-.NET 数据类型”一节第 556 页。

在 SQL 脚本中，可以使用名称或问号指定事件参数，但不能在一个脚本中混合使用名称和问号。如果使用问号，则参数必须按照如下所示的顺序并且仅当没有指定任何后继参数时才是可选的（例如，如果您想使用参数 2，则必须使用参数 1）。如果使用命名参数，则可以按照任何顺序指定任意参数子集。

参数	说明	顺序
s.remote_id	VARCHAR(128)。MobiLink 远程 ID。只有在使用命名参数时才能引用远程 ID。	不适用
s.username	VARCHAR(128)。MobiLink 用户名。	1
s.table	VARCHAR(128)。表名。	2

缺省操作

无。

注释

MobiLink 服务器执行此脚本是对上载的信息进行处理的最后一步。上载信息在单独的事务中进行处理。此脚本的执行是此事务中最后一步表特定的操作。

您可以为远程数据库的每个表编写一个 end_upload 脚本。

另请参见

- “脚本参数”一节第 299 页
- “添加和删除脚本”一节第 306 页
- “begin_upload 表事件”一节第 364 页
- “end_upload 连接事件”一节第 401 页
- “在脚本中使用远程 ID 和 MobiLink 用户名”一节 《MobiLink - 客户端管理》

SQL 示例

以下对 MobiLink 系统过程的调用将 end_upload 事件指派给名为 ULCustomerIDPool_maintain 的存储过程。

```
CALL ml_add_table_script(
   'custdb',
   'ULCustomerIDPool',
   'end_upload',
   'CALL ULCustomerIDPool_maintain( username );' );
```

Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

以下 SQL 语句创建 ULCustomerIDPool_maintain 存储过程。

```
CREATE OR REPLACE PROCEDURE ULCustomerIDPool_maintain(
  SyncUserID IN integer )
AS
  pool_count INTEGER;
  pool_max   INTEGER;
BEGIN
  -- Determine how many ids to add to the pool
  SELECT COUNT(*)
    INTO pool_count
    FROM ULCustomerIDPool
    WHERE pool_emp_id = SyncUserID;
  -- Determine the current Customer id max
  SELECT MAX(pool_cust_id)
    INTO pool_max
    FROM ULCustomerIDPool;
  -- Top up the pool with new ids
  WHILE pool_count < 20 LOOP
    pool_max := pool_max + 1;
      INSERT INTO ULCustomerIDPool(
          pool_cust_id, pool_emp_id )
            VALUES ( pool_max, SyncUserID );
      pool_count := pool_count + 1;
  END LOOP;
END;
```

Java 示例

以下对 MobiLink 系统过程的调用在同步脚本版本 ver1 时将名为 endUploadTable 的 Java 方法注册为 end_upload 表事件的脚本。

```
CALL ml_add_java_table_script(
   'ver1',
   'table1',
   'end_upload',
   'ExamplePackage.ExampleClass.endUploadTable' )
```

以下是 Java 方法 endUploadTable 示例。它为表生成一个删除操作，该表的名称与传入的表名相关。此语法用于 SQL Anywhere 统一数据库。

```
public String endUploadTable(
  String user,
  String table ) {
  return( "DELETE from '" + table + "_temp'" );
}
```

.NET 示例

以下对 MobiLink 系统过程的调用在同步脚本版本 ver1 和表 table1 时将名为 EndUpload 的 .NET 方法注册为 end_upload 表事件的脚本。

```
CALL ml_add_dnet_table_script(
   'ver1',
   'table1',
   'end_upload',
   'TestScripts.Test.EndUpload'
)
```

以下 .NET 示例将插入临时表的行移动到传入脚本的表中。

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

```
public string EndUpload( string user, string table ) {
  DBCommand stmt = curConn.CreateCommand();
  // Move the uploaded rows to the destination table.
  stmt.CommandText = "INSERT INTO "
    + table
    + " SELECT * FROM dnet_ul_temp";
  stmt.ExecuteNonQuery();
  stmt.Close();
  return ( null );
}
```

end_upload_deletes 表事件

在应用了从远程数据库中的特定表上载的删除后立即执行有关该特定表的语句。

参数

在下表中，说明部分提供 SQL 数据类型。如果您使用 Java 或 .NET 编写脚本，则应该使用相应的数据类型。请参见“SQL-Java 数据类型”一节第 498 页和“SQL-.NET 数据类型”一节第 556 页。

在 SQL 脚本中，可以使用名称或问号指定事件参数，但不能在一个脚本中混合使用名称和问号。如果使用问号，则参数必须按照如下所示的顺序并且仅当没有指定任何后继参数时才是可选的（例如，如果您想使用参数 2，则必须使用参数 1）。如果使用命名参数，则可以按照任何顺序指定任意参数子集。

SQL 脚本的参数名称	说明	顺序
s.remote_id	VARCHAR(128)。MobiLink 远程 ID。只有在使用命名参数时才能引用远程 ID。	不适用
s.username	VARCHAR(128)。MobiLink 用户名。	1
s.table	VARCHAR(128)。表名。	2

缺省操作

无。

注释

此脚本将在应用了远程表（由第二个参数指定）中行删除操作引起的更改后立即执行。

您可以为远程数据库的每个表编写一个 end_upload_deletes 脚本。

另请参见

- “脚本参数”一节第 299 页
- “添加和删除脚本”一节第 306 页
- “begin_upload_deletes 表事件”一节第 366 页
- “在脚本中使用远程 ID 和 MobiLink 用户名”一节 《MobiLink - 客户端管理》

SQL 示例

您可以在上载过程中使用此事件在中间表上处理删除的行。您可以比较基表和中间表中的行，从而决定如何处理删除的行。

以下对 MobiLink 系统过程的调用将 EndUploadDeletesLeads 存储过程指派给 end_upload_deletes 事件。

```
CALL ml_add_table_script(
  'version1',
  'Leads',
```

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

```
  'end_upload_deletes',
  'call EndUploadDeletesLeads()');
```

以下 SQL 语句创建 EndUploadDeletes 存储过程。

```
CREATE PROCEDURE EndUploadDeletesLeads ( )
Begin
  FOR names AS curs CURSOR FOR
  SELECT LeadID
    FROM Leads
    WHERE LeadID NOT IN (SELECT LeadID FROM T_Leads);
  DO
   CALL decide_what_to_do( LeadID )
  END FOR;
end
```

Java 示例

以下对 MobiLink 系统过程的调用在同步脚本版本 ver1 时将名为 endUploadDeletes 的 Java 方法注册为 end_upload_deletes 表事件的脚本。

```
CALL ml_add_java_table_script(
   'ver1',
   'table1',
   'end_upload_deletes',
   'ExamplePackage.ExampleClass.endUploadDeletes' )
```

以下是 Java 方法 endUploadDeletes 示例。它调用操纵数据库的 Java 方法。

```
public String endUploadDeletes(
  String user,
  String table )
  throws java.sql.SQLException {
  processUploadedDeletes( _syncConn, table );
  return ( null );
}
```

.NET 示例

以下对 MobiLink 系统过程的调用在同步脚本版本 ver1 和表 table1 时将名为 EndUploadDeletes 的 .NET 方法注册为 end_upload_deletes 表事件的脚本。

```
CALL ml_add_dnet_table_script(
   'ver1',
   'table1',
   'end_upload_deletes',
   'TestScripts.Test.EndUploadDeletes'
)
```

以下是 .NET 方法 EndUploadDeletes 示例。它调用操纵数据库的 .NET 方法。

```
public string EndUploadDeletes(
  string user,
  string table ) {
  processUploadedDeletes( _syncConn, table );
  return ( null );
}
```

end_upload_rows 表事件

在应用远程数据库特定表中上载的插入和更新后立即执行有关该特定表的语句。

参数

在下表中，说明部分提供 SQL 数据类型。如果您使用 Java 或 .NET 编写脚本，则应该使用相应的数据类型。请参见“SQL-Java 数据类型”一节第 498 页和“SQL-.NET 数据类型”一节第 556 页。

在 SQL 脚本中，可以使用名称或问号指定事件参数，但不能在一个脚本中混合使用名称和问号。如果使用问号，则参数必须按照如下所示的顺序并且仅当没有指定任何后继参数时才是可选的（例如，如果您想使用参数 2，则必须使用参数 1）。如果使用命名参数，则可以按照任何顺序指定任意参数子集。

SQL 脚本的参数名称	说明	顺序
s.remote_id	VARCHAR(128)。MobiLink 远程 ID。只有在使用命名参数时才能引用远程 ID。	不适用
s.username	VARCHAR(128)。MobiLink 用户名。	1
s.table	VARCHAR(128)。表名。	2

缺省操作

无。

注释

上载的信息可能要求在统一数据库中插入或更新行。此脚本将在应用了由远程表（由第二个参数指定）中的修改而引起的更改后立即运行。

您可以为远程数据库的每个表编写一个 end_upload_rows 脚本。

另请参见

- “添加和删除脚本”一节第 306 页
- “begin_upload_rows 表事件”一节第 368 页
- “在脚本中使用远程 ID 和 MobiLink 用户名”一节 《MobiLink - 客户端管理》

SQL 示例

以下对 MobiLink 系统过程的调用在同步脚本版本 ver1 时将名为 endUploadRows 的 SQL 方法注册为 EndUploadRows 表事件的脚本。

```
CALL ml_add_table_script(
  'version1',
  'table1',
  'end_upload_rows',
  'CALL EndUploadRows(
      { ml s.username },
      { ml s.table } )' )
```

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

以下是 SQL 方法 EndUploadRows 示例。它调用操纵数据库的 SQL 方法。

```
CREATE PROCEDURE EndUploadRows (
 IN user VARCHAR(128),
 IN table VARCHAR(128) ),
BEGIN
   CALL decide_what_to_do(table);
END;
```

Java 示例

以下对 MobiLink 系统过程的调用在同步脚本版本 ver1 时将名为 endUploadRows 的 Java 方法注册为 end_upload_rows 表事件的脚本。

```
CALL ml_add_java_table_script(
   'ver1',
   'table1',
   'end_upload_rows',
   'ExamplePackage.ExampleClass.endUploadRows' )
```

以下是 Java 方法 endUploadRows 示例。它调用操纵数据库的 Java 方法。

```
public String endUploadRows(
  String user,
  String table )
  throws java.sql.SQLException {
  processUploadedRows( _syncConn, table );
  return ( null );
}
```

.NET 示例

以下对 MobiLink 系统过程的调用在同步脚本版本 ver1 和表 table1 时将名为 EndUploadRows 的 .NET 方法注册为 end_upload_rows 表事件的脚本。

```
CALL ml_add_dnet_table_script(
  'ver1',
  'table1',
  'end_upload_rows',
  'TestScripts.Test.EndUploadRows'
)
```

以下是 .NET 方法 endUploadRows 示例。它调用操纵数据库的 .NET 方法。

```
public string EndUploadRows(
  string user,
  string table ) {
  processUploadedRows( _syncConn, table );
  return ( null );
}
```

handle_DownloadData 连接事件

由直接行处理用来创建一组要下载的行。

参数

无。

缺省操作

无。

注释

handle_DownloadData 事件允许您确定使用直接行处理将哪些操作下载到 MobiLink 客户端。

直接行处理用于对 MobiLink 支持的统一数据库以外的数据源进行同步。请参见“直接行处理” 第 609 页。

要创建直接下载，可以使用面向 Java 或 .NET 的 MobiLink 服务器 API 中的 DownloadData 和 DownloadTableData 类。

对于 Java，DBConnectionContext getDownloadData 方法返回当前同步的 DownloadData 实例。DownloadData 封装发送到远程客户端的所有下载操作。可以使用 DownloadData getDownloadTables 和 getDownloadTableByName 方法获得 DownloadTableData 实例。DownloadTableData 封装特定表的下载操作。可以使用 getUpsertPreparedStatement 方法获得插入和更新操作的准备语句。可以使用 DownloadTableData getDeletePreparedStatement 方法获得删除操作的准备语句。

对于 .NET，DBConnectionContext GetDownloadData 方法返回当前同步的 DownloadData 实例。DownloadData 封装发送到远程客户端的所有下载操作。可以使用 DownloadData GetDownloadTables 和 GetDownloadTableByName 方法获得 DownloadTableData 实例。DownloadTableData 封装特定表的下载操作。可以使用 GetUpsertCommand 方法获得插入和更新操作的命令。可以使用 DownloadTableData getDeleteCommand 方法获得删除操作的命令。

对于 Java，请参见“DBConnectionContext 接口”一节第 508 页。对于 .NET，请参见“DBConnectionContext 接口”一节第 570 页。

可以在 handle_DownloadData 或其它同步事件中创建下载。MobiLink 提供这种灵活性，使您可以在上载数据或发生特定事件时设置下载。如果您要在 handle_DownloadData 以外的事件中创建直接下载，则必须创建一个其方法不执行任何操作的 handle_DownloadData 脚本。MobiLink 需要定义此脚本以启用直接行处理。除仅上载同步的情况外，MobiLink 服务器需要至少定义一个 handle_DownloadData 脚本。

如果在 handle_DownloadData 以外的事件中创建直接下载，则该事件不得在 begin_synchronization 事件之前，也不得在 end_download 事件之后。

注意
此事件不能作为 SQL 实现。

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

另请参见

- “直接行处理”第 609 页
- “处理直接下载”一节第 620 页
- Java： “DownloadData 接口”一节第 513 页
- Java： “DownloadTableData 接口”一节第 515 页
- .NET： “DownloadData 接口”一节第 582 页
- .NET： “DownloadTableData 接口”一节第 584 页
- “handle_UploadData 连接事件”一节第 422 页
- “必需的脚本”一节第 305 页
- “添加和删除脚本”一节第 306 页

Java 示例

以下对 MobiLink 系统过程的调用在同步脚本版本 ver1 时，为 handle_DownloadData 连接事件注册名为 handleDownload 的 Java 方法。针对您的 MobiLink 统一数据库运行此系统过程。

```
CALL ml_add_java_connection_script(
 'ver1',
 'handle_DownloadData',
 'MyPackage.MyClass.handleDownload' )
```

请参见“ml_add_java_connection_script 系统过程”一节第 630 页。

以下示例显示如何使用 handleDownload 方法创建下载。

以下代码在名为 MobiLinkOrders 的类的构造函数中建立了一个类级别 DBConnectionContext 实例。

```
import ianywhere.ml.script.*;
import java.io.*;
import java.sql.*;
import java.lang.System;

public class  MobiLinkOrders{

  DBConnectionContext _cc;

  public MobiLinkOrders( DBConnectionContext cc ) {
    _cc = cc;
  }
```

在您的 HandleDownload 方法中，您使用 DBConnectionContext getDownloadData 方法返回当前同步的 DownloadData 实例。DownloadData getDownloadTableByName 方法返回 remoteOrders 表的 DownloadTableData 实例。DownloadTableData getUpsertPreparedStatement 方法返回一个 java.sql.PreparedStatement。要将某个操作添加到下载，需要设置所有列值并调用 executeUpdate 方法。

以下是 MobiLinkOrders 类的 handleDownload 方法。它将名为 remoteOrders 的表中的两行添加到下载。

```
// Method used for the handle_DownloadData event.
public void handleDownload() throws SQLException {
  // Get DownloadData instance for current synchronization.
  DownloadData downloadData = _cc.getDownloadData();

  // Get a DownloadTableData instance for the remoteOrders table.
  DownloadTableData td = downloadData.getDownloadTableByName("remoteOrders");
```

```
    // Get a java.sql.PreparedStatement for upsert (update/insert) operations.
    PreparedStatement upsertPS = td.getUpsertPreparedStatement();

    // Set values for one row.
    upsertPS.setInt( 1, 2300 );
    upsertPS.setInt( 2, 100 );

    // Add the values to the download.
    int updateResult = upsertPS.executeUpdate();

    // Set values for another row.
    upsertPS.setInt( 1, 2301 );
    upsertPS.setInt( 2, 50 );
    updateResult = upsertPS.executeUpdate();

    upsertPS.close();

    // ...
}
```

.NET 示例

以下对 MobiLink 系统过程的调用在同步脚本版本 ver1 时将名为 HandleDownload 的 .NET 方法注册为 handle_DownloadData 连接事件的脚本。此语法用于 SQL Anywhere 统一数据库。

```
CALL ml_add_dnet_connection_script(
   'ver1', 'handle_DownloadData',
   'TestScripts.Test.HandleDownload'
)
```

以下是 .NET 方法 HandleDownload 示例：

```
using System;
using System.Data;
using System.IO;
using iAnywhere.MobiLink.Script;
using iAnywhere.MobiLink;

namespace MyScripts
{
    /// <summary>
    /// Tests that scripts are called correctly for most sync events.
    /// </summary>
    public class MobiLinkOrders
    {
 private DBConnectionContext _cc;

 public MobiLinkOrders( DBConnectionContext cc )
 {
     _cc = cc;
 }

 ~MobiLinkOrders()
 {
 }

 public void handleDownload()
 {
     // Get DownloadData instance for current synchronization.
     DownloadData my_dd = _cc.GetDownloadData();

     // Get a DownloadTableData instance for the remoteOrders table.
     DownloadTableData td = my_dd.GetDownloadTableByName("remoteOrders");
```

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

```
        // Get an IdbCommand for upsert (update/insert) operations.
        IDbCommand upsert_stmt = td.GetUpsertCommand();

        IDataParameterCollection parameters = upsert_stmt.Parameters;

        // Set values for one row.
        parameters[ 0 ] = 2300;
        parameters[ 1 ] = 100;

        // Add the values to the download.
        int update_result = upsert_stmt.ExecuteNonQuery();

        // Set values for another row.
        parameters[ 0 ] = 2301;
        parameters[ 1 ] = 50;
        update_result = upsert_stmt.ExecuteNonQuery();

        // ...
    }

    }
}
```

handle_error 连接事件

每当 MobiLink 服务器遇到 SQL 错误时执行。

参数

在下表中，说明部分提供 SQL 数据类型。如果您使用 Java 或 .NET 编写脚本，则应该使用相应的数据类型。请参见“SQL-Java 数据类型”一节第 498 页和“SQL-.NET 数据类型”一节第 556 页。

在 SQL 脚本中，可以使用名称或问号指定事件参数，但不能在一个脚本中混合使用名称和问号。如果使用问号，则参数必须按照如下所示的顺序并且仅当没有指定任何后继参数时才是可选的（例如，如果您想使用参数 2，则必须使用参数 1）。如果使用命名参数，则可以按照任何顺序指定任意参数子集。

SQL 脚本的参数名称	说明	顺序
s.action_code	INTEGER。这是一个 INOUT 参数。	1
s.error_code	INTEGER	2
s.error_message	TEXT	3
s.remote_id	VARCHAR(128)。MobiLink 远程 ID。只有在使用命名参数时才能引用远程 ID。	不适用
s.username	VARCHAR(128)。MobiLink 用户名。	4
s.table	VARCHAR(128)。如果脚本不是表脚本，则表名为空。	5

缺省操作

在未定义 handle_error 脚本或此脚本导致错误时，缺省操作代码为 3000：回退当前事务并取消当前同步。

注释

MobiLink 服务器传递当前的操作代码。最初，对于每一组由一个单独的 SQL 操作引起的错误，该值都设置为 3000。通常情况下，每个 SQL 操作只会出现一个错误，但也可能出现多个。在一组错误中每出现一次错误时都会调用一次 handle_error 脚本。传递到第一个错误的动作代码是 3000，后续调用是在前一次调用所返回的动作代码中传递的。MobiLink 使用多个调用所返回的数值中的最大值。

您可以修改脚本中的操作代码并返回一个值以指示 MobiLink 如何继续。操作代码将告知 MobiLink 服务器下一步该做什么。在调用此脚本之前，MobiLink 服务器将操作代码设置为缺省值，该缺省值的大小取决于错误的严重程度。您可以使用脚本修改此值。脚本必须返回或设置一个操作代码。

操作代码参数可以使用以下值：

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

- **1000** 跳过当前行并继续执行。
- **3000** 回退当前事务并取消当前同步。这是缺省操作代码，在未定义 handle_error 脚本或此脚本导致错误时使用此代码。
- **4000** 回退当前事务，取消同步并关闭 MobiLink 服务器。

错误代码和消息可用于标识错误的性质。如果错误是作为同步的一部分发生的，则提供用户名。否则该值为空。

当 MobiLink 在上载事务期间处理插入、更新或删除脚本时或当其读取下载行时，如果发生 ODBC 错误，则 MobiLink 服务器执行此脚本。如果 ODBC 错误在其它时间发生，则 MobiLink 服务器调用 report_error 或 report_ODBC_error 脚本并中止同步。

如果在操作某特定表时发生错误，则提供表名称。否则该值为空。表名称是客户端应用程序中表的名称。此名称在统一数据库中是否存在直接对应的名称取决于同步系统的设计。

handle_error 事件的 SQL 脚本必须作为存储过程执行。

可以使用以下方法中的一种从 handle_error 脚本返回值：

- 将操作参数传递给存储过程的 OUTPUT 参数：

```
CALL my_handle_error( {ml s.action_code}, {ml s.error_code}, {ml
s.error_message}, {ml s.username}, {ml s.table} )
```

- 使用过程或函数返回值设置操作代码：

```
{ml s.action_code} = CALL my_handle_error( {ml s.error_code}, {ml
s.error_message}, {ml s.username}, {ml s.table} )
```

  大多数 RDBMS 会使用 RETURN 语句设置过程或函数的返回值。

CustDB 示例应用程序中包含用于各种数据库管理系统的错误处理程序。

另请参见

- “脚本参数”一节第 299 页
- “添加和删除脚本”一节第 306 页
- “report_error 连接事件”一节第 444 页
- “report_odbc_error 连接事件”一节第 447 页
- “handle_odbc_error 连接事件”一节第 418 页
- “在脚本中使用远程 ID 和 MobiLink 用户名”一节 《MobiLink - 客户端管理》

SQL 示例

以下示例在 SQL Anywhere 统一数据库中运行。它允许您的应用程序忽略冗余插入。

以下对 MobiLink 系统过程的调用将 ULHandleError 存储过程指派给 handle_error 事件。

```
CALL ml_add_connection_script(
    'ver1',
    'handle_error',
    'CALL ULHandleError(
      {ml s.action_code},
      {ml s.error_code},
      {ml s.error_message},
```

```
        {ml s.username},
        {ml s.table} )' )
```

以下 SQL 语句创建 ULHandleError 存储过程。

```
CREATE PROCEDURE ULHandleError(
   INOUT action integer,
   IN error_code integer,
   IN error_message varchar(1000),
   IN user_name varchar(128),
   IN table_name varchar(128) )
BEGIN
  -- -196 is SQLE_INDEX_NOT_UNIQUE
  -- -194 is SQLE_INVALID_FOREIGN_KEY
  IF error_code = -196 or error_code = -194 then
      -- ignore the error and keep going
      SET action = 1000;
  ELSE
      -- abort the synchronization
      SET action = 3000;
  END IF;
END
```

Java 示例

以下对 MobiLink 系统过程的调用在同步脚本版本 ver1 时将名为 handleError 的 Java 方法注册为 handle_error 连接事件的脚本。

```
CALL ml_add_java_connection_script(
    'ver1',
    'handle_error',
    'ExamplePackage.ExampleClass.handleError' )
```

以下是 Java 方法 handleError 示例。它根据传入的数据处理错误。同时还确定结果错误代码。

```
public String handleError(
  ianywhere.ml.script.InOutInteger actionCode,
  int errorCode,
  String errorMessage,
  String user,
  String table ) {
  int newAC;
  if( user == null ) {
    newAC = handleNonSyncError( errorCode,
    errorMessage ); }
  else if( table == null ) {
    newAC = handleConnectionError( errorCode,
    errorMessage, user ); }
  else {
    newAC = handleTableError( errorCode,
    errorMessage, user, table );
  }
  // Keep the most serious action code.
  if( actionCode.getValue() < newAC ) {
    actionCode.setValue( newAC );
  }
}
```

.NET 示例

以下对 MobiLink 系统过程的调用在同步脚本版本 ver1 时将名为 HandleError 的 .NET 方法注册为 handle_error 连接事件的脚本。

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

```
CALL ml_add_dnet_connection_script(
  'ver1',
  'handle_error',
  'TestScripts.Test.HandleError' )
```

以下是 .NET 方法 HandleError 示例。

```
public string HandleError() (
  ref int actionCode,
  int errorCode,
  string errorMessage,
  string user,
  string table ) {
  int new_ac;
  if( user == null ) {
    new_ac = HandleNonSyncError( errorCode,
    errorMessage ); }
  else if( table == null ) {
    new_ac = HandleConnectionError( errorCode,
    errorMessage, user ); }
  else {
    new_ac = HandleTableError( errorCode,
    errorMessage, user, table );
  }
  // Keep the most serious action code.
  if( actionCode < new_ac ) {
    actionCode = new_ac;
  }
}
```

handle_odbc_error 连接事件

每当 MobiLink 服务器遇到由 ODBC 驱动程序管理器触发的错误时都会执行。

参数

在下表中，说明部分提供 SQL 数据类型。如果您使用 Java 或 .NET 编写脚本，则应该使用相应的数据类型。请参见“SQL-Java 数据类型”一节第 498 页和“SQL-.NET 数据类型”一节第 556 页。

在 SQL 脚本中，可以使用名称或问号指定事件参数，但不能在一个脚本中混合使用名称和问号。如果使用问号，则参数必须按照如下所示的顺序并且仅当没有指定任何后继参数时才是可选的（例如，如果您想使用参数 2，则必须使用参数 1）。如果使用命名参数，则可以按照任何顺序指定任意参数子集。

SQL 脚本的参数名称	说明	顺序
s.action_code	INTEGER。这是一个 INOUT 参数。	1
s.ODBC_state	VARCHAR(5)	2
s.error_message	TEXT	3
s.remote_id	VARCHAR(128)。MobiLink 远程 ID。只有在使用命名参数时才能引用远程 ID。	不适用
s.username	VARCHAR(128)。MobiLink 用户名。	4
s.table	VARCHAR(128)	5

缺省操作

MobiLink 服务器选择缺省操作代码。您可以修改脚本中的操作代码并返回一个值以指示 MobiLink 如何继续。操作代码参数可以使用以下值：

- **1000** 跳过当前行并继续执行。
- **3000** 回退当前事务并取消当前同步。这是缺省操作代码，在未定义 handle_error 脚本或此脚本导致错误时使用此代码。
- **4000** 回退当前事务，取消同步并关闭 MobiLink 服务器。

注释

当 MobiLink 在上载事务期间处理插入、更新或删除脚本时或当 MobiLink 读取下载行时，如果发生 ODBC 驱动程序管理器标记的错误，则 MobiLink 服务器一遇到该错误就会执行此脚本。如果 ODBC 错误在其它时间发生，则 MobiLink 服务器调用 report_error 或 report_ODBC_error 脚本并中止同步。

错误代码可用于标识错误的性质。

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

操作代码将告知 MobiLink 服务器如何继续。在调用此脚本之前，MobiLink 服务器将操作代码设置为缺省值，该缺省值的大小取决于错误的严重程度。您可以使用脚本修改此值。脚本必须返回或设置一个操作代码。

handle_odbc_error 脚本在 handle_error 和 report_error 脚本之后、report_odbc_error 脚本之前调用。

如果只定义了一个（而非两个）error-handling 脚本，则该脚本的返回值将决定错误行为。如果定义了两个 error-handling 脚本，则 MobiLink 服务器将使用数值最大的操作代码。如果同时定义了 handle_error 和 handle_ODBC_error，则 MobiLink 将使用所有的调用所返回的最大数值的操作代码。

另请参见

- “脚本参数”一节第 299 页
- “添加和删除脚本”一节第 306 页
- “handle_error 连接事件”一节第 414 页
- “report_error 连接事件”一节第 444 页
- “report_odbc_error 连接事件”一节第 447 页
- “在脚本中使用远程 ID 和 MobiLink 用户名”一节 《MobiLink - 客户端管理》

SQL 示例

以下示例在 SQL Anywhere 统一数据库中运行。它允许应用程序忽略 ODBC 完整性约束违规。

以下对 MobiLink 系统过程的调用将 HandleODBCError 存储过程指派给 handle_odbc_error 事件。

```
CALL ml_add_connection_script(
 'ver1',
 'handle_odbc_error',
 'CALL HandleODBCError(
   {ml s.action_code},
   {ml s.ODBC_state},
   {ml s.error_message},
   {ml s.username},
   {ml s.table} )' )
```

以下 SQL 语句创建 HandleODBCError 存储过程。

```
CREATE PROCEDURE HandleODBCError(
   INOUT action integer,
   IN odbc_state varchar(5),
   IN error_message varchar(1000),
   IN user_name varchar(128),
   IN table_name varchar(128) )
  BEGIN
   IF odbc_state = '23000' then
      -- Ignore the error and keep going.
      SET action = 1000;
   ELSE
      -- Abort the synchronization.
      SET action = 3000;
   END IF;
END
```

Java 示例

以下对 MobiLink 系统过程的调用在同步脚本版本 ver1 时将名为 handleODBCError 的 Java 方法注册为 handle_odbc_error 事件的脚本。

```
CALL ml_add_java_connection_script(
 'ver1',
 'handle_odbc_error',
 'ExamplePackage.ExampleClass.handleODBCError'
)
```

以下是 Java 方法 handleODBCError 示例。它根据传入的数据处理错误。同时还确定结果错误代码。

```
public String handleODBCError(
  ianywhere.ml.script.InOutInteger actionCode,
  String ODBCState,
  String errorMessage,
  String user,
  String table ) {
  int newAC;
  if( user == null ) {
    newAC = handleNonSyncError( ODBCState,
  errorMessage );
  }
  else if( table == null ) {
    newAC = handleConnectionError( ODBCState,
     errorMessage, user );
  } else {
    newAC = handleTableError( ODBCState,
     errorMessage, user, table );
  }
  // Keep the most serious action code.
  if( actionCode.getValue() < newAC ) {
    actionCode.setValue( newAC );
  }
  return( null );
}
```

.NET 示例

以下对 MobiLink 系统过程的调用在同步脚本版本 ver1 时将名为 HandleODBCError 的 .NET 方法注册为 handle_odbc_event 的脚本。

```
CALL ml_add_dnet_connection_script(
 'ver1',
 'handle_odbc_error',
 'TestScripts.Test.HandleODBCError' )
```

以下是 .NET 方法 HandleODBCError 示例。

```
public string HandleODBCError (
  ref int actionCode,
  string ODBCState,
  string errorMessage,
  string user,
  string table ) {
  int new_ac;
  if( user == null ) {
    new_ac = HandleNonSyncError( ODBCState,
      errorMessage );
  }
  else if( table == null ) {
    new_ac = HandleConnectionError( ODBCState,
     errorMessage, user );
  } else {
    new_ac = HandleTableError( ODBCState,
     errorMessage, user, table );
  }
```

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

```
    // Keep the most serious action code.
    if( actionCode < new_ac ) {
      actionCode = new_ac;
    }
    return( null );
  }
```

handle_UploadData 连接事件

由直接行处理用来处理上载的行。

参数

SQL 脚本的参数名称	说明	顺序
UploadData	封装 MobiLink 客户端上载的表操作的 .NET 或 Java 类。该类在面向 Java 和 .NET 的 MobiLink 服务器 API 中定义。	1

缺省操作

无。

注释

handle_UploadData 事件允许您为 MobiLink 直接行处理处理上载。在同步中此事件对每个上载事务触发一次，除非您正在使用事务级上载，在这种情况下，它对每个事务进行触发。

请参见“直接行处理”第 609 页。

此事件使用单个 UploadData 参数。您的 Java 或 .NET 方法可以使用 UploadData getUploadedTables 或 getUploadedTableByName 方法获得 UploadedTableData 实例。UploadedTableData 允许您在当前同步中访问由 MobiLink 客户端上载的插入、更新和删除操作。

有关 UploadData 和 UploadedTableData 类的详细信息，请参见“处理直接上载”一节第 614 页。

如果您要读取列名元数据，则您必须指定 SendColumnNames MobiLink 客户端扩展选项或属性。否则，您可以按索引引用列，如在远程数据库中定义的那样。

请参见“SendColumnNames (scn) 扩展选项”一节 《MobiLink - 客户端管理》和“Send Column Names 同步参数”一节 《UltraLite - 数据库管理和参考》。

注意
此事件不能作为 SQL 实现。

另请参见

- “直接行处理”第 609 页
- “处理直接上载”一节第 614 页
- Java： “UploadData 接口”一节第 543 页
- Java： “UploadedTableData 接口”一节第 545 页
- .NET： “UploadData 接口”一节第 602 页
- .NET： “UploadedTableData 接口”一节第 604 页
- dbmlsync： “SendColumnNames (scn) 扩展选项”一节 《MobiLink - 客户端管理》
- UltraLite： “Send Column Names 同步参数”一节 《UltraLite - 数据库管理和参考》
- “handle_DownloadData 连接事件”一节第 410 页
- “必需的脚本”一节第 305 页
- “添加和删除脚本”一节第 306 页

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

Java 示例

以下对 MobiLink 系统过程的调用在同步脚本版本 ver1 时，为 handle_UploadData 连接事件注册名为 handleUpload 的 Java 方法。针对您的 MobiLink 统一数据库运行此系统过程。

```
CALL ml_add_java_connection_script(
  'ver1',
  'handle_UploadData',
  'MyPackage.MyClass.handleUpload' )
```

有关 ml_add_java_connection_script 的详细信息，请参见“ml_add_java_connection_script 系统过程”一节第 630 页。

以下 Java 方法处理 remoteOrders 表的上载。UploadData.getUploadedTableByName 方法为 remoteOrders 表返回一个 UploadedTableData 实例。UploadedTableData getInserts 方法返回一个表示新行的 java.sql.ResultSet 实例。

```
import ianywhere.ml.script.*;
import java.sql.*;
import java.io.*;
// ...

public void handleUpload( UploadData ut )
  throws SQLException, IOException {
  // Get an UploadedTableData instance representing the
  // remoteOrders table.
  UploadedTableData remoteOrdersTable =
ut.getUploadedTableByName("remoteOrders");
  // Get inserts uploaded by the MobiLink client.
  java.sql.ResultSet results = remoteOrdersTable.getInserts();
  while( results.next() ) {
    // You can reference column names here because SendColumnNames is on
    // Get the primary key.
    int pk = results.getInt("pk");

    // Get the uploaded num_ordered value.
    int numOrdered = results.getInt("num_ordered");

    // The current insert row is now ready to be uploaded to wherever
    // you want it to go (a file, a web service, and so on).

  }

  results.close();

}
```

以下示例输出由 MobiLink 远程数据库上载的插入、更新和删除操作。UploadData getUploadedTables 方法返回表示由远程数据库上载的所有表的 UploadedTableData 实例。此数组中表的顺序就是远程数据库对其进行上载的顺序。UploadedTableData getInserts、getUpdates 和 getDeletes 方法返回标准 JDBC 结果集。可以使用 println 方法或将数据输出到文本文件或其它位置。

```
import ianywhere.ml.script.*;
import java.sql.*;
import java.io.*;
// ...

public void handleUpload( UploadData ud )
  throws SQLException, IOException {
  UploadedTableData tables[] = ud.getUploadedTables();
  for( int i = 0; i < tables.length; i++ ) {
```

```
      UploadedTableData currentTable = tables[i];
      println( "table " + java.lang.Integer.toString( i ) +
       " name: " + currentTable.getName() );
      // Print out delete result set.
      println( "Deletes" );
      printRSInfo( currentTable.getDeletes() );
      // Print out insert result set.
      println( "Inserts" );
      printRSInfo( currentTable.getInserts() );
      // print out update result set
      println( "Updates" );
      printUpdateRSInfo( currentTable.getUpdates() );
    }
  }
```

printRSInfo 方法输出插入、更新或删除结果集并接受单个 java.sql.ResultSet 对象。详细列信息（包括列标签）由 ResultSet getMetaData 方法返回的 ResultSetMetaData 对象提供。列标签仅在客户端启用了 SendColumnNames 选项时可用。printRow 方法输出结果集中的每一行。

```
public void printRSInfo( ResultSet results )
  throws SQLException, IOException {

  // Obtain the result set metadata.
  ResultSetMetaData metaData = results.getMetaData();
  String columnHeading = "";

  // Print out column headings.
  for( int c = 1; c <= metaData.getColumnCount(); c++ ) {
    columnHeading += metaData.getColumnLabel(c);
    if( c < metaData.getColumnCount() ) {
      columnHeading += ", ";
    }
  }

  println( columnHeading );
  while( results.next() ) {

    // Print out each row.
    printRow( results, metaData.getColumnCount() );
  }

  // Close the java.sql.ResultSet.
  results.close();
}
```

如下所示，printRow 方法使用 ResultSet getString 方法获得每个列值。

```
public void printRow( ResultSet results, int colCount )
  throws SQLException, IOException {
  String row = "( ";

  for( int c = 1; c <= colCount; c++ ) {
    // Get a column value.
    String currentColumn = results.getString( c );

    // Check for null values.
    if( currentColumn == null ) {
      currentColumn = "<NULL>";
    }

    // Add the column value to the row string.
    row += cur_col;
    if( c < colCount ) {
```

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

```
        row += ", ";
      }
    }

    row += " )";

    // Print out the row.
    println( row );
  }
```

.NET 示例

以下对 MobiLink 系统过程的调用在同步脚本版本 ver1 时，为 handle_UploadData 连接事件注册名为 HandleUpload 的 .NET 方法。针对您的 MobiLink 统一数据库运行此系统过程。

```
CALL ml_add_dnet_connection_script(
  'ver1',
  'handle_UploadData',
  'TestScripts.Test.HandleUpload' )
```

以下 .NET 方法处理 remoteOrders 表的上载。

```
using System;
using System.Data;
using System.IO;
using iAnywhere.MobiLink.Script;
using iAnywhere.MobiLink;

namespace MyScripts
{
    public class MyUpload
    {
 public MyUpload( DBConnectionContext cc )
 {
 }

 ~MyUpload()
 {
 }

 public void handleUpload( UploadData ut )
 {
     int i;
     UploadedTableData[] tables = ut.GetUploadedTables();

     for( i=0; i<tables.Length; i+=1 ) {
  UploadedTableData cur_table = tables[i];
  Console.Write( "table " + i + " name: " + cur_table.GetName() );
  // Print out delete result set.
  Console.Write( "Deletes" );
  printRSInfo( cur_table.GetDeletes() );

  // Print out insert result set.
  Console.Write( "Inserts" );
  printRSInfo( cur_table.GetInserts() );

  // print out update result set
  Console.Write( "Updates" );
  printUpdateRSInfo( cur_table.GetUpdates() );
     }
 }

 public void printRSInfo( IDataReader dr )
```

```
{
    // Obtain the result set metadata.
    DataTable dt = dr.GetSchemaTable();
    DataColumnCollection cc = dt.Columns;
    DataColumn dc;
    String columnHeading = "";

    // Print out column headings.
    for( int c=0; c < cc.Count; c = c + 1 ) {
 dc = cc[ c ];
 columnHeading += dc.ColumnName;
 if( c < cc.Count - 1 ) {
     columnHeading += ", ";
 }
    }
    Console.Write( columnHeading );

    while( dr.Read() ) {
 // Print out each row.
 printRow( dr, cc.Count );
    }

    // Close the java.sql.ResultSet.
    dr.Close();
}

public void printUpdateRSInfo( UpdateDataReader utr )
{
    // Obtain the result set metadata.
    DataTable dt = utr.GetSchemaTable();
    DataColumnCollection cc = dt.Columns;
    DataColumn dc;
    String columnHeading = "TYPE, ";

    // Print out column headings.
    for( int c = 0; c < cc.Count; c = c + 1 ) {
 dc = cc[ c ];
 columnHeading += dc.ColumnName;
 if( c < cc.Count - 1 ) {
     columnHeading += ", ";
 }
    }
    Console.Write( columnHeading );

    while( utr.Read() ) {
 // Print out the new values for the row.
 utr.SetNewRowValues();
 Console.Write( "NEW:" );
 printRow( utr, cc.Count );

 // Print out the old values for the row.
 utr.SetOldRowValues();
 Console.Write( "OLD:" );
 printRow( utr, cc.Count );
    }

    // Close the java.sql.ResultSet.
    utr.Close();
}

public void printRow( IDataReader dr, int col_count )
{
    String row = "( ";
    int c;
```

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

```
            for( c = 0; c < col_count; c = c + 1 ) {
          // Get a column value.
          String cur_col = dr.GetString( c );

          // Check for null values.
          if( cur_col == null ) {
              cur_col = "<NULL>";
          }

          // Add the column value to the row string.
          row += cur_col;
          if( c < col_count ) {
              row += ", ";
          }
             }

             row += " )";

             // Print out the row.
             Console.Write( row );
         }
            }
        }
```

modify_error_message 连接事件

该脚本可用于自定义发送给远程数据库的消息文本（错误、警告和信息）。

参数

在下表中，说明部分提供 SQL 数据类型。如果您使用 Java 或 .NET 编写脚本，则应该使用相应的数据类型。请参见“SQL-Java 数据类型”一节第 498 页和“SQL-.NET 数据类型”一节第 556 页。

在 SQL 脚本中，可以使用名称或问号指定事件参数，但不能在一个脚本中混合使用名称和问号。如果使用问号，则参数必须按照如下所示的顺序并且仅当没有指定任何后继参数时才是可选的（例如，如果您想使用参数 2，则必须使用参数 1）。如果使用命名参数，则可以按照任何顺序指定任意参数子集。

SQL 脚本的参数名称	说明	顺序
s.error_message	VARBINARY(1024)。这是一个 INOUT 参数。	1
s.remote_id	VARCHAR(128)。MobiLink 远程 ID。只有在使用命名参数时才能引用远程 ID。	不适用
s.username	VARCHAR(128)。MobiLink 用户名。	2
s.error_code	INT。	3

缺省操作

无。

注释

modify_error_message 事件的 SQL 脚本必须作为存储过程实现。

另请参见

- “脚本参数”一节第 299 页
- “添加和删除脚本”一节第 306 页
- “在脚本中使用远程 ID 和 MobiLink 用户名”一节 《MobiLink - 客户端管理》

SQL 示例

下面的示例下载前一天的所有内容，不管那之后数据库是否进行过同步。

以下 SQL 语句创建 ModifyLastErrorMessage 存储过程：

```
CREATE PROCEDURE ModifyLastErrorMessage(
  inout error_message VARBINARY(1024),
  in username VARCHAR(128),
  in error_code INT )
BEGIN
```

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

```
  SELECT dateadd(day, -1, last_download_time )
  INTO last_download_time
END
```

以下对 MobiLink 系统过程的调用为脚本版本 modify_ts_test 将 ModifyLastErrorMessage 指派给 modify_error_message 连接事件：

```
CALL ml_add_connection_script(
  'modify_ts_test',
  'modify_error_message',
  'CALL ModifyLastErrorMessage (
    {ml s.error_message},
    {ml s.username},
    {ml s.error_code} )' );
```

Java 示例

以下对 MobiLink 系统过程的调用在同步脚本版本 ver1 时将名为 modifyLastErrorMessage 的 Java 方法注册为 modify_error_message 连接事件的脚本。

```
CALL ml_add_java_connection_script(
  'ver1',
  'modify_error_message',
  'ExamplePackage.ExampleClass.modifyLastErrorMessage' )
```

以下是 Java 方法 modifyLastErrorMessage 示例。它输出当前错误消息和错误代码。

```
public String modifyLastErrorMessage(
  String lastErrorMessage,
  String userName,
  int errorCode ) {
  java.lang.System.out.println( "error message: " +
   lastErrorMessage );
  java.lang.System.out.println( "error code: " +
   String.valueOf(errorCode) );
  return( null );
}
```

.NET 示例

以下对 MobiLink 系统过程的调用在同步脚本版本 ver1 时将名为 ModifyLastErrorMessage 的 .NET 方法注册为 modify_error_message 连接事件的脚本。

```
CALL ml_add_dnet_connection_script(
  'ver1',
  'modify_error_message',
  'TestScripts.Test.ModifyLastErrorMessage' )
```

以下是 .NET 方法 ModifyLastErrorMessage 示例。它输出当前错误代码和错误消息。

```
public string ModifyLastErrorMessage (
  string errorMessage,
  string userName,
  string errorCode  ) {
  System.Console.WriteLine( "error message: " + errorMessage );
  System.Console.WriteLine( "error code: " +  errorCode );
  return ( null );
}
```

modify_last_download_timestamp 连接事件

该脚本可用于修改当前同步的上次下载时间。

参数

在下表中，说明部分提供 SQL 数据类型。如果您使用 Java 或 .NET 编写脚本，则应该使用相应的数据类型。请参见“SQL-Java 数据类型”一节第 498 页和“SQL-.NET 数据类型”一节第 556 页。

在 SQL 脚本中，可以使用名称或问号指定事件参数，但不能在一个脚本中混合使用名称和问号。如果使用问号，则参数必须按照如下所示的顺序并且仅当没有指定任何后继参数时才是可选的（例如，如果您想使用参数 2，则必须使用参数 1）。如果使用命名参数，则可以按照任何顺序指定任意参数子集。

SQL 脚本的参数名称	说明	顺序
s.last_download	TIMESTAMP。任何已同步表的上一次下载时间。这是一个 INOUT 参数。	1
s.remote_id	VARCHAR(128)。MobiLink 远程 ID。只有在使用命名参数时才能引用远程 ID。	不适用
s.username	VARCHAR(128)。MobiLink 用户名。	2

缺省操作

无。

注释

该脚本可用于修改当前同步的 last_download 时间戳。如果定义了此脚本，则 MobiLink 服务器将使用修改过的 last_download 时间戳作为传递到下载脚本的 last_download 时间戳。此脚本的一种典型应用是恢复远程数据库上的丢失数据。您可以将 last_download 时间戳重置为早期的时间（例如 1900-01-01 00:00），以使下一个同步下载所有数据。

modify_last_download_timestamp 事件的 SQL 脚本必须作为存储过程实现。MobiLink 服务器传入 last_download 时间戳作为传递给存储过程的第一个参数，并用该存储过程传出的第一个值替换该时间戳。

该脚本紧接 prepare_for_download 脚本之前在同一事务中执行。

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

另请参见

- “脚本参数”一节第 299 页
- “添加和删除脚本”一节第 306 页
- “在脚本中使用远程 ID 和 MobiLink 用户名”一节 《MobiLink - 客户端管理》
- “在脚本中使用上次下载时间”一节第 122 页
- “如何生成和使用下载时间戳”一节第 122 页
- “modify_next_last_download_timestamp 连接事件”一节第 433 页

SQL 示例

以下 SQL 语句创建一个存储过程。以下语法用于 Oracle 统一数据库：

```
CREATE PROCEDURE ModifyDownloadTimestamp(
        download_timestamp  OUT DATE,
        user_name    IN VARCHAR )
        AS
        BEGIN
         -- N is the maximum replication latency in the consolidated cluster
        download_timestamp := download_timestamp - N;
        END;
```

以下语法用于 SQL Anywhere、Adaptive Server Enterprise 和 SQL Server 统一数据库：

```
CREATE PROCEDURE ModifyDownloadTimestamp
        @download_timestamp DATETIME OUTPUT,
        @t_name     VARCHAR( 128 )
        AS
        BEGIN
         -- N is the maximum replication latency in consolidated cluster
        SELECT @download_timestamp = @download_timestamp - N
        END
```

以下语法用于 DB2 主机统一数据库：

```
CREATE PROCEDURE modify_ldts(
        OUT ts       TIMESTAMP,
        IN  t_name   VARCHAR(128) )
        LANGUAGE SQL
        BEGIN
            set ts = TIMESTAMP_FORMAT('2000-01-02 03:04:05','YYYY-MM-DD
HH24:MI:SS');
        END
```

以下对 MobiLink 系统过程的调用将 ModifyDownloadTimestamp 存储过程指派给 modify_last_download_timestamp 事件。以下语法用于 SQL Anywhere 统一数据库：

```
CALL ml_add_connection_script(
  'my_version',
  'modify_last_download_timestamp',
  '{CALL ModifyDownloadTimestamp(
   {ml s.last_download},
   {ml s.username} ) }' )
```

Java 示例

以下对 MobiLink 系统过程的调用在同步脚本版本 ver1 时将名为 modifyLastDownloadTimestamp 的 Java 方法注册为 modify_last_download_timestamp 连接事件的脚本。

```
CALL ml_add_java_connection_script(
  'ver1',
```

```
  'modify_last_download_timestamp',
  'ExamplePackage.ExampleClass.modifyLastDownloadTimestamp' )
```

以下是 Java 方法 modifyLastDownloadTimestamp 示例。它输出当前新的时间戳并修改传入的时间戳。

```
public String modifyLastDownloadTimestamp(
     Timestamp lastDownloadTime,
     String userName ) {
   java.lang.System.out.println( "old date: " +
    lastDownloadTime.toString() );
   lastDownloadTime.setDate(
    lastDownloadTime.getDate() -1 );
   java.lang.System.out.println( "new date: " +
    lastDownloadTime.toString() );
   return( null );
}
```

.NET 示例

以下对 MobiLink 系统过程的调用在同步脚本版本 ver1 时将名为 ModifyLastDownloadTimestamp 的 .NET 方法注册为 modify_last_download_timestamp 连接事件的脚本。

```
CALL ml_add_dnet_connection_script(
  'ver1',
  'modify_last_download_timestamp',
  'TestScripts.Test.ModifyLastDownloadTimestamp' )
```

以下是 .NET 方法 ModifyLastDownloadTimestamp 示例。

```
public string ModifyLastDownloadTimestamp(
  DateTime lastDownloadTime,
  string userName ) {
  System.Console.WriteLine( "old date: " +
   last_download_time.ToString() );
  last_download_time = DateTime::Now;
  System.Console.WriteLine( "new date: " +
   last_download_time.ToString() );
  return( null );
}
```

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

modify_next_last_download_timestamp 连接事件

该脚本可用于修改下一个同步的上次下载时间。

参数

在下表中，说明部分提供 SQL 数据类型。如果您使用 Java 或 .NET 编写脚本，则应该使用相应的数据类型。请参见“SQL-Java 数据类型”一节第 498 页和“SQL-.NET 数据类型”一节第 556 页。

在 SQL 脚本中，可以使用名称或问号指定事件参数，但不能在一个脚本中混合使用名称和问号。如果使用问号，则参数必须按照如下所示的顺序并且仅当没有指定任何后继参数时才是可选的（例如，如果您想使用参数 2，则必须使用参数 1）。如果使用命名参数，则可以按照任何顺序指定任意参数子集。

SQL 脚本的参数名称	说明	顺序
s.next _last_download	TIMESTAMP。这是一个 INOUT 参数。在提交上载之后 MobiLink 服务器即生成此值。	1
s.last_download	TIMESTAMP。这是一个 IN 参数。这是当前同步的上次下载时间。通过确保您不将 next_last_download 时间戳修改为早于 last_download 时间戳，它在避免重复方面会非常有用。	2
s.remote_id	VARCHAR(128)。MobiLink 远程 ID。只有在使用命名参数时才能引用远程 ID。	不适用
s.username	VARCHAR(128)。MobiLink 用户名。	3

缺省操作

无。

注释

此脚本允许您更改 next_last_download 时间戳，这会有效地更改下一个同步的 last_download 时间戳。这样您就可以重置下一个同步而不影响当前同步。

modify_next_last_download_timestamp 事件的 SQL 脚本必须作为存储过程实现。MobiLink 服务器传入 next_last_download 时间戳作为传递给存储过程的第一个参数，并用该存储过程传出的第一个值替换该时间戳。

该脚本在下载用户表后于下载事务中执行。

另请参见

- “脚本参数”一节第 299 页
- “添加和删除脚本”一节第 306 页
- “在脚本中使用远程 ID 和 MobiLink 用户名”一节 《MobiLink - 客户端管理》
- “在脚本中使用上次下载时间”一节第 122 页
- “如何生成和使用下载时间戳”一节第 122 页
- “modify_last_download_timestamp 连接事件”一节第 430 页

SQL 示例

以下是此脚本的一个应用示例。创建一个存储过程。以下语法用于 SQL Anywhere 统一数据库：

```
CREATE PROCEDURE ModifyNextDownloadTimestamp(
  inout download_timestamp TIMESTAMP ,
  in last_download TIMESTAMP ,
  in user_name VARCHAR(128) )
  BEGIN
    SELECT dateadd(hour, -1, download_timestamp )
    INTO download_timestamp
END
```

将脚本安装到您的 SQL Anywhere 统一数据库中：

```
CALL ml_add_connection_script(
  'modify_ts_test',
  'modify_next_last_download_timestamp',
  'CALL  ModifyNextDownloadTimestamp (
   {ml_s.next_last_download},
   {ml_s.last_download},
   {ml_s.username} )' )
```

Java 示例

以下对 MobiLink 系统过程的调用在同步脚本版本 ver1 时将名为 modifyNextDownloadTimestamp 的 Java 方法注册为 modify_next_last_download_timestamp 连接事件的脚本。

```
CALL ml_add_java_connection_script(
   'ver1',
   'modify_next_last_download_timestamp',
   'ExamplePackage.ExampleClass.modifyNextDownloadTimestamp' )
```

以下是 Java 方法 modifyNextDownloadTimestamp 示例。它将下载时间戳向后设置一小时。

```
public String modifyNextDownloadTimestamp(
  Timestamp downloadTimestamp,
  Timestamp lastDownload,
  String userName ) {
  downloadTimestamp.setHours(
  downloadTimestamp.getHours() -1 );
  return( null );
}
```

.NET 示例

以下对 MobiLink 系统过程的调用在同步脚本版本 ver1 时将名为 ModifyNextDownloadTimestamp 的 .NET 方法注册为 modify_next_last_download_timestamp 连接事件的脚本。

```
CALL ml_add_dnet_connection_script(
   'ver1',
```

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

```
    'modify_next_last_download_timestamp',
    'TestScripts.Test.ModifyNextDownloadTimestamp' )
```

以下是 .NET 方法 ModifyNextDownloadTimestamp 示例。它将下载时间戳向后设置一小时。

```
public string ModifyNextDownloadTimestamp (
  DateTime download_timestamp,
  DateTime last_download,
  string user_name ) {
  download_timestamp = download_timestamp.AddHours( -1 );
  return ( null );
}
```

modify_user 连接事件

提供 MobiLink 用户名。

参数

在下表中，说明部分提供 SQL 数据类型。如果您使用 Java 或 .NET 编写脚本，则应该使用相应的数据类型。请参见“SQL-Java 数据类型”一节第 498 页和“SQL-.NET 数据类型”一节第 556 页。

SQL 脚本的参数名称	说明	顺序
s.remote_id	VARCHAR(128)。MobiLink 远程 ID。只有在使用命名参数时才能引用远程 ID。	不适用
s.username	VARCHAR(128)。MobiLink 用户名。这是一个 INOUT 参数。	1

缺省操作

无。

注释

MobiLink 服务器在调用脚本时以参数形式提供用户名，用户名由 MobiLink 客户端发送。在某些情况下，您需要使用另一个用户名。此脚本用于修改在调用 MobiLink 脚本过程中使用的用户名。

用户名参数必须足够长以保存用户名。

modify_user 事件的 SQL 脚本必须作为存储过程实现。

另请参见

- “脚本参数”一节第 299 页
- “添加和删除脚本”一节第 306 页
- “authenticate_user 连接事件”一节第 335 页
- “authenticate_user_hashed 连接事件”一节第 339 页
- “在脚本中使用远程 ID 和 MobiLink 用户名”一节 《MobiLink - 客户端管理》

SQL 示例

以下示例通过使用名为 user_device 的映射表将远程数据库用户名映射到使用该设备的用户的 ID。当同一个人拥有需要使用相同的同步逻辑（根据该用户的名称或 ID）的多个远程数据库（如 PDA 和膝上型计算机）时，可以采用此技术。

以下对 MobiLink 系统过程的调用将 ModifyUser 存储过程指派给 modify_user 事件。此语法用于 SQL Anywhere 统一数据库。

```
CALL ml_add_connection_script(
 'ver1',
 'modify_user',
 'call ModifyUser( {ml s.username} )' )
```

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

以下 SQL 语句创建 ModifyUser 存储过程。

```
CREATE PROCEDURE ModifyUser( INOUT u_name varchar(128) )
BEGIN
 SELECT user_name
    INTO u_name
    FROM user_device
    WHERE device_name = u_name;
END
```

Java 示例

以下对 MobiLink 系统过程的调用在同步脚本版本 ver1 时将名为 modifyUser 的 Java 方法注册为 modify_user 连接事件的脚本。

```
CALL ml_add_java_connection_script(
   'ver1',
   'modify_user',
   'ExamplePackage.ExampleClass.modifyUser' )
```

以下是 Java 方法 modifyUser 示例。它从数据库获取用户 ID，然后使用该 ID 设置用户名。

```
public String modifyUser(
  InOutString ioUserName )
  throws SQLException {
  Statement uidSelect = curConn.createStatement();
  ResultSet uidResult = uidSelect.executeQuery(
   "SELECT rep_id FROM SalesRep WHERE name = '" +
   ioUserName.getValue() + "' " );
  uidResult.next();
  ioUserName.setValue(
   java.lang.Integer.toString(uidResult.getInt( 1 ));
   uidResult.close();
   uidSelect.close();
  return ( null );
}
```

.NET 示例

以下对 MobiLink 系统过程的调用在同步脚本版本 ver1 时将名为 ModUser 的 .NET 方法注册为 modify_user 连接事件的脚本。

```
CALL ml_add_dnet_connection_script(
  'ver1',
  'modify_user',
  'TestScripts.Test.ModUser'
)
```

以下是 .NET 方法 ModUser 示例。

```
public string ModUser(
  string user ) {
  return ( "SELECT rep_id FROM SalesRep WHERE name = '" + user + "' " );
}
```

nonblocking_download_ack 连接事件

使用非阻塞下载确认时，此脚本会提供用来记录有关下载内容已被成功应用的信息的位置。

参数

在下表中，说明部分提供 SQL 数据类型。如果您使用 Java 或 .NET 编写脚本，则应该使用相应的数据类型。请参见“SQL-Java 数据类型”一节第 498 页和“SQL-.NET 数据类型”一节第 556 页。

在 SQL 脚本中，可以使用名称或问号指定事件参数，但不能在一个脚本中混合使用名称和问号。如果使用问号，则参数必须按照如下所示的顺序并且仅当没有指定任何后继参数时才是可选的（例如，如果您想使用参数 2，则必须使用参数 1）。如果使用命名参数，则可以按照任何顺序指定任意参数子集。

SQL 脚本的参数名称	说明	顺序
s.remote_id	VARCHAR(128)。MobiLink 远程 ID。只有在使用命名参数时才能引用远程 ID。	不适用
s.username	VARCHAR(128)。MobiLink 用户名。	2
s.last_download	TIMESTAMP。这是一个 IN 参数。这是当前同步的上次下载时间。通过确保您不将 next_last_download 时间戳修改为早于 last_download 时间戳，它在避免重复方面会非常有用。	3

注释

当在远程数据库中成功应用下载时，此事件允许您记录时间。

仅在使用非阻塞下载确认时才调用此事件。如果在处于非阻塞模式时发送下载，则将提交下载事务并结束同步。同步客户端确认成功下载时调用此事件。在原始同步的 end_synchronization 脚本之后，于新连接上调用此事件。此事件的操作与 MobiLink 系统表中下载时间的更新一起提交。

由于此脚本的特殊性质，执行此事件时，同步过程中设置的任何连接级变量均不可用。

另请参见

- “publication_nonblocking_download_ack 连接事件”一节第 442 页
- “-nba 选项”一节第 70 页
- dbmlsync： “SendDownloadACK (sa) 扩展选项”一节 《MobiLink - 客户端管理》
- UltraLite： “发送下载确认同步参数”一节 《UltraLite - 数据库管理和参考》

SQL 示例

以下脚本将记录添加到表 download_pubs_acked 中。该记录包含远程 ID、第一个验证参数和下载时间戳。

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

```
INSERT INTO download_pubs_acked( rem_id, auth_parm, last_download )
   VALUES( {ml s.remote_id},  {ml a.1}, {ml s.last_publication_download} )
```

prepare_for_download 连接事件

在上载和下载事务之间处理所需的操作。

参数

在下表中，说明部分提供 SQL 数据类型。如果您使用 Java 或 .NET 编写脚本，则应该使用相应的数据类型。请参见“SQL-Java 数据类型”一节第 498 页和“SQL-.NET 数据类型”一节第 556 页。

在 SQL 脚本中，可以使用名称或问号指定事件参数，但不能在一个脚本中混合使用名称和问号。如果使用问号，则参数必须按照如下所示的顺序并且仅当没有指定任何后继参数时才是可选的（例如，如果您想使用参数 2，则必须使用参数 1）。如果使用命名参数，则可以按照任何顺序指定任意参数子集。

SQL 脚本的参数名称	说明	顺序
s.last_download	TIMESTAMP。任何已同步表的上一次下载时间。	1
s.remote_id	VARCHAR(128)。MobiLink 远程 ID。只有在使用命名参数时才能引用远程 ID。	不适用
s.username	VARCHAR(128)。MobiLink 用户名。	2

缺省操作

无。

注释

MobiLink 服务器在上载事务和下载事务的开始之间以独立事务执行此脚本。

另请参见

- “脚本参数”一节第 299 页
- “添加和删除脚本”一节第 306 页
- “end_upload 连接事件”一节第 401 页
- “begin_download 连接事件”一节第 345 页
- “在脚本中使用远程 ID 和 MobiLink 用户名”一节 《MobiLink - 客户端管理》
- “在脚本中使用上次下载时间”一节第 122 页

SQL 示例

以下对 MobiLink 系统过程的调用在同步脚本版本 ver1 时将名为 prepareForDownload 的 SQL 方法注册为 prepare_for_download 事件的脚本。

```
CALL ml_add_connection_script(
   'ver1',
   'prepare_for_download',
   'CALL prepareForDownload(
       { ml s.current_time },
       { ml s.username } )' )
```

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

以下是示例 SQL 方法 prepareForDownload。它调用 SQL 方法修改数据库中的某些行。

```
CREATE PROCEDURE prepareForDownload (
  IN ts TIMESTAMP,
  IN user VARCHAR(128))
BEGIN
  CALL adjustUploadedRows(user)
END;
```

Java 示例

以下对 MobiLink 系统过程的调用在同步脚本版本 ver1 时将名为 prepareForDownload 的 Java 方法注册为 prepare_for_download 事件的脚本。

```
CALL ml_add_java_connection_script(
   'ver1',
   'prepare_for_download',
   'ExamplePackage.ExampleClass.prepareForDownload' )
```

以下是 Java 方法 prepareForDownload 示例。它调用 Java 方法修改数据库中的某些行。

```
public String prepareForDownload(
  Timestamp ts,
  String user ) {
  adjustUploadedRows( _syncConn, user );
  return( null );
}
```

.NET 示例

以下对 MobiLink 系统过程的调用在同步脚本版本 ver1 时将名为 PrepareForDownload 的 .NET 方法注册为 prepare_for_download 连接事件的脚本。

```
CALL ml_add_dnet_connection_script(
 'ver1',
 'prepare_for_download',
 'TestScripts.Test.PrepareForDownload'
)
```

以下是 .NET 方法 PrepareForDownload 示例。它调用 .NET 方法修改数据库中的某些行。

```
public string PrepareForDownload(
  DateTime timestamp,
  string user ) {
  AdjustUploadedRows ( _syncConn, user );
  return ( null );
}
```

publication_nonblocking_download_ack 连接事件

使用非阻塞下载确认时，此脚本会提供用于记录有关已成功下载发布的信息的位置。

参数

在下表中，说明部分提供 SQL 数据类型。如果您使用 Java 或 .NET 编写脚本，则应该使用相应的数据类型。请参见“SQL-Java 数据类型”一节第 498 页和“SQL-.NET 数据类型”一节第 556 页。

在 SQL 脚本中，可以使用名称或问号指定事件参数，但不能在一个脚本中混合使用名称和问号。如果使用问号，则参数必须按照如下所示的顺序并且仅当没有指定任何后继参数时才是可选的（例如，如果您想使用参数 2，则必须使用参数 1）。如果使用命名参数，则可以按照任何顺序指定任意参数子集。

SQL 脚本的参数名称	说明	顺序
s.remote_id	VARCHAR(128)。MobiLink 远程 ID。只有在使用命名参数时才能引用远程 ID。	不适用
s.username	VARCHAR(128)。MobiLink 用户名。	2
s.last_publication_download	TIMESTAMP。任何已同步表的上一次下载时间。	3
s.publication name	VARCHAR(128)。发布的名称。	4
s.subscription_id	VARCHAR(128)。发布 ID。	5

注释

当在远程数据库中成功应用此发布的下载时，此事件允许您记录时间。

仅在使用非阻塞下载确认时才调用此事件。如果在处于非阻塞模式时发送下载，则将提交下载事务并结束同步。当同步客户端确认下载成功时，下载中的每个发布都会调用该事件一次。在原始同步的 end_synchronization 脚本之后，于新连接上调用此事件。此事件的操作与 MobiLink 系统表中下载时间的更新一起提交。

由于此脚本的特殊性质，执行此事件时，同步过程中设置的任何连接级变量均不可用。

另请参见

- “nonblocking_download_ack 连接事件”一节第 438 页
- “-nba 选项”一节第 70 页
- dbmlsync： “SendDownloadACK (sa) 扩展选项”一节 《MobiLink - 客户端管理》
- UltraLite： “发送下载确认同步参数”一节 《UltraLite - 数据库管理和参考》

SQL 示例

以下脚本将记录添加到名为 download_pubs_acked 的表中。该记录包含发布名称、第一个验证参数和下载时间戳。

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

```
INSERT INTO download_pubs_acked( pub_name, auth_parm, last_download )
   VALUES( {ml s.publication_name},  {ml a.1}, {ml
s.last_publication_download} )
```

report_error 连接事件

可用于记录错误以及 handle_error 脚本选择的操作。

参数

在下表中，说明部分提供 SQL 数据类型。如果您使用 Java 或 .NET 编写脚本，则应该使用相应的数据类型。请参见“SQL-Java 数据类型”一节第 498 页和“SQL-.NET 数据类型”一节第 556 页。

在 SQL 脚本中，可以使用名称或问号指定事件参数，但不能在一个脚本中混合使用名称和问号。如果使用问号，则参数必须按照如下所示的顺序并且仅当没有指定任何后继参数时才是可选的（例如，如果您想使用参数 2，则必须使用参数 1）。如果使用命名参数，则可以按照任何顺序指定任意参数子集。

SQL 脚本的参数名称	说明	顺序
s.action_code	INTEGER。这是一个 INOUT 参数。此参数是强制性的。	1
s.error_code	INTEGER。	2
s.error_message	TEXT。	3
s.remote_id	VARCHAR(128)。MobiLink 远程 ID。只有在使用已命名的参数时才能引用远程 ID。	不适用
s.username	VARCHAR(128)。MobiLink 用户名。	4
s.table	VARCHAR(128)。	5

缺省操作

无。

注释

此脚本可用于记录错误以及 handle_error 脚本选择的操作。不管是否定义了 handle_error 脚本，此脚本都在 handle_error 事件之后执行。在与同步连接（管理/信息连接）不同的数据库连接上，它总是在其自己的事务中执行。

错误代码和错误消息可用于标识错误的性质。针对导致当前错误的 SQL 操作，对错误处理脚本的最后一次调用将返回操作代码值。

如果错误是作为同步的一部分发生的，则提供用户名。否则该值为空。

如果在操作某特定表时发生错误，则提供表名称。否则该值为空。表名是远程数据库中表的名称。此名称在统一数据库中是否存在直接对应的名称取决于同步系统的设计。

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

另请参见

- “脚本参数”一节第 299 页
- “添加和删除脚本”一节第 306 页
- “handle_error 连接事件”一节第 414 页
- “handle_odbc_error 连接事件”一节第 418 页
- “report_odbc_error 连接事件”一节第 447 页
- “在脚本中使用远程 ID 和 MobiLink 用户名”一节 《MobiLink - 客户端管理》

SQL 示例

以下示例在 SQL Anywhere 统一数据库中运行。它将行插入用于记录同步错误的表中。

```
CALL ml_add_connection_script(
 'ver1',
 'report_error',
 'INSERT INTO sync_error(
   action_code,
   error_code,
   error_message,
   user_name,
   table_name )
  VALUES (
   {ml s.action_code},
   {ml s.error_code},
   {ml s.error_message},
   {ml s.username},
   {ml s.table} )'  )
```

Java 示例

以下对 MobiLink 系统过程的调用在同步脚本版本 ver1 时将名为 reportError 的 Java 方法注册为 report_error 连接事件的脚本。

```
CALL ml_add_java_connection_script(
   'ver1',
   'report_error',
   'ExamplePackage.ExampleClass.reportError' )
```

以下是 Java 方法 reportError 示例。它使用 MobiLink 提供的 JDBC 连接将错误记录到表中。它还将设置操作代码。

```
public String reportError(
  ianywhere.ml.script.InOutInteger actionCode,
  int errorCode,
  String errorMessage,
  String user,
  String table )
  throws java.sql.SQLException {
   // Insert error information in a table,
   JDBCLogError( _syncConn, errorCode, errorMessage,
    user, table );
   actionCode.setValue( getActionCode( errorCode ) );
   return( null );
}
```

.NET 示例

以下对 MobiLink 系统过程的调用在同步脚本版本 ver1 时将名为 ReportError 的 .NET 方法注册为 report_error 连接事件的脚本。

```
CALL ml_add_dnet_connection_script(
   'ver1',
   'report_error,
   'TestScripts.Test.ReportError' )
```

以下是 .NET 方法 ReportError 示例。它使用 .NET 方法将错误记录到表中。

```
public string ReportError(
  ref int actionCode,
  int errorCode,
  string errorMessage,
  string user,
  string table ) {
  LogError(_syncConn, errorCode, errorMessage, user, table);
}
```

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

report_odbc_error 连接事件

可用于记录错误以及 handle_odbc_error 脚本选择的操作。

参数

在下表中，说明部分提供 SQL 数据类型。如果您使用 Java 或 .NET 编写脚本，则应该使用相应的数据类型。请参见“SQL-Java 数据类型”一节第 498 页和“SQL-.NET 数据类型”一节第 556 页。

在 SQL 脚本中，可以使用名称或问号指定事件参数，但不能在一个脚本中混合使用名称和问号。如果使用问号，则参数必须按照如下所示的顺序并且仅当没有指定任何后继参数时才是可选的（例如，如果您想使用参数 2，则必须使用参数 1）。如果使用命名参数，则可以按照任何顺序指定任意参数子集。

SQL 脚本的参数名称	说明	顺序
s.action_code	INTEGER。这是一个 INOUT 参数。此参数是强制性的。	1
s.ODBC_state	VARCHAR(5)。	2
s.error_message	TEXT。	3
s.remote_id	VARCHAR(128)。MobiLink 远程 ID。只有在使用命名参数时才能引用远程 ID。	不适用
s.username	VARCHAR(128)。MobiLink 用户名。	4
s.table	VARCHAR(128)。	5

缺省操作

无。

注释

此脚本可用于记录错误以及 handle_odbc_error 脚本选择的操作。不管是否定义了 handle_odbc_error 脚本，此脚本都在 handle_odbc_error 事件之后执行。在与同步连接（管理/信息连接）不同的数据库连接上，它总是在其自己的事务中执行。

错误代码和错误消息可用于标识错误的性质。针对导致当前错误的 SQL 操作，对错误处理脚本的最后一次调用将返回操作代码值。

如果错误是作为同步的一部分发生的，则提供用户名。否则该值为空。

如果在操作某特定表时发生错误，则提供表名称。否则该值为空。表名是远程数据库中表的名称。此名称在统一数据库中是否存在直接对应的名称取决于同步系统的设计。

另请参见

- “脚本参数”一节第 299 页
- “添加和删除脚本”一节第 306 页
- “handle_error 连接事件”一节第 414 页
- “handle_odbc_error 连接事件”一节第 418 页
- “report_error 连接事件”一节第 444 页
- “在脚本中使用远程 ID 和 MobiLink 用户名”一节 《MobiLink - 客户端管理》

SQL 示例

以下示例在 SQL Anywhere 统一数据库中运行。它将行插入用于记录同步错误的表中。

```
CALL ml_add_connection_script(
 'ver1',
 'report_odbc_error',
 'INSERT INTO sync_error(
    action_code,
    odbc_state,
    error_message,
    user_name,
    table_name )
  VALUES(
   {ml s.action_code},
   {ml s.ODBC_state},
   {ml s.error_message},
   {ml s.username},
   {ml s.table} )' )
```

Java 示例

以下对 MobiLink 系统过程的调用在同步脚本版本 ver1 时将名为 reportODBCError 的 Java 方法注册为 report_odbc_error 事件的脚本。

```
CALL ml_add_java_connection_script(
   'ver1',
   'report_odbc_error',
   'ExamplePackage.ExampleClass.reportODBCError' )
```

以下是 Java 方法 reportODBCError 示例。它使用 MobiLink 提供的 JDBC 连接将错误记录到表中。它还将设置操作代码。

```
public String reportODBCError(
  ianywhere.ml.script.InOutInteger actionCode,
  String ODBCState,
  String errorMessage,
  String user,
  String table )
  throws java.sql.SQLException {
  JDBCLogError( _syncConn, ODBCState, errorMessage,
   user, table );
  actionCode.setValue( getActionCode( ODBCState ) );
  return ( null );
}
```

.NET 示例

以下对 MobiLink 系统过程的调用在同步脚本版本 ver1 时将名为 ReportODBCError 的 .NET 方法注册为 report_odbc_error 事件的脚本。

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

```
CALL ml_add_dnet_connection_script(
   'ver1',
   'report_odbc_error',
   'TestScripts.Test.ReportODBCError' )
```

以下是 .NET 方法 ReportODBCError 示例。它使用 .NET 方法将错误记录到表中。

```
public string ReportODBCError (
  ref int actionCode,
  string ODBCState,
  string errorMessage,
  string user,
  string table ) {
  LogError(_syncConn, ODBCState, errorMessage, user, table);
  return ( null );
}
```

resolve_conflict 表事件

定义用于解决特定表中冲突的过程。

参数

在下表中，说明部分提供 SQL 数据类型。如果您使用 Java 或 .NET 编写脚本，则应该使用相应的数据类型。请参见“SQL-Java 数据类型”一节第 498 页和“SQL-.NET 数据类型”一节第 556 页。

在 SQL 脚本中，可以使用名称或问号指定事件参数，但不能在一个脚本中混合使用名称和问号。如果使用问号，则参数必须按照如下所示的顺序并且仅当没有指定任何后继参数时才是可选的（例如，如果您想使用参数 2，则必须使用参数 1）。如果使用命名参数，则可以按照任何顺序指定任意参数子集。

SQL 脚本的参数名称	说明	顺序
s.remote_id	VARCHAR(128)。MobiLink 远程 ID。只有在使用命名参数时才能引用远程 ID。	不适用
s.username	VARCHAR(128)。MobiLink 用户名。	1
s.table	VARCHAR(128)。表名。	2

缺省操作

无。

注释

在远程数据库上更新行时，MobiLink 客户端将保存原始值的副本。客户端将数据的旧值和新值一同发送到 MobiLink 服务器。

在 MobiLink 服务器接收到更新的行时，它会将原始值与统一数据库中的当前值进行比较。使用 upload_fetch 脚本执行比较。

如果旧的上载值与统一数据库中的当前值不匹配，则该行发生冲突。MobiLink 服务器会将旧值和新值同时插入到统一数据库中，而不是更新该行。分别使用 upload_old_row_insert 和 upload_new_row_insert 脚本处理旧行和新行。

一旦插入值，MobiLink 服务器将执行 resolve_conflict 脚本。它可用于解决冲突。您可以实现您选择的任何一种方案。

每发生一次冲突该脚本执行一次。

或者，您可以不定义 resolve_conflict 脚本，而是通过将冲突解决逻辑写入 end_upload_rows 脚本或 end_upload 表脚本以面向组的方式来解决冲突。

您可以为远程数据库的每个表编写一个 resolve_conflict 脚本。

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

另请参见

- “脚本参数”一节第 299 页
- “添加和删除脚本”一节第 306 页
- “upload_old_row_insert 表事件”一节第 476 页
- “upload_new_row_insert 表事件”一节第 473 页
- “upload_update 表事件”一节第 488 页
- “end_upload_rows 表事件”一节第 408 页
- “在脚本中使用远程 ID 和 MobiLink 用户名”一节 《MobiLink - 客户端管理》

SQL 示例

以下语句定义了适用于 Oracle 安装中 CustDB 示例应用程序的 resolve_conflict 脚本。它调用 ULResolveOrderConflict 存储过程。

```
exec ml_add_table_script(
   'custdb', 'ULOrder', 'resolve_conflict',
   'begin ULResolveOrderConflict();
end; ')
CREATE OR REPLACE PROCEDURE ULResolveOrderConflict()
AS
  new_order_id integer;
  new_status   varchar(20);
  new_notes    varchar(50);
BEGIN
  -- approval overrides denial
  SELECT order_id, status, notes
    INTO new_order_id, new_status, new_notes
    FROM ULNewOrder
   WHERE syncuser_id = SyncUserID;
  IF new_status = 'Approved' THEN
    UPDATE ULOrder o
       SET o.status = new_status, o.notes =
        new_notes
     WHERE o.order_id = new_order_id;
  END IF;
  DELETE FROM ULOldOrder;
  DELETE FROM ULNewOrder;
END;
```

Java 示例

以下对 MobiLink 系统过程的调用在同步脚本版本 ver1 时将名为 resolveConflict 的 Java 方法注册为 resolve_conflict 表事件的脚本。

```
CALL ml_add_java_table_script(
   'ver1',
   'table1',
   'resolve_conflict',
   'ExamplePackage.ExampleClass.resolveConflict' )
```

以下是 Java 方法 resolveConflict 示例。它调用的 Java 方法将使用 MobiLink 提供的 JDBC 连接来解决冲突。

```
public String resolveConflict(
  String user,
  String table) {
  resolveRows(_syncConn, user );
}
```

.NET 示例

以下对 MobiLink 系统过程的调用在同步脚本版本 ver1 时将名为 ResolveConflict 的 .NET 方法注册为 resolve_conflict 表事件的脚本。

```
CALL ml_add_dnet_table_script(
   'ver1',
   'table1',
   'resolve_conflict',
   'TestScripts.Test.ResolveConflict' )
```

以下是 .NET 方法 ResolveConflict 示例。它调用一个解决冲突的 .NET 方法。

```
public string ResolveConflict(
  String user,
  String table) {
  ResolveRows(_syncConn, user );
}
```

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

synchronization_statistics 连接事件

跟踪同步统计信息。

参数

在下表中，说明部分提供 SQL 数据类型。如果您使用 Java 或 .NET 编写脚本，则应该使用相应的数据类型。请参见“SQL-Java 数据类型”一节第 498 页和“SQL-.NET 数据类型”一节第 556 页。

在 SQL 脚本中，可以使用名称或问号指定事件参数，但不能在一个脚本中混合使用名称和问号。如果使用问号，则参数必须按照如下所示的顺序并且仅当没有指定任何后继参数时才是可选的（例如，如果您想使用参数 2，则必须使用参数 1）。如果使用命名参数，则可以按照任何顺序指定任意参数子集。

SQL 脚本的参数名称	说明	顺序
s.remote_id	VARCHAR(128)。MobiLink 远程 ID。只有在使用命名参数时才能引用远程 ID。	不适用
s.username	VARCHAR(128)。MobiLink 用户名。	1
s.warnings	INTEGER。同步过程中发出的警告的数目。	2
s.errors	INTEGER。同步过程中发生的错误数。	3
s.deadlocks	INTEGER。统一数据库中为同步已检测到的死锁的数目。	4
s.synchronized_tables	INTEGER。同步中涉及的客户端表数。	5
s.connection_retries	INTEGER。MobiLink 服务器重试连接统一数据库的次数。	6

缺省操作

无。

注释

synchronization_statistics 事件可用于为任何用户和连接收集有关当前同步的各种统计信息。synchronization_statistics 连接脚本在最终同步事务结束时执行提交操作之前被调用。

另请参见

- “脚本参数”一节第 299 页
- “添加和删除脚本”一节第 306 页
- “download_statistics 连接事件”一节第 376 页
- “download_statistics 表事件”一节第 379 页
- “upload_statistics 连接事件”一节第 479 页
- “upload_statistics 表事件”一节第 483 页
- “synchronization_statistics 表事件”一节第 456 页
- “time_statistics 连接事件”一节第 459 页
- “time_statistics 表事件”一节第 462 页
- “MobiLink 监控器”第 167 页
- “在脚本中使用远程 ID 和 MobiLink 用户名”一节 《MobiLink - 客户端管理》

SQL 示例

下面的示例在 sync_con_audit 表中插入同步统计信息。

```
CALL ml_add_connection_script(
  'ver1',
  'synchronization_statistics',
 'INSERT INTO sync_con_audit(
   ml_user,
   warnings,
   errors,
   deadlocks,
   synchronized_tables,
   connection_retries)
  VALUES (
   {ml s.username},
   {ml s.warnings},
   {ml s.errors},
   {ml s.deadlocks},
   {ml s.synchronized_tables},
   {ml s.connection_retries})' )
```

在审计表中插入统计信息之后，您将可以使用这些统计信息监控同步过程，并在条件允许时进行优化。

Java 示例

以下对 MobiLink 系统过程的调用在同步脚本版本 ver1 时将名为 synchronizationStatisticsConnection 的 Java 方法注册为 synchronization_statistics 连接事件的脚本。

```
CALL ml_add_java_connection_script(
  'ver1',
  'synchronization_statistics',
  'ExamplePackage.ExampleClass.synchronizationStatisticsConnection'
)
```

以下是 Java 方法 synchronizationStatisticsConnection 示例。它将某些统计信息记录到 MobiLink 消息日志。（请注意：将统计信息记录到 MobiLink 消息日志在开发时可能会有帮助，但会降低生产服务器的性能。）

```
public String synchronizationStatisticsConnection(
  String user,
  int warnings,
  int errors,
```

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

```
  int deadlocks,
  int synchronizedTables,
  int connectionRetries ) {
  java.lang.System.out.println(
   "synch statistics number of deadlocks: "
   + deadlocks ;
  return( null );
}
```

.NET 示例

以下对 MobiLink 系统过程的调用在同步脚本版本 ver1 时将名为 SyncStats 的 .NET 方法注册为 synchronization_statistics 连接事件的脚本。

```
CALL ml_add_dnet_connection_script(
  'ver1',
  'synchronization_statistics',
  'TestScripts.Test.SyncStats'
)
```

以下是 .NET 方法 SyncStats 示例。它将某些统计信息记录到 MobiLink 消息日志。（请注意：将统计信息记录到 MobiLink 消息日志在开发时可能会有帮助，但会降低生产服务器的性能。）

```
public string SyncStats(
  string user,
  int warnings,
  int errors,
  int deadLocks,
  int syncedTables,
  int connRetries ) {
  System.Console.WriteLine( "synch statistics
   number of deadlocks: " + deadlocks ;
  return( null );
}
```

synchronization_statistics 表事件

跟踪同步统计信息。

参数

在下表中，说明部分提供 SQL 数据类型。如果您使用 Java 或 .NET 编写脚本，则应该使用相应的数据类型。请参见“SQL-Java 数据类型”一节第 498 页和“SQL-.NET 数据类型”一节第 556 页。

在 SQL 脚本中，可以使用名称或问号指定事件参数，但不能在一个脚本中混合使用名称和问号。如果使用问号，则参数必须按照如下所示的顺序并且仅当没有指定任何后继参数时才是可选的（例如，如果您想使用参数 2，则必须使用参数 1）。如果使用命名参数，则可以按照任何顺序指定任意参数子集。

SQL 脚本的参数名称	说明	顺序
s.remote_id	VARCHAR(128)。MobiLink 远程 ID。只有在使用命名参数时才能引用远程 ID。	不适用
s.username	VARCHAR(128)。MobiLink 用户名。	1
s.table	VARCHAR(128)。表名。	2
s.warnings	INTEGER。同步过程中表发生的警告数。	3
s.errors	INTEGER。同步过程中与表相关的错误数。	4

缺省操作

无。

注释

synchronization_statistics 事件可用于为任何用户和表收集同步过程中发生的警告和错误的数目。synchronization_statistics 表脚本在最终同步事务结束时执行提交操作之前被调用。

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

另请参见

- “脚本参数”一节第 299 页
- “添加和删除脚本”一节第 306 页
- “download_statistics 连接事件”一节第 376 页
- “download_statistics 表事件”一节第 379 页
- “upload_statistics 连接事件”一节第 479 页
- “upload_statistics 表事件”一节第 483 页
- “synchronization_statistics 连接事件”一节第 453 页
- “time_statistics 连接事件”一节第 459 页
- “time_statistics 表事件”一节第 462 页
- “MobiLink 监控器”第 167 页
- “在脚本中使用远程 ID 和 MobiLink 用户名”一节 《MobiLink - 客户端管理》

SQL 示例

下面的示例在 sync_tab_audit 表中插入同步统计信息。

```
CALL ml_add_table_script(
  'ver1',
  'table1',
  'upload_insert',
  'INSERT INTO sync_tab_audit (
    ml_user,
    table,
    warnings,
    errors)
   VALUES (
    {ml s.username},
    {ml s.table},
    {ml s.warnings},
    {ml s.errors} ) ' )
```

在审计表中插入同步统计信息之后，您将可以使用这些统计信息监控同步过程并在条件允许时进行优化。

Java 示例

以下对 MobiLink 系统过程的调用在同步脚本版本 ver1 时将名为 synchronizationStatisticsTable 的 Java 方法注册为 synchronization_statistics 表事件的脚本。

```
CALL ml_add_java_table_script(
  'ver1',
  'table1',
  'synchronization_statistics',
  'ExamplePackage.ExampleClass.synchronizationStatisticsTable'
)
```

以下是 Java 方法 synchronizationStatisticsTable 示例。它将某些统计信息记录到 MobiLink 消息日志。（请注意：将统计信息记录到 MobiLink 消息日志在开发时可能会有帮助，但会降低生产服务器的性能。）

```
public String synchronizationStatisticsTable(
  String user,
  String table,
  int warnings,
  int errors ) {
  java.lang.System.out.println( "synch statistics for
```

```
    table: " + table + " errors: " + errors );
  return( null );
}
```

.NET 示例

以下对 MobiLink 系统过程的调用在同步脚本版本 ver1 和表 table1 时将名为 SyncTableStats 的 .NET 方法注册为 synchronization_statistics 表事件的脚本。

```
CALL ml_add_dnet_table_script(
  'ver1',
  'table1',
  'synchronization_statistics',
  'TestScripts.Test.SyncTableStats'
)
```

以下是 .NET 方法 SyncTableStats 示例。它将某些统计信息记录到 MobiLink 消息日志。（请注意：将统计信息记录到 MobiLink 消息日志在开发时可能会有帮助，但会降低生产服务器的性能。）

```
public string SyncTableStats(
  string user,
  string table,
  int warnings,
  int errors ) {
  System.Console.WriteLine( "synch statistics for
   table: " + table + " errors: " + errors );
  return( null );
}
```

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

time_statistics 连接事件

为用户和事件跟踪时间统计信息。

参数

在下表中，说明部分提供 SQL 数据类型。如果您使用 Java 或 .NET 编写脚本，则应该使用相应的数据类型。请参见“SQL-Java 数据类型”一节第 498 页和“SQL-.NET 数据类型”一节第 556 页。

在 SQL 脚本中，可以使用名称或问号指定事件参数，但不能在一个脚本中混合使用名称和问号。如果使用问号，则参数必须按照如下所示的顺序并且仅当没有指定任何后继参数时才是可选的（例如，如果您想使用参数 2，则必须使用参数 1）。如果使用命名参数，则可以按照任何顺序指定任意参数子集。

SQL 脚本的参数名称	说明	顺序
s.remote_id	VARCHAR(128)。MobiLink 远程 ID。只有在使用命名参数时才能引用远程 ID。	不适用
s.username	VARCHAR(128)。MobiLink 用户名。	1
s.event_name	VARCHAR(128)	2
s.num_of_calls	INTEGER。调用该脚本的次数。	3
s.minimum_time	INTEGER。以毫秒为单位。此同步过程中执行一个脚本花费的最短时间。	4
s.maximum_time	INTEGER。以毫秒为单位。此同步过程中执行一个脚本花费的最长时间。	5
s.total_time	INTEGER。以毫秒为单位。同步过程中执行所有脚本花费的总时间。（这与同步的时间长度不同。）	6

缺省操作

无。

注释

time_statistics 事件可用于在同步过程中为任何用户收集时间统计信息。并且仅为有相应脚本的事件收集统计信息。该脚本在某一事件发生多次时收集汇总数据。它在进行用户、事件、表之间的时间比较时非常有用。

另请参见

- “脚本参数”一节第 299 页
- “添加和删除脚本”一节第 306 页
- “time_statistics 表事件”一节第 462 页
- “download_statistics 连接事件”一节第 376 页
- “download_statistics 表事件”一节第 379 页
- “upload_statistics 连接事件”一节第 479 页
- “upload_statistics 表事件”一节第 483 页
- “synchronization_statistics 连接事件”一节第 453 页
- “synchronization_statistics 表事件”一节第 456 页
- “MobiLink 监控器”第 167 页
- “在脚本中使用远程 ID 和 MobiLink 用户名”一节 《MobiLink - 客户端管理》

SQL 示例

以下示例将统计信息插入到 time_statistics 表中。

```
CALL ml_add_connection_script(
  'ver1',
  'time_statistics',
  'INSERT INTO time_statistics (
    id,
    ml_user,
    table,
    event_name,
    number_of_calls,
    minimum_time,
    maximum_time,
    total_time)
   VALUES (
    ts_id.nextval,
    {ml s.username},
    {ml s.event_name},
    {ml s.number_of_calls},
    {ml s.minimum_time},
    {ml s.maximum_time},
    {ml s.total_time} ) ' )
```

Java 示例

以下对 MobiLink 系统过程的调用在同步脚本版本 ver1 时将名为 timeStatisticsConnection 的 Java 方法注册为 time_statistics 连接事件的脚本。

```
CALL ml_add_java_connection_script(
  'ver1',
  'time_statistics',
  'ExamplePackage.ExampleClass.timeStatisticsConnection' )
```

以下是 Java 方法 timeStatisticsConnection 示例。它输出 prepare_for_download 事件的统计信息。（请注意：将统计信息输出到 MobiLink 消息日志在开发时可能会有帮助，但会降低生产服务器的性能。）

```
public String timeStatisticsConnection(
  String username,
  String tableName,
  String eventName,
  int numberOfCalls,
```

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

```
  int minimumTime,
  int maximumTime,
  int totalTime ) {
  if( eventName.equals( "prepare_for_download") ) {
    java.lang.System.out.println(
     "prepare_for_download num_calls: " + numCalls +
     "total_time: " + totalTime );
  }
  return ( null );
}
```

.NET 示例

以下对 MobiLink 系统过程的调用在同步脚本版本 ver1 时将名为 TimeStats 的 .NET 方法注册为 time_statistics 连接事件的脚本。

```
CALL ml_add_dnet_connection_script(
  'ver1',
  'time_statistics',
  'TestScripts.Test.TimeStats'
)
```

以下是 .NET 方法 TimeStats 示例。它输出 prepare_for_download 事件的统计信息。（请注意：将统计信息输出到 MobiLink 消息日志在开发时可能会有帮助，但会降低生产服务器的性能。）

```
public string TimeStats(
  string user,
  string eventName,
  int numberOfCalls,
  int minimumTime,
  int maximumTime,
  int totTime ) {
  if( event_name=="prepare_for_download") {
    System.Console.WriteLine(
     "prepare_for_download num_calls: " + num_calls +
     "total_time: " + total_time );
  }
  return ( null );
}
```

time_statistics 表事件

跟踪时间统计信息。

参数

在下表中，说明部分提供 SQL 数据类型。如果您使用 Java 或 .NET 编写脚本，则应该使用相应的数据类型。请参见“SQL-Java 数据类型”一节第 498 页和“SQL-.NET 数据类型”一节第 556 页。

在 SQL 脚本中，可以使用名称或问号指定事件参数，但不能在一个脚本中混合使用名称和问号。如果使用问号，则参数必须按照如下所示的顺序并且仅当没有指定任何后继参数时才是可选的（例如，如果您想使用参数 2，则必须使用参数 1）。如果使用命名参数，则可以按照任何顺序指定任意参数子集。

SQL 脚本的参数名称	说明	顺序
s.remote_id	VARCHAR(128)。MobiLink 远程 ID。只有在使用命名参数时才能引用远程 ID。	不适用
s.username	VARCHAR(128)。MobiLink 用户名。	1
s.table	VARCHAR(128)。表名。	2
s.event_name	VARCHAR(128)	3
s.number_of_calls	INTEGER。调用该脚本的次数。	4
s.minimum_time	INTEGER。以毫秒为单位。此表的同步过程中执行一个脚本花费的最短时间。	5
s.maximum_time	INTEGER。以毫秒为单位。此表的同步过程中执行一个脚本花费的最长时间。	6
s.total_time	INTEGER。以毫秒为单位。此表的同步过程中执行所有脚本花费的总时间。（这与同步的时间长度不同。）	7

缺省操作

无。

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

注释

time_statistics 表事件可用于在同步过程中为任何用户和表收集时间统计信息。并且仅为有相应脚本的事件收集统计信息。该脚本在某一事件发生多次时收集汇总数据。它在进行用户、事件、表之间的时间比较时非常有用。

另请参见

- “脚本参数”一节第 299 页
- “添加和删除脚本”一节第 306 页
- “time_statistics 连接事件”一节第 459 页
- “download_statistics 连接事件”一节第 376 页
- “download_statistics 表事件”一节第 379 页
- “upload_statistics 连接事件”一节第 479 页
- “upload_statistics 表事件”一节第 483 页
- “synchronization_statistics 连接事件”一节第 453 页
- “synchronization_statistics 表事件”一节第 456 页
- “MobiLink 监控器”第 167 页
- “在脚本中使用远程 ID 和 MobiLink 用户名”一节 《MobiLink - 客户端管理》

SQL 示例

以下示例将统计信息插入到 time_statistics 表中。

```
CALL ml_add_table_script (
 'ver1',
 'table1',
 'time_statistics',
 'INSERT INTO time_statistics(
   ml_user,
   table,
   event_name,
   number_of_calls,
   minimum_time,
   maximum_time,
   total_time)
  VALUES (
   {ml s.username},
   {ml s.table},
   {ml s.event_name},
   {ml s.number_of_calls},
   {ml s.minimum_time},
   {ml s.maximum_time},
   {ml s.total_time} )' );
```

Java 示例

以下对 MobiLink 系统过程的调用在同步脚本版本 ver1 时将名为 timeStatisticsTable 的 Java 方法注册为 time_statistics 表事件的脚本。

```
CALL ml_add_java_table_script(
  'ver1',
  'table1',
  'time_statistics',
  'ExamplePackage.ExampleClass.timeStatisticsTable' )
```

以下是 Java 方法 timeStatisticsTable 示例。它输出 upload_old_row_insert 事件的统计信息。

```
public String timeStatisticsTable(
  String username,
  String tableName,
  String eventName,
  int numberOfCalls,
  int minimumTime,
  int maximumTime,
  int totalTime ) {
  if( eventName.equals( "upload_old_row_insert") ) {
    java.lang.System.out.println(
     "upload_old_row_insert num_calls: " + numCalls +
     "total_time: " + totalTime );
  }
  return ( null );
}
```

.NET 示例

以下对 MobiLink 系统过程的调用在同步脚本版本 ver1 和表 table1 时将名为 TimeTableStats 的 .NET 方法注册为 time_statistics 表事件的脚本。

```
CALL ml_add_dnet_table_script(
  'ver1',
  'table1',
  'time_statistics',
  'TestScripts.Test.TimeTableStats'
)
```

以下是 .NET 方法 TimeTableStats 示例。它输出 upload_old_row_insert 事件的统计信息。

```
public string TimeTableStats(
  string user,
  string table,
  string eventName,
  int numberOfCalls,
  int minimumTime,
  int maximumTime,
  int totTime ) {
  if( event_name == "upload_old_row_insert") {
    System.Console.WriteLine(
      "upload_old_row_insert num_calls: " + num_calls +
      "total_time: " + total_time );
  }
  return ( null );
}
```

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

upload_delete 表事件

为 MobiLink 服务器提供一个在上载处理过程中使用的事件，该事件用于处理从远程数据库中删除的行。

参数

SQL 脚本的参数名称	顺序
r.*pk-column-1*	1
...	...
r.*pk-column-N*	*N*
r.*column-1*	*N* + 1
...	...
r.*column-M*	*N* + *M*

缺省操作

无。

注释

基于语句的 upload_delete 脚本用于处理远程数据库中删除的行。统一数据库中所执行的操作可以是 DELETE 语句，但不一定必须是。

您可以为远程数据库的每个表编写一个 upload_delete 脚本。

对于 Java 和 .NET 应用程序，此脚本必须返回有效的 SQL。

另请参见

- “脚本参数”一节第 299 页
- “添加和删除脚本”一节第 306 页
- “upload_insert 表事件”一节第 471 页
- “upload_update 表事件”一节第 488 页

SQL 示例

此示例来自 Contact 示例，并可在 *Samples\MobiLink\Contact\build_consol.sql* 中找到。它将从远程数据库中删除的客户标记为非活动。

```
CALL ml_add_table_script(
  'ver1',
  'table1',
  'upload_delete',
  'UPDATE Customer
   SET active = 0
   WHERE cust_id={ml r.cust_id}' )
```

Java 示例

以下对 MobiLink 系统过程的调用在同步脚本版本 ver1 时将名为 uploadDeleteTable 的 Java 方法注册为 upload_delete 表事件的脚本。

```
CALL ml_add_java_table_script(
  'ver1',
  'table1',
  'upload_delete',
  'ExamplePackage.ExampleClass.uploadDeleteTable' )
```

以下是 Java 方法 uploadDeleteTable 示例。它调用 genUD，从而动态生成一个 UPLOAD 语句。

```
public String uploadDeleteTable() {
  return( genUD(_curTable) );
}
```

.NET 示例

以下对 MobiLink 系统过程的调用在同步脚本版本 ver1 和表 table1 时将名为 UploadDelete 的 .NET 方法注册为 upload_delete 表事件的脚本。

```
CALL ml_add_dnet_table_script(
  'ver1',
  'table1',
  'upload_delete',
  'TestScripts.Test.UploadDelete'
)
```

以下是 .NET 方法 UploadDelete 示例。它调用 genUD，从而动态生成一个 UPLOAD 语句。

```
public string UploadDelete( object pk1 ) {
  return( genUD(_curTable) );
}
```


Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

upload_fetch 表事件

为检测行级冲突的目的从统一数据库的已同步表中读取行。

参数

SQL 脚本的参数名称	顺序
r.*primary-key-1*	1
r.*primary-key-2*	2
...	...
r.*primary-key-N*	N

缺省操作

无。

注释

基于语句的 upload_fetch 脚本用于从同步表中读取行以便检测冲突。它随 upload_update 事件使用。

结果集的列数必须与从远程数据库为此表上载的列数相匹配。如果返回值与上载的行中的前映像不匹配，则发生冲突。

不要在 upload_fetch 脚本中使用 READPAST 表提示。如果该脚本使用 READPAST 跳过锁定行，则同步逻辑会认为该行已被删除。根据您所定义的脚本，这会造成忽略上载的更新或触发冲突解决。忽略更新可能是不可接受的行为并可能是有害的。触发冲突解决可能不是一个问题，它取决于您已实现的解决逻辑。

您只能为远程数据库中的每个表编写一个 upload_fetch 脚本或一个 upload_fetch_column_conflict 脚本。

此脚本在未定义以下脚本时可以被忽略：upload_new_row_insert、upload_old_row_insert 和 resolve_conflict。

另请参见

- “脚本参数”一节第 299 页
- “添加和删除脚本”一节第 306 页
- “检测冲突”一节第 138 页
- “resolve_conflict 表事件”一节第 450 页
- “upload_delete 表事件”一节第 465 页
- “upload_insert 表事件”一节第 471 页
- “upload_update 表事件”一节第 488 页
- 在 “FROM 子句”一节 《SQL Anywhere 服务器 - SQL 参考》 中 "对 MobiLink 同步使用 READPAST"

Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

SQL 示例

以下 SQL 脚本来自 Contact 示例，并可在 *samples-dir\MobiLink\Contact\build_consol.sql* 中找到。它用于识别在上载远程数据库中 Product 表的更新行时发生的冲突。此脚本从名称也为 Product 的表中选择行，这两个表的名称也可以不匹配，匹配与否取决于统一数据库与远程数据库的模式。

```
CALL ml_add_table_script(
  'ver1',
  'Product',
  'upload_fetch',
  'SELECT id, name, size, quantity, unit_price
   FROM Product
   WHERE id={ml r.id}' )
```

Java 示例

此脚本必须返回有效的 SQL。

以下对 MobiLink 系统过程的调用在同步脚本版本 ver1 时将名为 uploadFetchTable 的 Java 方法注册为 upload_fetch 表事件的脚本。

```
CALL ml_add_java_table_script(
  'ver1',
  'Product',
  'upload_fetch',
  'ExamplePackage.ExampleClass.uploadFetchTable' )
```

以下是 Java 方法 uploadFetchTable 示例。它调用 genUF 来动态生成一条 UPLOAD 语句。

```
public String uploadFetchTable() {
  return( genUF(_curTable) );
}
```

.NET 示例

此脚本必须返回有效的 SQL。

以下对 MobiLink 系统过程的调用在同步脚本版本 ver1 时将名为 UploadFetchTable 的 .NET 方法注册为 upload_fetch 表事件的脚本。

```
CALL ml_add_dnet_table_script(
  'ver1',
  'Product',
  'upload_fetch',
  'TestScripts.Test.UploadFetchTable' )
```

以下是 .NET 方法 UploadFetchTable 示例。它调用 GenUF 来动态生成一条 UPLOAD 语句。

```
public string UploadFetchTable() {
  return( GenUF(_curTable) );
}
```

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

upload_fetch_column_conflict 表事件

从统一数据库的已同步表中读取行，以便进行列级别冲突检测。

参数

SQL 脚本的参数名称	顺序
r.*pk-column-1*	1
...	...
r.*pk-column-N*	*N*
r.*column-1*	*N* + 1
...	...
r.*column-M*	*N* + *M*

缺省操作

无。

注释

基于语句的 upload_fetch_column_conflict 脚本用于从已同步的表中读取列以便检测冲突。它随 upload_update 事件使用。

此脚本仅在两个用户更新同一列时会检测到冲突。不同的用户可更新同一行，只要他们不更新同一列，就不会产生冲突。

例如，当您的一个远程用户更新 ULOrder 表的 quant 列而另一个远程用户更新同一行的 notes 列时，使用 upload_fetch_column_conflict 脚本可以避免检测到冲突。仅在他们两人都更新 quant 列时，您才会检测到冲突。

您只能为远程数据库中的每个表编写一个 upload_fetch 脚本或一个 upload_fetch_column_conflict 脚本。

此脚本在未定义以下脚本时可以被忽略：upload_new_row_insert、upload_old_row_insert 和 resolve_conflict。

Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

另请参见

- “脚本参数”一节第 299 页
- “添加和删除脚本”一节第 306 页
- “检测冲突”一节第 138 页
- “upload_fetch 表事件”一节第 467 页
- “resolve_conflict 表事件”一节第 450 页
- “upload_delete 表事件”一节第 465 页
- “upload_insert 表事件”一节第 471 页
- “upload_update 表事件”一节第 488 页

Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

upload_insert 表事件

为 MobiLink 服务器提供一个在上载处理过程中使用的事件，该事件用于处理插入到远程数据库中的行。

参数

SQL 脚本的参数名称	顺序
r.*pk-column-1*	1
...	...
r.*pk-column-N*	*N*
r.*column-1*	*N* +1
...	...
r.*column-M*	*N* + *M*

缺省操作

无。

注释

基于语句的 upload_insert 脚本将用于直接插入列值。

您可以为远程数据库的每个表编写一个 upload_insert 脚本。

对于 Java 和 .NET 应用程序，此脚本必须返回有效的 SQL。

另请参见

- “脚本参数”一节第 299 页
- “添加和删除脚本”一节第 306 页
- “upload_delete 表事件”一节第 465 页
- “upload_update 表事件”一节第 488 页
- “upload_fetch 表事件”一节第 467 页

SQL 示例

此示例处理在远程数据库的 Customer 表中进行的插入。该脚本将值插入到统一数据库的名为 Customer 的表中。该表的最后一列将 Customer 标识为活动。最后一列不会在远程数据库中出现。

```
CALL ml_add_table_script(
 'ver1',
 'Customer',
 'upload_insert',
 'INSERT INTO Customer(
   cust_id,
   name,
   rep_id,
```

Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

```
    active )
  VALUES (
   {ml r.cust_id},
   {ml r.name},
   {ml r.rep_id},
   1 )' );
```

Java 示例

以下对 MobiLink 系统过程的调用在同步脚本版本 ver1 时将名为 uploadInsertTable 的 Java 方法注册为 upload_insert 表事件的脚本。

```
CALL ml_add_java_table_script(
  'ver1',
  'table1',
  'upload_insert',
  'ExamplePackage.ExampleClass.uploadInsertTable' )
```

以下是 Java 方法 uploadInsertTable 示例。它动态生成 INSERT 语句。此语法用于 SQL Anywhere 统一数据库。

```
public String uploadInsertTable() {
  return("INSERT INTO " + _curTable + getCols(_curTable)
    + " VALUES " + getQM(_curTable));
}
```

.NET 示例

以下对 MobiLink 系统过程的调用在同步脚本版本 ver1 和表 table1 时将名为 UploadInsert 的 .NET 方法注册为 upload_insert 表事件的脚本。此语法用于 SQL Anywhere 统一数据库。

```
CALL ml_add_dnet_table_script(
  'ver1',
  'table1',
  'upload_insert',
  'TestScripts.Test.UploadInsert'
)
```

以下是 .NET 方法 UploadInsert 示例。它返回一条针对 ULCustomer 表的 INSERT 语句。

```
public static string UploadInsert() {
   return("INSERT INTO ULCustomer( cust_id, cust_name ) VALUES ( {ml
r.cust_id}, {ml r.cust_name} )");
 }
```

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

upload_new_row_insert 表事件

用于基于语句的上载的冲突解决脚本通常要求访问从远程数据库上载的行的旧值及新值。此事件可用于处理从远程数据库中上载的行的新值和更新值。

参数

在下表中，说明部分提供 SQL 数据类型。如果您使用 Java 或 .NET 编写脚本，则应该使用相应的数据类型。请参见“SQL-Java 数据类型”一节第 498 页和“SQL-.NET 数据类型”一节第 556 页。

在 SQL 脚本中，可以使用名称或问号指定事件参数，但不能在一个脚本中混合使用名称和问号。如果使用问号，则参数必须按照如下所示的顺序并且仅当没有指定任何后继参数时才是可选的（例如，如果您想使用参数 2，则必须使用参数 1）。如果使用命名参数，则可以按照任何顺序指定任意参数子集。

SQL 脚本的参数名称	说明	顺序
s.remote_id	VARCHAR(128)。MobiLink 远程 ID。只有在使用命名参数时才能引用远程 ID。	不适用
s.username	VARCHAR(128)。MobiLink 用户名。此参数为可选参数。	可选
r.*pk-column-1*	旧（前映像）行的列值，其中已命名的参数被指定为以 **r.** 开头的列名	1（如果引用用户名则为 2）
...		...
r.*pk-column-N*	旧（前映像）行的列值，其中已命名的参数被指定为以 **r.** 开头的列名	*N*（如果引用用户名则为 *N* + 1）
r.*column-1*	旧（前映像）行的列值，其中已命名的参数被指定为以 **r.** 开头的列名	*N* + 1（如果引用用户名则为 *N*+2）
...	...	...
r.*column-M*	旧（前映像）行的列值，其中已命名的参数被指定为以 **r.** 开头的列名	*N* + *M*（如果引用用户名则为 *N*+*M*+1）

缺省操作

无。

注释

在 MobiLink 客户端向 MobiLink 服务器发送一个更新后的行时，发送的数据中不仅包含该行的新值（后映像），而且还将包含旧行值的副本（前映像）。前映像与统一数据库中的当前值不匹配时，则会检测到冲突。

此事件允许您将后映像的值保存到表中。此事件可以帮助您为基于语句的更新开发冲突解决过程。在对相应的统一数据库表执行更新之前，此事件的参数将保存远程数据库中的新行值。此事件也可用于在基于语句的强制冲突模式中插入行。

此事件的脚本通常是一条插入语句，它将新行插入到临时表中以供 resolve_conflict 脚本使用。

您可以为远程数据库的每个表编写一个 upload_new_row_insert 脚本。

对于 Java 和 .NET 应用程序，此脚本必须返回有效的 SQL。

另请参见

- “脚本参数”一节第 299 页
- “添加和删除脚本”一节第 306 页
- “冲突处理”一节第 137 页
- “resolve_conflict 表事件”一节第 450 页
- “upload_old_row_insert 表事件”一节第 476 页
- “upload_update 表事件”一节第 488 页
- “强制冲突”一节第 145 页
- “在脚本中使用远程 ID 和 MobiLink 用户名”一节 《MobiLink - 客户端管理》

SQL 示例

此示例处理远程数据库中对 product 表进行的更新。该脚本将新行值插入到名为 product_conflict 的全局临时表中。表的最后一列将该行标识为新行。

```
CALL ml_add_table_script(
 'ver1',
 'table1',
 'upload_new_row_insert',
 'INSERT INTO DBA.product_conflict(
   id,
   name,
   size,
   quantity,
   unit_price,
   row_type )
  VALUES(
   {ml r.id},
   {ml r.name},
   {ml r.size},
   {ml r.quantity},
   {ml r.unit_price},
   ''New'' )' )
```

Java 示例

以下对 MobiLink 系统过程的调用在同步脚本版本 ver1 时将名为 uploadNewRowInsertTable 的 Java 方法注册为 upload_new_row_insert 表事件的脚本。

```
CALL ml_add_java_table_script(
  'ver1',
  'table1',
  'upload_new_row_insert',
  'ExamplePackage.ExampleClass.uploadNewRowInsertTable'
)
```

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

以下是 Java 方法 uploadNewRowInsertTable 示例。它动态生成 INSERT 语句。此语法用于 SQL Anywhere 统一数据库。

```
public String uploadNewRowInsertTable() {
  return("insert into" + _curTable + "_new" +
    getCols(_curTable) + "values" + getNamedParams(_curTable));
}
```

.NET 示例

以下对 MobiLink 系统过程的调用在同步脚本版本 ver1 时将名为 UploadNewRowInsertTable 的 .NET 方法注册为 upload_new_row_insert 表事件的脚本。

```
CALL ml_add_dnet_table_script(
  'ver1',
  'table1',
  'upload_new_row_insert',
  'TestScripts.Test.UploadNewRowInsertTable'
)
```

以下是 .NET 方法 UploadNewRowInsertTable 示例。它动态生成 INSERT 语句。此语法用于 SQL Anywhere 统一数据库。

```
public string UploadNewRowInsertTable() {
  return("insert into" + _curTable + "_new" +
    GetCols(_curTable) + "values" + GetNamedParams(_curTable));
}
```

upload_old_row_insert 表事件

用于基于语句的上载的冲突解决脚本通常要求访问从远程数据库上载的行的旧值及新值。此事件可用于处理从远程数据库中上载的行的旧值。

参数

在下表中，说明部分提供 SQL 数据类型。如果您使用 Java 或 .NET 编写脚本，则应该使用相应的数据类型。请参见“SQL-Java 数据类型”一节第 498 页和“SQL-.NET 数据类型”一节第 556 页。

在 SQL 脚本中，可以使用名称或问号指定事件参数，但不能在一个脚本中混合使用名称和问号。如果使用问号，则参数必须按照如下所示的顺序并且仅当没有指定任何后继参数时才是可选的（例如，如果您想使用参数 2，则必须使用参数 1）。如果使用命名参数，则可以按照任何顺序指定任意参数子集。

SQL 脚本的参数名称	说明	顺序
s.remote_id	VARCHAR(128)。MobiLink 远程 ID。只有在使用命名参数时才能引用远程 ID。	不适用
s.username	VARCHAR(128)。MobiLink 用户名。此参数是可选的。	可选
r.*pk-column-1*	旧（前映像）行的列值，其中已命名的参数被指定为以 **r.** 开头的列名	1（如果引用用户名则为 2）
...		...
r.*pk-column-N*	旧（前映像）行的列值，其中已命名的参数被指定为以 **r.** 开头的列名	*N*（如果引用用户名则为 *N* + 1）
r.*column-1*	旧（前映像）行的列值，其中已命名的参数被指定为以 **r.** 开头的列名	*N* + 1（如果引用用户名则为 *N*+2）
...	...	...
r.*column-M*	旧（前映像）行的列值，其中已命名的参数被指定为以 **r.** 开头的列名	*N* + *M*（如果引用用户名则为 *N*+*M*+1）

缺省操作

无。

注释

在 MobiLink 客户端向 MobiLink 服务器发送一个更新后的行时，发送的数据中不仅包含该行的新值（后映像），而且还将包含旧行值的副本（前映像）。前映像与统一数据库中的当前值不匹配时，则会检测到冲突。

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

此事件允许您将前映像的值保存到表中。此事件可以帮助您为基于语句的更新开发冲突解决过程。在对相应的统一数据库表执行更新之前，此事件的参数将保存远程数据库中的旧行值。此事件也可用于在基于语句的强制冲突模式中插入行。

此事件的脚本通常是一条插入语句，它将旧行插入到临时表中以供 resolve_conflict 脚本使用。

您可以为远程数据库的每个表编写一个 upload_old_row_insert 脚本。

对于 Java 和 .NET 应用程序，此脚本必须返回有效的 SQL。

另请参见

- “脚本参数”一节第 299 页
- “添加和删除脚本”一节第 306 页
- “冲突处理”一节第 137 页
- “resolve_conflict 表事件”一节第 450 页
- “upload_new_row_insert 表事件”一节第 473 页
- “upload_update 表事件”一节第 488 页
- “强制冲突”一节第 145 页
- “在脚本中使用远程 ID 和 MobiLink 用户名”一节 《MobiLink - 客户端管理》

SQL 示例

此示例处理远程数据库中对 product 表进行的更新。该脚本将旧的行值插入到名为 product_conflict 的全局临时表中。表的最后一列将该行标识为旧行。

```
CALL ml_add_table_script(
 'ver1',
 'table1',
 'upload_old_row_insert',
 'INSERT INTO DBA.product_conflict (
   id,
   name,
   size,
   quantity,
   unit_price,
   row_type )
  VALUES (
   {ml r.id},
   {ml r.name},
   {ml r.size},
   {ml r.quantity},
   {ml r.unit_price},
   ''Old'' )' )
```

Java 示例

以下对 MobiLink 系统过程的调用在同步脚本版本 ver1 时将名为 uploadOldRowInsertTable 的 Java 方法注册为 upload_old_row_insert 表事件的脚本。

```
CALL ml_add_java_table_script(
  'ver1',
  'table1',
  'upload_old_row_insert',
  'ExamplePackage.ExampleClass.uploadOldRowInsertTable'
)
```

以下是 Java 方法 uploadOldRowInsertTable 示例。它动态生成 INSERT 语句。

```
public String uploadOldRowInsertTable() {
  return( "old" + getCols(_curTable) +
    "values" + getNamedParams(_curTable));
}
```

.NET 示例

以下对 MobiLink 系统过程的调用在同步脚本版本 ver1 时将名为 UploadOldRowInsertTable 的 .NET 方法注册为 upload_old_row_insert 表事件的脚本。

```
CALL ml_add_dnet_table_script(
  'ver1',
  'table1',
  'upload_old_row_insert',
  'TestScripts.Test.UploadOldRowInsertTable'
)
```

以下是 .NET 方法 UploadOldRowInsertTable 示例。它动态生成 UPLOAD 语句。

```
public string UploadOldRowInsertTable() {
  return( "old" + GetCols(_curTable) +
    "values" + GetNamedParams(_curTable));
}
```

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

upload_statistics 连接事件

为上载操作跟踪同步统计信息。

参数

在下表中，说明部分提供 SQL 数据类型。如果您使用 Java 或 .NET 编写脚本，则应该使用相应的数据类型。请参见“SQL-Java 数据类型”一节第 498 页和“SQL-.NET 数据类型”一节第 556 页。

在 SQL 脚本中，可以使用名称或问号指定事件参数，但不能在一个脚本中混合使用名称和问号。如果使用问号，则参数必须按照如下所示的顺序并且仅当没有指定任何后继参数时才是可选的（例如，如果您想使用参数 2，则必须使用参数 1）。如果使用命名参数，则可以按照任何顺序指定任意参数子集。

SQL 脚本的参数名称	说明	顺序
s.remote_id	VARCHAR(128)。MobiLink 远程 ID。只有在使用命名参数时才能引用远程 ID。	不适用
s.username	VARCHAR(128)。MobiLink 用户名。	1
s.warnings	INTEGER。发生的警告的数目。	2
s.errors	INTEGER。发生的错误的数目。	3
s.inserted_rows	INTEGER。成功插入到统一数据库中的行数。	4
s.deleted_rows	INTEGER。从统一数据库中成功删除的行数。	5
s.updated_rows	INTEGER。在统一数据库中成功更新的行数。	6
s.conflicted_inserts	INTEGER。始终为零。	7
s.conflicted_deletes	INTEGER。始终为零。	8
s.conflicted_updates	INTEGER。导致冲突的更新行数目。仅当对某行成功调用了解决冲突脚本时才包括该行。	9

SQL 脚本的参数名称	说明	顺序
s.ignored_inserts	INTEGER。忽略的上载插入行总数。它们被忽略的原因是：1) 在正常模式中没有 upload_insert 脚本或在强制冲突模式中没有 upload_new_row_insert 脚本，2) MobiLink 服务器调用相应脚本时出现了错误并且 handle_error 或 handle_odbc_error 事件返回 1000。	10
s.ignored_deletes	INTEGER。如果定义了 handle_error 或 handle_odbc_error 并返回 1000 或者对给定表没有定义 upload_delete 脚本，当调用 upload_delete 脚本时导致出错的上载删除行数。	11
s.ignored_updates	INTEGER。导致冲突但冲突解决脚本未能成功调用或者未定义 upload_update 脚本的上载更新行数。	12
s.bytes	INTEGER。MobiLink 服务器内用来存储上载的内存量。	13
s.deadlocks	INTEGER。统一数据库中为同步已检测到的死锁的数目。	14

缺省操作

无。

注释

upload_statistics 事件可用于为所有用户收集与上载有关的统计信息。upload_statistics 连接脚本在上载事务结束时执行提交操作之前被调用。

另请参见

- “脚本参数”一节第 299 页
- “添加和删除脚本”一节第 306 页
- “download_statistics 连接事件”一节第 376 页
- “download_statistics 表事件”一节第 379 页
- “upload_statistics 表事件”一节第 483 页
- “synchronization_statistics 连接事件”一节第 453 页
- “synchronization_statistics 表事件”一节第 456 页
- “time_statistics 连接事件”一节第 459 页
- “time_statistics 表事件”一节第 462 页
- “MobiLink 监控器”第 167 页
- “在脚本中使用远程 ID 和 MobiLink 用户名”一节 《MobiLink - 客户端管理》

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

SQL 示例

下面的示例将上载操作的同步统计信息插入到表 upload_summary_audit 中。

```
CALL ml_add_connection_script (
 'ver1',
 'upload_statistics',
 'INSERT INTO upload_summary_audit (
   ml_user,
   warnings,
   errors,
   inserted_rows,
   deleted_rows,
   updated_rows,
   conflicted_inserts,
   conflicted_deletes,
   conflicted_updates,
   bytes,
   ignored_inserts,
   ignored_deletes,
   ignored_updates,
   bytes, deadlocks )
  VALUES (
   {ml s.username},
   {ml s.warnings},
   {ml s.errors},
   {ml s.inserted_rows},
   {ml s.deleted_rows},
   {ml s.updated_rows},
   {ml s.conflicted_inserts},
   {ml s.conflicted_deletes},
   {ml s.conflicted_updates},
   {ml s.ignored_inserts},
   {ml s.ignored_deletes},
   {ml s.ignored_updates},
   {ml s.bytes},
   {ml s.deadlocks} ) ' )
```

在审计表中插入统计信息之后，您将可以使用这些统计信息监控同步过程，并在条件允许时进行优化。

Java 示例

以下对 MobiLink 系统过程的调用在同步脚本版本 ver1 时将名为 uploadStatisticsConnection 的 Java 方法注册为 upload_statistics 连接事件的脚本。

```
CALL ml_add_java_connection_script(
   'ver1',
   'upload_statistics',
   'ExamplePackage.ExampleClass.uploadStatisticsConnection' )
```

以下是 Java 方法 uploadStatisticsConnection 示例。它将某些统计信息记录到 MobiLink 消息日志。（请注意：将统计信息记录到 MobiLink 消息日志在开发时可能会有帮助，但会降低生产服务器的性能。）

```
public String uploadStatisticsConnection(
  String user,
  int warnings,
  int errors,
  int insertedRows,
  int deletedRows,
  int updatedRows,
```

```
  int conflictedInserts,
  int conflictedDeletes,
  int conflictedUpdates,
  int ignoredInserts,
  int ignoredDeletes,
  int ignoredUpdates,
  int bytes,
  int deadlocks ) {
  java.lang.System.out.println( "updated rows: " +
   updatedRows );
  return ( null );
}
```

.NET 示例

以下对 MobiLink 系统过程的调用在同步脚本版本 ver1 时将名为 UploadStats 的 .NET 方法注册为 upload_statistics 连接事件的脚本。

```
CALL ml_add_dnet_connection_script(
  'ver1',
  'upload_statistics',
  'TestScripts.Test.UploadStats'
)
```

以下是 .NET 方法 UploadStats 示例。它将某些统计信息记录到 MobiLink 消息日志。（请注意：将统计信息记录到 MobiLink 消息日志在开发时可能会有帮助，但会降低生产服务器的性能。）

```
public string UploadStats (
  string user,
  int warnings,
  int errors,
  int insertedRows,
  int deletedRows,
  int updatedRows,
  int conflictInserts,
  int conflictDeletes,
  int conflictUpdates,
  int ignoredInserts,
  int ignoredDeletes,
  int ignoredUpdates,
  int bytes,
  int deadlocks ) {
  System.Console.WriteLine( "updated rows: " +
    updatedRows );
  return ( null );
}
```

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

upload_statistics 表事件

为特定表跟踪有关上载操作的同步统计信息。

参数

在下表中，说明部分提供 SQL 数据类型。如果您使用 Java 或 .NET 编写脚本，则应该使用相应的数据类型。请参见“SQL-Java 数据类型”一节第 498 页和“SQL-.NET 数据类型”一节第 556 页。

在 SQL 脚本中，可以使用名称或问号指定事件参数，但不能在一个脚本中混合使用名称和问号。如果使用问号，则参数必须按照如下所示的顺序并且仅当没有指定任何后继参数时才是可选的（例如，如果您想使用参数 2，则必须使用参数 1）。如果使用命名参数，则可以按照任何顺序指定任意参数子集。

SQL 脚本的参数名称	说明	顺序
s.remote_id	VARCHAR(128)。MobiLink 远程 ID。只有在使用命名参数时才能引用远程 ID。	不适用
s.username	VARCHAR(128)。MobiLink 用户名。	1
s.table	VARCHAR(128)。表名。	2
s.warnings	INTEGER。表上载过程中发出的警告的数目。	3
s.errors	INTEGER。表上载过程中发生的错误的数目，包括已处理的错误。	4
s.inserted_rows	INTEGER。成功插入到统一数据库中的行数。	5
s.deleted_rows	INTEGER。从统一数据库中成功删除的行数。	6
s.updated_rows	INTEGER。	7
s.conflicted_inserts	INTEGER。始终为零。	8
s.conflicted_deletes	INTEGER。始终为零。	9
s.conflicted_updates	INTEGER。导致冲突的更新行数目。仅当对某行成功调用了解决冲突脚本时才包括该行。	10

Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

SQL 脚本的参数名称	说明	顺序
s.ignored_inserts	INTEGER。忽略的上载插入行总数。它们被忽略的原因是：1) 在正常模式中没有 upload_insert 脚本或在强制冲突模式中没有 upload_new_row_insert 脚本，2) MobiLink 服务器调用相应脚本时出现了错误并且 handle_error 或 handle_odbc_error 事件返回 1000。	11
s.ignored_deletes	INTEGER。如果定义了 handle_error 或 handle_odbc_error 并返回 1000 或者对给定表没有定义 upload_delete 脚本，当调用 upload_delete 脚本时导致出错的上载删除行数。	12
s.ignored_updates	INTEGER。导致冲突但冲突解决脚本未能成功调用或者未定义 upload_update 脚本的上载更新行数。	13
s.bytes	INTEGER。MobiLink 服务器内用来存储上载的内存量。	14
s.deadlocks	INTEGER。统一数据库中为同步已检测到的死锁的数目。	15

缺省操作

无。

注释

upload_statistics 事件可用于为所有用户收集有关同步事件（当它们应用于任何表时）的重要统计信息。upload_statistics 表脚本在上载事务结束时执行提交操作之前被调用。

另请参见

- “脚本参数”一节第 299 页
- “添加和删除脚本”一节第 306 页
- “download_statistics 连接事件”一节第 376 页
- “upload_statistics 连接事件”一节第 479 页
- “upload_statistics 表事件”一节第 483 页
- “synchronization_statistics 连接事件”一节第 453 页
- “synchronization_statistics 表事件”一节第 456 页
- “time_statistics 连接事件”一节第 459 页
- “time_statistics 表事件”一节第 462 页
- “MobiLink 监控器”第 167 页
- “在脚本中使用远程 ID 和 MobiLink 用户名”一节 《MobiLink - 客户端管理》

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

SQL 示例

以下示例在用于跟踪上载统计信息的表中插入一行。

```
CALL ml_add_connection_script(
 'ver1',
 'upload_statistics',
 'INSERT INTO my_upload_statistics (
   user_name,
   table_name,
   num_warnings,
   num_errors,
   inserted_rows,
   deleted_rows,
   updated_rows,
   conflicted_inserts,
   conflicted_deletes,
   conflicted_updates,
   ignored_inserts,
   ignored_deletes,
   ignored_updates, bytes,
   deadlocks )
  VALUES(
   {ml s.username},
   {ml s.table},
   {ml s.warnings},
   {ml s.errors},
   {ml s.inserted_rows},
   {ml s.deleted_rows},
   {ml s.updated_rows},
   {ml s.conflicted_inserts},
   {ml s.conflicted_deletes},
   {ml s.conflicted_updates},
   {ml s.ignored_inserts},
   {ml s.ignored_deletes},
   {ml s.ignored_updates},
   {ml s.bytes},
   {ml s.deadlocks} )'  )
```

以下示例在 Oracle 统一数据库中运行。

```
CALL ml_add_connection_script(
 'ver1',
 'upload_statistics',
 'INSERT INTO upload_tables_audit (
  id,
  user_name,
  table,
  warnings,
  errors,
  inserted_rows,
  deleted_rows,
  updated_rows,
  conflicted_inserts,
  conflicted_deletes,
  conflicted_updates,
  ignored_inserts,
  ignored_deletes,
  ignored_updates,
  bytes,
  deadlocks )
 VALUES (
   ut_audit.nextval,
   {ml s.username},
```

```
    {ml s.table},
    {ml s.warnings},
    {ml s.errors},
    {ml s.inserted_rows},
    {ml s.deleted_rows},
    {ml s.updated_rows},
    {ml s.conflicted_inserts},
    {ml s.conflicted_deletes},
    {ml s.conflicted_updates},
    {ml s.ignored_inserts},
    {ml s.ignored_deletes},
    {ml s.ignored_updates},
    {ml s.bytes},
    {ml s.deadlocks} )' )
```

在审计表中插入统计信息之后，您将可以使用这些统计信息监控同步过程，并在条件允许时进行优化。

Java 示例

以下对 MobiLink 系统过程的调用在同步脚本版本 ver1 时将名为 uploadStatisticsTable 的 Java 方法注册为 upload_statistics 表事件的脚本。

```
CALL ml_add_java_table_script(
  'ver1',
  'table1',
  'upload_statistics',
  'ExamplePackage.ExampleClass.uploadStatisticsTable' )
```

以下是 Java 方法 uploadStatisticsTable 示例。它将某些统计信息记录到 MobiLink 消息日志。（请注意：将统计信息记录到 MobiLink 消息日志在开发时可能会有帮助，但会降低生产服务器的性能。）

```
public String uploadStatisticsTable(
  String user,
  int warnings,
  int errors,
  int insertedRows,
  int deletedRows,
  int updatedRows,
  int conflictedInserts,
  int conflictedDeletes,
  int conflictedUpdates,
  int ignoredInserts,
  int ignoredDeletes,
  int ignoredUpdates,
  int bytes,
  int deadlocks ) {
  java.lang.System.out.println( "updated rows: " +
    updatedRows );
  return ( null );
}
```

.NET 示例

以下对 MobiLink 系统过程的调用在同步脚本版本 ver1 和表 table1 时将名为 UploadTableStats 的 .NET 方法注册为 upload_statistics 表事件的脚本。

```
CALL ml_add_dnet_table_script(
  'ver1',
  'table1',
  'upload_statistics',
```

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

```
    'TestScripts.Test.UploadTableStats'
  )
```

以下是 .NET 方法 uploadStatisticsTable 示例。它将某些统计信息记录到 MobiLink 消息日志。（请注意：将统计信息记录到 MobiLink 消息日志在开发时可能会有帮助，但会降低生产服务器的性能。）

```
public string UploadTableStats(
  string user,
  string table,
  int warnings,
  int errors,
  int insertedRows,
  int deletedRows,
  int updatedRows,
  int conflictInserts,
  int conflictDeletes,
  int conflictUpdates,
  int ignoredInserts,
  int ignoredDeletes,
  int ignoredUpdates,
  int bytes,
  int deadlocks ) {
  System.Console.WriteLine( "updated rows: " +
    updatedRows );
  return ( null );
}
```

upload_update 表事件

为 MobiLink 服务器提供一个在上载处理过程中使用的事件，该事件用于处理在远程数据库中进行更新的行。

参数

参数	顺序
r.*column-1*	1
...	...
r.*column-M*	M
r.*pk-column-1*	M + 1
...	...
r.*pk-column-N*	M + N
o.*column-N*	M + N + 1
...	...
o.*column-M*	M + N + M

缺省操作

无。

注释

基于语句的 upload_update 脚本可用于直接更新列值，该列值由 UPLOAD 语句指定。

WHERE 子句必须包括正被同步的所有主键列。SET 子句必须包含正被同步的所有非主键列。

您可以在 SET 子句中使用表中存在的任意非主键列，MobiLink 将发送正确的列值数目。同样，在 WHERE 子句中，您也可以使用任意数目的主键，但它们必须都在此处指定，并且 MobiLink 能够发送正确的值。MobiLink 将发送这些列值和主键值，发送的顺序与表模式的 MobiLink 报告中列和主键出现的顺序相同。您可以使用 -vh 选项为此表模式确定列的顺序。

例如，在以下 upload_update 脚本中，问号的顺序正确：

```
UPDATE MyTable
  SET column_1 = ?, ..., column_M = ?
  WHERE pk_column_1 = ? AND ... AND pk_column_N = ?
```

您可以为远程数据库的每个表编写一个 upload_update 脚本。

对于 Java 和 .NET 应用程序，此脚本必须返回有效的 SQL。

要使用 upload_update 脚本检测冲突，请在 WHERE 子句中包括所有非主键列。

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

```
UPDATE table-name
SET col1 = {ml r.col1}, col2 = {ml r. col2 } ...
WHERE pk1 = {ml r.pk1} AND pk2 = {ml r.pk2} ...
  AND col1 = {ml o.col1} AND col2 ={ml o.col2} ...
```

在此语句中，col1 和 col2 是非主键列，而 pk1 和 pk2 是主键列。传递到第二组非主键列中的值是更新行的前映像。WHERE 子句将比较从远程数据库上载的旧值与统一数据库中的当前值。如果两个值不匹配，将忽略更新，保留统一数据库中已有的值。

另请参见

- “脚本参数”一节第 299 页
- “添加和删除脚本”一节第 306 页
- “使用 upload_update 脚本检测冲突”一节第 139 页
- “使用 upload_update 脚本解决冲突”一节第 142 页
- “upload_delete 表事件”一节第 465 页
- “upload_fetch 表事件”一节第 467 页
- “upload_insert 表事件”一节第 471 页

SQL 示例

此示例处理远程数据库中对 Customer 表进行的更新。该脚本将更新统一数据库中名为 Customer 的表中的值。

```
CALL ml_add_table_script(
  'ver1',
  'table1',
  'upload_update',
'UPDATE Customer
   SET name = {ml r.name}, rep_id = {ml r.rep_id}
      WHERE cust_id = {ml o.cust_id}')
```

Java 示例

以下对 MobiLink 系统过程的调用在同步脚本版本 ver1 时将名为 uploadUpdateTable 的 Java 方法注册为 upload_update 表事件的脚本。

```
CALL ml_add_java_table_script(
  'ver1',
  'table1',
  'upload_update',
  'ExamplePackage.ExampleClass.uploadUpdateTable' )
```

以下是 Java 方法 uploadUpdateTable 示例。它调用名为 genUU 的方法来动态生成一条 UPLOAD 语句。

```
public String uploadUpdateTable() {
  return( genUU(_curTable) );
}
```

.NET 示例

以下对 MobiLink 系统过程的调用在同步脚本版本 ver1 和表 table1 时将名为 UploadUpdate 的 .NET 方法注册为 upload_update 表事件的脚本。

```
CALL ml_add_dnet_table_script(
  'ver1',
```

```
  'table1',
  'upload_update',
  'TestScripts.Test.UploadUpdate'
)
```

以下是 .NET 方法 UploadUpdate 示例。它调用名为 GenUU 的方法来动态生成一条 UPLOAD 语句。

```
public string UploadUpdate() {
  return ( genUU(_curTable) );
}
```

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

MobiLink 服务器 API

本节介绍用于 Java 和 .NET 的 MobiLink 服务器 API。

使用 Java 语言编写同步脚本 493
使用 .NET 编写同步脚本 551
直接行处理 609

使用 Java 语言编写同步脚本

目录

Java 同步逻辑简介 494
设置 Java 同步逻辑 495
编写 Java 同步逻辑 497
Java 同步示例 504
用于 Java 的 MobiLink 服务器 API 参考 508

Java 同步逻辑简介

MobiLink 服务器的操作通过编写同步脚本进行控制。您可以使用 SQL、.NET 或 Java 实现这些脚本。Java 同步逻辑可以像 SQL 逻辑一样工作：正如 MobiLink 服务器可以在 MobiLink 事件发生时访问 SQL 脚本一样，它也可以在 MobiLink 事件发生时调用 Java 方法。Java 方法可以向 MobiLink 返回一个 SQL 字符串。

本节将介绍如何设置、开发和运行 Java 同步逻辑。其中包含一个示例应用程序和用于 Java 参考的 MobiLink 服务器 API。

另请参见

- “教程：使用 Java 同步逻辑” 《MobiLink - 入门》
- “用于编写服务器端同步逻辑的选项”一节 《MobiLink - 入门》
- “编写同步脚本”第 293 页

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

设置 Java 同步逻辑

安装 SQL Anywhere 时，安装程序会自动设置用于 Java 的 MobiLink 服务器 API 类的位置。启动 MobiLink 服务器时，它会自动将这些类加入类路径中。用于 Java 的 MobiLink 服务器 API 类位于 *install-dir\Java\mlscript.jar* 中。

♦ 使用 Java 实现同步脚本

1. 创建自己的类。为每个所需的同步脚本编写方法。这些方法必须是公共的。类在程序包中必须是公共的。

   请参见“方法”一节第 499 页。

   每个包含非静态方法的类都应有一个公共的构造函数。在第一次调用每个类中的方法时，MobiLink 服务器都会自动实例化此类。

   请参见“构造函数”一节第 498 页。

2. 编译类时必须包括 JAR 文件 *java\mlscript.jar*。

   例如，

   ```
   javac MyClass.java -classpath "c:\Program Files\SQL Anywhere 11\java
   \mlscript.jar"
   ```

3. 在统一数据库的 MobiLink 系统表中，为每个同步脚本指定要调用的程序包、类和方法的名称。每个脚本版本中只允许存在一个类。

   例如，可以使用 ml_add_java_connection_script 存储过程或 ml_add_java_table_script 存储过程将此信息添加到 MobiLink 系统表中。

   例如，如果在 SQL Anywhere 数据库中运行以下 SQL 语句，会指定每当发生 authenticate_user 连接级事件时脚本版本 ver1 将运行 myPackage.myClass.myMethod。指定的方法必须是公共 Java 方法的完全限定名称，而且名称应区分大小写。

   ```
   call ml_add_java_connection_script('ver1',
   'authenticate_user', 'myPackage.myClass.myMethod')
   ```

   有关添加脚本的详细信息，请参见：

   - “用于添加或删除脚本的系统过程”一节第 624 页
   - “ml_add_java_connection_script 系统过程”一节第 630 页
   - “ml_add_java_table_script 系统过程”一节第 631 页

4. 指示 MobiLink 服务器装载类。设置 Java 同步逻辑中至关重要的部分是告知虚拟机到哪里寻找 Java 类。有两种方法可实现这一点：

   - 使用 mlsrv11 -sl java -cp 选项来指定要在其中搜索类的一组目录或 jar 文件。例如，可运行以下命令：

     ```
     mlsrv11 -c "dsn=consolidated1" -sl java (-cp %classpath%;c:\local\Java
     \myclasses.jar)
     ```

     MobiLink 服务器会自动将用于 Java 的 MobiLink 服务器 API 类 (java\mlscript.jar) 的位置附加到这组目录或 jar 文件中。-sl java 选项还强制在服务器启动时装载 Java VM。

Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

有关可用 Java 选项的详细信息，请参见“-sl java 选项”一节第 87 页。

- 显式设置类路径。使用下面这样的语句为用户定义的类设置类路径：

  ```
  SET classpath=%classpath%;c:\local\Java\myclasses.jar
  ```

  如果系统类路径中已经包含 Java 同步逻辑类，则无需更改 MobiLink 服务器命令行。

  可以使用 -sl java 选项强制在服务器启动时装载 Java 虚拟机。否则 Java 虚拟机将在第一个 Java 方法执行时启动。

  有关可用 Java 选项的详细信息，请参见“-sl java 选项”一节第 87 页。

5. 在 Unix 上，如果想要装载特定的 JRE，应当设置 LD_LIBRARY_PATH（在 AIX 上为 LIBPATH，在 HP-UX 上为 SHLIB_PATH），使之包括含有该 JRE 的目录。此目录必须列在所有 SQL Anywhere 安装目录之前。

另请参见

- “编写 Java 同步逻辑”一节第 497 页
- “Java 同步示例”一节第 504 页
- “教程：使用 Java 同步逻辑” 《MobiLink - 入门》
- “用于 Java 的 MobiLink 服务器 API 参考”一节第 508 页
- “用于编写服务器端同步逻辑的选项”一节 《MobiLink - 入门》
- “编写同步脚本”第 293 页

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

编写 Java 同步逻辑

要编写 Java 同步逻辑，需要了解有关 MobiLink 事件的知识、一些有关 Java 的知识以及有关用于 Java 的 MobiLink 服务器 API 的知识。

有关 API 的完整说明，请参见“用于 Java 的 MobiLink 服务器 API 参考”一节第 508 页。

Java 同步逻辑可用于维护状态信息以及实现围绕上载和下载事件的逻辑。例如，使用 Java 编写的 begin_synchronization 脚本能够将 MobiLink 用户名存储在变量中。在同步过程后期调用的脚本可以访问此变量。此外，还可以在统一数据库中的行提交前后，使用 Java 访问这些行。

使用 Java 可以降低对统一数据库的依赖。通过将统一数据库升级到新版本或切换到另一数据库管理系统，可使行为受到的影响得以减少。

直接行处理

可以使用 MobiLink 直接行处理将远程数据传递给任何中央数据源、应用程序或 Web 服务。直接行处理使用用于 Java 或 .NET 的 MobiLink 服务器 API 中的特殊类来直接访问同步数据。

请参见“直接行处理”第 609 页。

类实例

MobiLink 服务器在连接级别将类实例化。当到达某个已经为其编写了非静态 Java 方法的事件时，如果在现有连接上尚未创建实例，MobiLink 服务器会自动创建该类的一个实例。

请参见“构造函数”一节第 556 页。

对于一个脚本版本，所有与连接级别或表级别事件直接关联的方法**必须属于同一个类**。

对于每个数据库连接，类在实例化后会一直持续，直到连接关闭。因此，多个连续的同步会话可能使用同一个实例。除非将其显式地清除，公共或专用变量中的信息会跨同一连接上的多个同步而持续存在。

您也可以使用静态的类或变量。在这种情况下，这些值在所有连接间都可用。

只有在与统一数据库的连接关闭时，MobiLink 服务器才会自动删除类实例。

事务

有关事务的一般规则适用于 Java 方法。数据库事务的启动和运行时间对同步进程来说至关重要。事务只能由 MobiLink 服务器启动和结束。在 Java 方法中显式地通过同步连接提交或回退事务会破坏同步进程的完整性，并且可能导致错误。

这些规则只适用于由 MobiLink 服务器创建的数据库连接，特别是由方法返回的 SQL 语句。如果您的类创建了其它数据库连接，则可使用现有管理规则来管理通过其它数据库连接创建的类。

Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

SQL-Java 数据类型

下表显示了 SQL 数据类型和相应的 Java 数据类型。

SQL 数据类型	相应的 Java 数据类型
VARCHAR	java.lang.String
CHAR	java.lang.String
INTEGER	int 或 Integer
BINARY	byte[ ]
TIMESTAMP	java.sql.Timestamp
INOUT INTEGER	ianywhere.ml.script.InOutInteger
INOUT VARCHAR	ianywhere.ml.script.InOutString
INOUT CHAR	ianywhere.ml.script.InOutString
INOUT BYTEARRAY	ianywhere.ml.script.InOutByteArray

如果 ianywhere.ml.script 程序包不在类路径中，MobiLink 服务器会自动添加到其中。但在编译类时，需要添加 *install-dir\Java\mlscript.jar* 路径。

构造函数

类的构造函数可能会有两个可能签名中的一个。

```
public MyScriptClass(ianywhere.ml.script.DBConnectionContext sc)
```

或者

```
public MyScriptClass()
```

传递给您的同步上下文是针对某个连接的，MobiLink 服务器通过该连接与当前用户保持同步。

DBConnectionContext.getConnection 方法返回的数据库连接与 MobiLink 用来与当前用户保持同步的数据库连接相同。您可以在此连接上执行语句，但是不能提交或回退事务。这些事务由 MobiLink 服务器进行管理。

MobiLink 服务器倾向于使用具有第一个签名的构造函数。只有在具有第一个签名的构造函数不存在时，才会使用无参数的构造函数。

请参见“DBConnectionContext 接口”一节第 508 页。

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

方法

一般说来，您为每个同步事件实现一种方法。这些方法必须是公共的。如果这些方法是专用的，则 MobiLink 服务器无法使用它们且无法识别它们的存在。

方法的名称并不重要，只要与统一数据库 ml_script 表中所指定的名称匹配即可。但在本文档所含的示例中，方法名称与 MobiLink 事件的名称相同，因为这种命名约定使 Java 代码更具可读性。

方法的签名应与该事件脚本的签名相匹配，不过如果参数列表尾的参数值不再需要，可以将参数列表截断。您应该只接受所需的参数，因为开销的大小与传递参数的多少有关。

但该方法不能重载。在 ml_script 系统表中，每个类只能有一个方法原型。

注册方法

创建方法之后，必须将其注册。注册方法时，会在统一数据库 MobiLink 系统表中创建对此方法的引用，以便事件发生时对其进行调用。方法注册与同步脚本添加采用同样的方式，不同的是前者只是将方法名称添加到 MobiLink 系统表中，而不是添加整个 SQL 脚本。

请参见“添加和删除脚本”一节第 306 页。

返回值

为进行基于 SQL 的上载或下载所调用的方法必须返回有效的 SQL 语言语句。这些方法的返回类型必须是 java.lang.String，而不允许是其它返回类型。

所有其它脚本的返回类型必须是 java.lang.String 或 void。而不允许是其它类型。如果返回类型是 string 并且非空，MobiLink 服务器将假设该字符串中包含有效的 SQL 语句，并将在统一数据库中执行该语句，如同执行正常的 SQL 语言同步脚本一样。如果某个方法通常返回字符串，但并不想基于其返回的内容对数据库执行 SQL 语句，则它可以返回空值。

调试 Java 类

MobiLink 提供了各种可以帮助您调试 Java 代码的信息和工具。本节介绍您可以在何处找到此信息以及如何利用这些功能。

MobiLink 服务器日志文件中的信息

MobiLink 服务器将消息写入消息日志文件中。服务器日志文件包含以下信息：

- Java 运行时环境。可以使用 -jrepath 选项在启动 MobiLink 服务器时请求特定的 JRE。缺省路径为随 SQL Anywhere 11 安装的 JRE 的路径。
- 标准 Java 类装载路径。如果您没有显式指定这些类，MobiLink 同步服务器会在调用 Java 虚拟机之前自动将其添加到类路径中。
- 所调用特定方法的完全限定名称。可以使用此信息来确认 MobiLink 服务器所调用的方法是否正确。
- 任何使用 Java 方法编写并输出到 java.lang.System.out 或 java.lang.System.err 的输出内容将被重定向到 MobiLink 服务器日志文件。

- 可以使用 mlsrv11 命令行选项 -verbose。

  请参见“-v 选项”一节第 97 页。

使用 Java 调试程序

可以使用标准的 Java 调试程序来调试 Java 类。在 mlsrv11 命令行上使用 -sl java 选项指定必需的参数。

请参见“-sl java 选项”一节第 87 页。

指定调试程序将使 Java 虚拟机暂停并等待来自 Java 调试程序的连接。

打印来自 Java 的信息

或者，也可以使用 Java.lang.System.err 或 Java.lang.System.Out，选择向 Java 方法添加一些语句，将信息输出到 MobiLink 消息日志。这样做可以帮助您跟踪类的进程和行为。

性能提示
采用此法打印信息可作为非常有用的监控工具，但不推荐在生产方案中使用。

同样的技术也可以用于记录任意同步信息或收集脚本使用情况的统计信息。

编写您自己的测试驱动程序

最好编写自己的驱动程序来测试 Java 类。这种方法非常有用，因为它可以将 Java 方法的操作与 MobiLink 系统的其它部分相隔离。

使用 Java 处理 MobiLink 服务器错误

如果扫描日志无法获取足够的信息，可以通过编程方式监控应用程序。例如，可以通过电子邮件发送某类消息。

您可以编写方法，用于接收代表输出到日志中每条错误或警告消息的类。这样可以帮助您监控和审计 MobiLink 服务器。

下面的代码为所有警告消息安装 LogListener 并将信息写到一个文件中。

```
class TestLogListener implements LogListener {
    FileOutputStream _out_file;

    public TestLogListener(FileOutputStream out_file) {
        _out_file = out_file;
    }

    public void messageLogged(ServerContext sc, LogMessage msg) {
        String  type;
        String  user;

        try {
            if (msg.getType() == LogMessage.ERROR) {
                type = "ERROR";
            } else if (msg.getType() == LogMessage.WARNING) {
                type = "WARNING";
            }
```

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

```
                else {
                    type = "UNKNOWN!!!";
                }

                user = msg.getUser();
                if (user == null) {
                    user = "NULL";
                }
                _out_file.write(("Caught msg type="
                    + type
                    + " user=" + user
                    + " text=" + msg.getText()
                    + "\n").getBytes()
                );
                _out_file.flush();
            }
            catch(Exception e) {
                // Print some error output to the MobiLink log.
                e.printStackTrace();
            }
        }
    }
```

以下代码注册 TestLogListener 来接收警告消息。可以从有权访问 ServerContext 的任何位置（例如类构造函数或同步脚本）调用此代码。

```
// ServerContext serv_context;
serv_context.addWarningListener(
      new MyLogListener(ll_out_file)
);
```

另请参见

- “addErrorListener 方法”一节第 532 页
- “removeErrorListener 方法”一节第 536 页
- “addWarningListener 方法”一节第 532 页
- “removeWarningListener 方法”一节第 537 页
- “LogListener 接口”一节第 524 页
- “LogMessage 类”一节第 525 页

用户定义的启动类

可以定义在服务器启动时自动装载的启动类。利用这一功能，您可以编写在 MobiLink 服务器启动 JVM 时（第一次同步之前）执行的 Java 代码。这意味着您可以在用户同步请求之前创建连接或缓存数据。

可使用 mlsrv11 -sl java 选项的 DMLStartClasses 选项来进行创建。例如，以下是 mlsrv11 命令行的部分内容。它将使 mycl1 和 mycl2 作为启动类装载。

```
-sl java(-DMLStartClasses=com.test.mycl1,com.test.mycl2)
```

将按所列顺序装载各个类。如果多次列出同一个类，则会创建多个实例。

所有启动类必须都是公共的，并且必须具有一个不接受任何参数或接受一个类型为 ianywhere.ml.script.ServerContext 的参数的公共构造函数。

装载的启动类的名称输出到 MobiLink 日志，显示以下消息："已装载 JAVA 启动类:*classname*"。

有关 Java 虚拟机选项的详细信息，请参见“-sl java 选项”一节第 87 页。

要查看服务器启动时所构造的启动类，请参见“getStartClassInstances 方法”一节第 535 页。

示例

下面是一个启动类模板。它启动一个处理事件并创建数据库连接的守护程序线程。（并非所有启动类都需要创建线程，但如果生成线程，则应是守护程序线程。）

```
import ianywhere.ml.script.*;
import java.sql.*;

public class StartTemplate extends
    Thread implements ShutdownListener {
    ServerContext   _sc;
    Connection      _conn;
    boolean         _exit_loop;

    public StartTemplate(ServerContext sc) throws SQLException {

        // Perform setup first so that an exception
        // causes MobiLink startup to fail.
        _sc = sc;

        // Create a connection for use later.
        _conn = _sc.makeConnection();

        _exit_loop = false;
        setDaemon(true);
        start();
    }

    public void run() {
        _sc.addShutdownListener(this);

        // run() cannot throw exceptions.
        try {
            handlerLoop();
            _conn.close();
            _conn = null;
        }
        catch(Exception e) {

            // Print some error output to the MobiLink log.
            e.printStackTrace();

            // This thread shuts down and so does not
            // need to be notified of shutdown.
            _sc.removeShutdownListener(this);

            // Ask server to shutdown so that this fatal
            // error is fixed.
            _sc.shutdown();
        }
        // Shortly after return this thread no longer exists.
        return;
    }

    // stop our event handler loop
    public void shutdownPerformed(ServerContext sc) {
        try {
            // Wait max 10 seconds for thread to die.
            join(10*1000);
```

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

```
            }
            catch(Exception e) {
                // Print some error output to the MobiLink log.
                e.printStackTrace();
            }
        }

        private void handlerLoop() throws InterruptedException {
            while (!_exit_loop) {
                // Handle events in this loop. Sleep not
                // needed, block on event queue.
                sleep(1 * 1000);
            }
        }
    }
```

Java 同步示例

Java 同步逻辑与 MobiLink 及公用 Java 类配合工作，使您能够灵活地使用 MobiLink 服务器部署应用程序。下节将以简单的示例介绍这个广泛适用的功能。

本节介绍了 Java 同步逻辑的一个工作示例。在试图使用此类或编写您自己的类之前，请使用如下项目清单来确保在开始编译类之前做好一切准备工作。

- 例如，使用伪代码等方法计划所需的功能。
- 创建数据库表和列的结构图。
- 确保已经在 MobiLink 系统表中指定了 Java 同步方法的语言类型和位置，从而针对 Java 同步配置好统一数据库。

  请参见“设置 Java 同步逻辑”一节第 495 页。
- 创建将在 Java 类运行过程中调用的 Java 类相关列表。
- 将您的 Java 类存储在 MobiLink 同步服务器类路径中的某个位置。

计划

此示例的 Java 同步逻辑指向为此示例的运行提供所需功能的相关 Java 文件和类。它将显示如何创建 CustEmpScripts 类。以及如何为连接设置同步上下文。最后，此示例还提供了实现如下功能的 Java 方法：

- 验证 MobiLink 用户
- 使用游标为每个数据库表执行下载和上载操作。

模式

要进行同步的表是 emp 和 cust。emp 表有三个列，分别为 emp_id、emp_name 和 manager。cust 表有三个列，分别为 cust_id、cust_name 和 emp_id。每个表中的所有列都进行同步。从统一数据库到远程数据库的映射保证在两个数据库中的表名和列名相同。另外在统一数据库中还添加了一个审计表。

Java 类文件

本示例中所使用的文件包含在 *Samples\MobiLink\JavaAuthentication* 目录中。

设置

以下代码将设置 Java 同步逻辑。其中 import 语句用于告知 Java 虚拟机所需文件的位置。public class 语句用于声明类。

```
// Use a package when you create your own script.
import ianywhere.ml.script.InOutInteger;
import ianywhere.ml.script.DBConnectionContext;
import ianywhere.ml.script.ServerContext;
import java.sql.*;

public class CustEmpScripts {
    // Context for this synchronization connection.
    DBConnectionContext _conn_context;
```

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

```
    // Same connection MobiLink uses for sync.
    // Do not commit or close this.
    Connection _sync_connection;
    Connection _audit_connection;

    //Get a user id given the user name. On audit connection.
    PreparedStatement _get_user_id_pstmt;

    // Add record of user logins added. On audit connection.
    PreparedStatement _insert_login_pstmt;

    // Prepared statement to add a record to the audit table.
    // On audit connection.
    PreparedStatement _insert_audit_pstmt;

    // ...
}
```

CustEmpScripts 构造函数为 authenticateUser 方法设置所有预备语句。同时也设置成员数据。

```
public CustEmpScripts(DBConnectionContext cc) throws SQLException {
    try {
        _conn_context = cc;
        _sync_connection = _conn_context.getConnection();

        ServerContext serv_context =
        _conn_context.getServerContext();
        _audit_connection = serv_context.makeConnection();

        // Get the prepared statements ready.
        _get_user_id_pstmt =
            _audit_connection.prepareStatement(
                "select user_id from ml_user where name = ?"
            );

        _insert_login_pstmt =
            _audit_connection.prepareStatement(
                "INSERT INTO login_added(ml_user, add_time) "
                    + "VALUES (?, { fn CONVERT({ fn NOW() }, SQL_VARCHAR) })"
            );

        _insert_audit_pstmt =
            _audit_connection.prepareStatement(
               "INSERT INTO login_audit(ml_user_id, audit_time, audit_action)
"
                    + "VALUES (?, { fn CONVERT({ fn NOW() },
SQL_VARCHAR) }, ?)"
            );
    }
    catch(SQLException e) {
        freeJDBCResources();
        throw e;
    }
    catch(Error e) {
        freeJDBCResources();
        throw e;
    }
}
```

如果没有调用 end_connection，finalize 方法将清理 JDBC 资源。该方法会调用 freeJDBCResources 方法，用以释放分配的内存并关闭审计连接。

```
protected void finalize() throws SQLException, Throwable {
    super.finalize();
    freeJDBCResources();
}

private void freeJDBCResources() throws SQLException {
    if (_get_user_id_pstmt != null) {
        _get_user_id_pstmt.close();
    }
    if (_insert_login_pstmt != null) {
        _insert_login_pstmt.close();
    }
    if (_insert_audit_pstmt != null) {
        _insert_audit_pstmt.close();
    }
    if (_audit_connection != null) {
        _audit_connection.close();
    }
    _conn_context       = null;
    _sync_connection    = null;
    _audit_connection   = null;
    _get_user_id_pstmt  = null;
    _insert_login_pstmt = null;
    _insert_audit_pstmt = null;
}
```

endConnection 方法会在资源不再需要时将其清理掉。

```
public void endConnection() throws SQLException {
    freeJDBCResources();
}
```

下面的 authenticateUser 方法将批准所有用户登录，并将用户信息记录到数据库表中。如果用户不在 ml_user 表中，则他们将被记录到 login_added 中。如果用户 ID 在 ml_user 中，则他们将被记录到 login_audit 中。在真实的系统中不会忽略 user_password，但为简单起见，本示例将批准所有用户。如果任何数据库操作因异常而失败，endConnection 方法都将抛出 SQLException。

```
public void authenticateUser(
    InOutInteger authentication_status,
    String user_name) throws SQLException
{

    boolean new_user;
    int user_id;

    // Get ml_user id.
    _get_user_id_pstmt.setString(1, user_name);

    ResultSet user_id_rs =
    _get_user_id_pstmt.executeQuery();
    new_user = !user_id_rs.next();
    if (!new_user) {
        user_id = user_id_rs.getInt(1);
    }
    else {
        user_id = 0;
    }

    user_id_rs.close();
    user_id_rs = null;

    // In this tutorial always allow the login.
```

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

```
        authentication_status.setValue(1000);

        if (new_user) {
            _insert_login_pstmt.setString(1, user_name);
            _insert_login_pstmt.executeUpdate();
            java.lang.System.out.println("user: " + user_name + " added. ");
        }
        else {
            _insert_audit_pstmt.setInt(1, user_id);
            _insert_audit_pstmt.setString(2, "LOGIN ALLOWED");
            _insert_audit_pstmt.executeUpdate();
        }
        _audit_connection.commit();
        return;
    }
```

以下方法使用 SQL 代码充当数据库表上的游标。既然它们是游标脚本，因此必须返回一个 SQL 字符串。

```
    public static String empUploadInsertStmt() {
        return("INSERT INTO emp(emp_id, emp_name) VALUES(?, ?)");
    }

    public static String empUploadDeleteStmt() {
        return("DELETE FROM emp WHERE emp_id = ?");
    }

    public static String empUploadUpdateStmt() {
        return("UPDATE emp SET emp_name = ? WHERE emp_id = ?");
    }

    public static String empDownloadCursor() {
        return("SELECT emp_id, emp_name FROM emp");
    }

    public static String custUploadInsertStmt() {
        return("INSERT INTO cust(cust_id, emp_id, cust_name) VALUES (?, ?, ?)");
    }

    public static String custUploadDeleteStmt() {
        return("DELETE FROM cust WHERE cust_id = ?");
    }

    public static String custUploadUpdateStmt() {
        return("UPDATE cust SET emp_id = ?, cust_name = ? WHERE cust_id = ?");
    }

    public static String custDownloadCursor() {
        return("SELECT cust_id, emp_id, cust_name FROM cust");
    }
```

使用以下命令编译代码：

```
javac -cp %sqlany11%\java\mlscript.jar CustEmpScripts.java
```

以 CustEmpScripts.class 在类路径中的位置运行 MobiLink 服务器。下面是命令行的一部分：

```
mlsrv11 ... -sl java (-cp <class_location>)
```

用于 Java 的 MobiLink 服务器 API 参考

本节介绍 MobiLink Java 接口和类，及其相关方法和构造函数。要使用这些类，请引用位于 *install-dir\java* 中的 *mlscript.jar* 程序集。

DBConnectionContext 接口

语法

public **ianywhere.ml.script.DBConnectionContext**

注释

用于获取和访问当前数据库连接相关信息的接口。DBConnectionContext 实例会传递给包含脚本的类的构造函数。如果后台线程或在连接生存期以外需要上下文，请使用 ServerContext 类。

另请参见

- “构造函数”一节第 498 页
- “ServerContext 接口”一节第 530 页

成员

ianywhere.ml.script.DBConnectionContext 的所有成员，包括所有继承的成员。

- “getConnection 方法”一节第 509 页
- “getDownloadData 方法”一节第 509 页
- “getProperties 方法”一节第 510 页
- “getRemoteID 方法”一节第 511 页
- “getServerContext 方法”一节第 511 页
- “getVersion 方法”一节第 512 页

示例

以下示例说明应如何创建用于同步脚本中的类级 DBConnectionContext 实例。DBConnectionContext getConnection 方法会获取表示当前 MobiLink 统一数据库连接的 java.sql.Connection 实例。

```
import ianywhere.ml.script.*;
import java.io.*;
import java.sql.*;

public class OrderProcessor {
    DBConnectionContext _cc;

    public OrderProcessor(DBConnectionContext cc) {
        _cc = cc;
    }

    // The method used for the handle_DownloadData event.
    public void HandleEvent() throws SQLException {
        java.sql.Connection my_connection = _cc.getConnection();
        // ...
```

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

```
        }
        // ...
    }
```

小心
DBConnectionContext 实例不能在调用 Java 代码的线程之外使用。

getConnection 方法

语法

public java.sql.Connection **getConnection( )**
throws java.sql.SQLException

注释

将现有 MobiLink 统一数据库连接作为 JDBC 连接返回。由此方法返回的 java.sql.Connection 对象所表示的连接即为 MobiLink 服务器用于执行 SQL 脚本的连接。

切勿以任何将会影响 MobiLink 服务使用该连接的方式来提交、关闭或变更该连接。返回的连接仅在基础 MobiLink 连接生存期内有效。在为连接调用了 end_connection 事件之后，不要再使用该连接。

如果将现有连接作为 JDBC 连接捆绑时出错，则会抛出 java.sql.SQLException。

如果需要具有完全访问权限的服务器连接，请使用 ServerContext.makeConnection()。

返回值

作为 JDBC 连接的现有 MobiLink 统一数据库连接。

示例

请参见“DBConnectionContext 接口”一节第 508 页。

getDownloadData 方法

语法

public DownloadData **getDownloadData()**

注释

返回当前同步的 DownloadData 实例。使用 DownloadData 类可创建直接行处理下载。

返回值

当前同步的 DownloadData 实例。

另请参见

- “DownloadData 接口”一节第 513 页
- “直接行处理”第 609 页

示例

以下示例告诉您如何使用 DBConnectionContext getDownloadData 方法来获取当前同步的 DownloadData 实例。

注意
此示例假定您拥有一个名为 _cc 的 DBConnectionContext 实例。

```
// The method used for the handle_DownloadData event.
public void HandleDownload() throws SQLException {
    // get the DownloadData for the current synchronization
    DownloadData my_dd = _cc.getDownloadData();

    // ...
}

// ...
```

getProperties 方法

语法

public java.util.Properties **getProperties( )**

注释

基于此连接的脚本版本返回此连接的属性。属性存储在 ml_property 表中。

有关 java.util.Properties 的详细信息，请查阅 Java SDK 文档。

返回值

此连接的属性。

另请参见

- “ml_property”一节第 670 页
- “ml_add_property 系统过程”一节第 637 页

示例

以下示例显示如何输出 DBConnectionContext 的属性。

注意
此示例假定您拥有一个名为 _cc 的 DBConnectionContext 实例。

```
// The method used to output the connection properties.
public void outputProperties() {
    // output the Properties for the current synchronization
```

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

```
        java.util.Properties properties = _cc.getProperties();
        System.out.println(properties.toString());
    }
```

getRemoteID 方法

语法

public java.lang.String **getRemoteID( )**

注释

返回当前正在此连接上同步的数据库的远程 ID。如果远程数据库早于版本 10，则将返回 MobiLink 用户名。

返回值

远程 ID。

另请参见

- “远程 ID”一节 《MobiLink - 客户端管理》

示例

以下示例显示如何输出 DBConnectionContext 的远程 ID。

注意
此示例假定您拥有一个名为 _cc 的 DBConnectionContext 实例。

```
// The method used to output the remote ID.
public void outputRemoteID() {
    // output the Remote ID for the current synchronization
    String remoteID = _cc.getRemoteID();
    System.out.println(remoteID);
}
```

getServerContext 方法

语法

public ServerContext **getServerContext( )**

注释

为 MobiLink 服务器返回 ServerContext。

返回值

此 MobiLink 服务器的 ServerContext。

另请参见

- “ServerContext 接口”一节第 530 页

示例

以下示例显示如何获取 DBConnectionContext 的 ServerContext 实例并关闭服务器。

注意
此示例假定您拥有一个名为 _cc 的 DBConnectionContext 实例。

```
// A method that uses an instance of the ServerContext to shut down the
server
public void shutDownServer() {
    ServerContext context = _cc.getServerContext();
    context.shutdown();
}
```

getVersion 方法

语法

public java.lang.String **getVersion( )**

注释

返回脚本版本字符串。

返回值

脚本版本字符串。

另请参见

- “ml_property”一节第 670 页
- “ml_add_property 系统过程”一节第 637 页

示例

以下示例显示如何获取脚本版本并使用它作决定。

注意
此示例假定您拥有一个名为 _cc 的 DBConnectionContext 实例。

```
// A method that uses the script version
public void handleEvent() {
    // ...

    String version = _cc.getVersion();
    if (version.equals("My Version 1")) {
        // ...
    } else if (version.equals("My Version 2")) {
        // ...
    }
}
```

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

```
// ...
```

DownloadData 接口

语法

public **ianywhere.ml.script.DownloadData**

注释

封装用于直接行处理的下载数据操作。要获得 DownloadData 实例，请使用 DBConnectionContext getDownloadData 方法。

使用 DownloadData.getDownloadTables 和 getDownloadTableByName 方法可返回 DownloadTableData 实例。

可通过 DBConnectionContext 使用此下载数据。在 begin_synchronization 事件之前或 end_download 事件之后无法访问下载数据。在仅上载同步中也无法访问 DownloadData。

另请参见

- “DownloadTableData 接口”一节第 515 页
- “handle_DownloadData 连接事件”一节第 410 页
- DBConnectionContext “getDownloadData 方法”一节第 509 页
- “直接行处理”第 609 页

成员

ianywhere.ml.script.DownloadData 的所有成员，包括所有继承的成员。

- “getDownloadTableByName 方法”一节第 514 页
- “getDownloadTables 方法”一节第 514 页

示例

以下示例告诉您如何使用 DBConnectionContext getDownloadData 方法来获取当前同步的 DownloadData 实例。

```
DBConnectionContext _cc;

// Your class constructor.
public OrderProcessor(DBConnectionContext cc) {
    _cc = cc;
}

// The method used for the handle_DownloadData event.
public void handleDownload() throws SQLException {
    // Get the DownloadData for the current synchronization.
    DownloadData my_dd = _cc.getDownloadData();

    // ...
}
```

getDownloadTableByName 方法

语法

public DownloadTableData getDownloadTableByName(
string *table-name*);

注释

为此同步获取指定的下载表。如果此同步中不存在具有指定名称的表，则会返回空值。

参数

- **table_name** 想要获取其下载数据的表的名称。

返回值

返回表示指定表的 DownloadTableData 实例，或者如果当前同步中不存在具有指定名称的表，则返回空值。

另请参见

- “DownloadData 接口”一节第 513 页
- “DownloadTableData 接口”一节第 515 页
- “DBConnectionContext 接口”一节第 508 页
- “直接行处理”第 609 页

示例

以下示例使用 getDownloadTableByName 方法来返回 remoteOrders 表的 DownloadTableData 实例。

> **注意**
> 此示例假定您拥有一个名为 _cc 的 DBConnectionContext 实例。

```
// The method used for the handle_DownloadData event.
public void handleDownload() throws SQLException {
    // Get the DownloadData for the current synchronization.
    DownloadData my_dd = _cc.getDownloadData();

    // Get the DownloadTableData for the remoteOrders table.
    DownloadTableData my_download_table =
my_dd.getDownloadTableByName("remoteOrders");

    // ...
}
```

getDownloadTables 方法

语法

public DownloadTableData[ ] **getDownloadTables( )**

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

注释

获取包含此同步中要下载数据的所有表的数组。对此表执行的操作会发送到远程数据库。

返回值

包含当前同步中 DownloadTableData 对象的数组。数组中表的顺序与远程数据库的上载顺序相同。

另请参见

- “DownloadData 接口”一节第 513 页
- “DownloadTableData 接口”一节第 515 页
- “DBConnectionContext 接口”一节第 508 页
- “直接行处理”第 609 页

示例

以下示例使用 DownloadData.getDownloadTables 方法来获取包含当前同步中 DownloadTableData 对象的一个数组。本示例假定您拥有一个名为 _cc 的 DBConnectionContext 实例。

```
// The method used for the handle_DownloadData event.
public void handleDownload() throws SQLException {
    // Get the DownloadData for the current synchronization.
    DownloadData my_dd = _cc.getDownloadData();

    // Get an array of tables to set download operations.
    DownloadTableData[] download_tables = my_dd.getDownloadTables();

    // Get the first table in the DownloadTableData array.
    DownloadTableData my_download_table = download_tables[0];

    // ...
}
```

DownloadTableData 接口

语法

public **ianywhere.ml.script.DownloadTableData**

注释

封装用于 MobiLink 直接下载的表操作。使用此接口可设置将下载到客户端的数据操作。要获取当前同步的 DownloadTableData 实例，请使用 DownloadData 接口。可以分别使用 DownloadTableData.getUpsertPreparedStatement 和 getDeletePreparedStatement 方法来获取用于插入和更新操作以及删除操作的 Java 预准备语句。java.sql.PreparedStatement.executeUpdate 方法可以注册要下载的操作。

> **注意**
> 对于插入和更新预准备语句，必须设置所有列值。对于删除操作，则需设置主键值。
>
> delete 和 upsert 预准备语句不能同时打开。

有关 java.sql.PreparedStatement 的详细信息，请参考 Java SDK 文档。

另请参见

- “DownloadData 接口”一节第 513 页
- “handle_DownloadData 连接事件”一节第 410 页
- “直接行处理”第 609 页

成员

ianywhere.ml.script.DownloadTableData 的所有成员，包括所有继承的成员。

- “getDeletePreparedStatement 方法”一节第 517 页
- “getUpsertPreparedStatement 方法”一节第 518 页
- “getName 方法”一节第 519 页
- “getMetaData 方法”一节第 520 页
- “getLastDownloadTime 方法”一节第 521 页

示例

假定使用 MobiLink 客户端数据库中名为 remoteOrders 的表。

```
CREATE TABLE remoteOrders (
    pk INT NOT NULL,
    col1 VARCHAR(200),
    PRIMARY KEY (pk)
);
```

以下示例使用 DownloadData.getDownloadTableByName 方法返回表示 remoteOrders 表的 DownloadTableData 实例。

注意
此示例假定您拥有一个名为 _cc 的 DBConnectionContext 实例。

```
// The method used for the handle_DownloadData event
public void handleDownload() throws SQLException {

    // Get the DownloadData for the current synchronization.
    DownloadData my_dd = _cc.getDownloadData();

    // Get the DownloadTableData for the remoteOrders table.
    DownloadTableData td = my_dd.getDownloadTableByName("remoteOrders");

    // User defined-methods to set download operations.
    setDownloadInserts(td);
    setDownloadDeletes(td);

    // ...
}
```

在本例中，setDownloadInserts 方法使用 DownloadTableData.getUpsertPreparedStatement 来为想要插入或更新的行获取预准备语句。PreparedStatement.setInt 和 PreparedStatement.setString 方法会设置要插入远程数据库中的列值。

```
void setDownloadInserts(DownloadTableData td) {
    java.sql.PreparedStatement insert_ps = td.getUpsertPreparedStatement();

    // The following method calls are the same as the following SQL
statement:
```

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

```
        // INSERT INTO remoteOrders(pk, col1) values(2300, "truck");
        insert_ps.setInt(1, 2300);
        insert_ps.setString(2, "truck");

        int update_result = insert_ps.executeUpdate();
        if (update_result == 0) {
            // Insert was filtered because it was uploaded
            // in the same synchronization.
        }
        else {
            // Insert was not filtered.
        }
    }
```

setDownloadDeletes 方法使用 DownloadTableData.getDeletePreparedStatement 来为想要删除的行获取预准备语句。java.sql.PreparedStatement.setInt 方法用于设置要在远程数据库中删除的行的主键值，java.sql.PreparedStatement.executeUpdate 方法用于注册要下载的行值。

```
    void setDownloadDeletes(DownloadTableData td) {
        java.sql.PreparedStatement delete_ps = td.getDeletePreparedStatement();

        // The following method calls are the same as the following SQL
    statement:
        // DELETE FROM remoteOrders  where pk=2300;
        delete_ps.setInt(1, 2300);
        delete_ps.executeUpdate();
    }
```

getDeletePreparedStatement 方法

语法

public java.sql.PreparedStatement **getDeletePreparedStatement( )** throws **SQLException**

注释

返回允许用户向下载中添加删除操作的 java.sql.PreparedStatement 实例。此预准备语句将应用于下载表，表中每个主键列在语句中对应一个参数。

此预准备语句将应用于 DownloadTableData 实例，表中每个主键列在语句中对应一个参数。

要在下载中包括删除操作，请在 java.sql.PreparedStatement 中设置所有列，然后调用 java.sql.PreparedStatement.executeUpdate 方法。

> **注意**
> 您必须为下载删除操作设置所有主键值。

返回值

用于将删除操作添加到下载中的 java.sql.PreparedStatement 实例。

异常

- **SQLException**　如果在检索删除 java.sql.PreparedStatement 实例时出现问题，则将其抛出。

另请参见

- “DownloadTableData 接口”一节第 515 页
- “直接行处理”第 609 页

示例

在以下示例中，setDownloadDeletes 方法使用 DownloadTableData.getDeletePreparedStatement 来为想要删除的行获取预准备语句。java.sql.PreparedStatement.setInt 方法用于设置要在远程数据库中删除的行的主键值，java.sql.PreparedStatement.executeUpdate 方法用于设置下载的行值。

```
void setDownloadDeletes(DownloadTableData td) {
    java.sql.PreparedStatement delete_ps = td.getDeletePreparedStatement();
    // This is the same as executing the following SQL statement:
    // DELETE FROM remoteOrders  where pk=2300;
    delete_ps.setInt(1, 2300);
    delete_ps.executeUpdate();
    delete_ps.close();
}
```

getUpsertPreparedStatement 方法

语法

public java.sql.PreparedStatement **getUpsertPreparedStatement( )** throws **SQLException**

注释

返回允许用户向同步下载中添加 upsert（插入或更新）操作的 java.sql.PreparedStatement 实例。此预准备语句将应用于 DownloadTableData 实例，表中每列在语句中对应一个参数。

要在下载中包括插入或更新操作，请在 java.sql.PreparedStatement 中设置所有列值，然后调用 java.sql.PreparedStatement.executeUpdate 方法。如果插入或更新操作被过滤，则对预准备语句调用 java.sql.PreparedStatement.executeUpdate 将返回 0；而如果操作未被过滤，则将返回 1。操作如果是在同一同步中上载的，则会被过滤。

> **注意**
> 您必须为插入和更新操作设置所有列值。

返回值

用于将插入和更新操作添加到下载中的 java.sql.PreparedStatement 实例。

异常

- **SQLException**　如果在检索 upsert java.sql.PreparedStatement 实例时出现问题，则将其抛出。

另请参见

- “DownloadTableData 接口”一节第 515 页
- “直接行处理”第 609 页

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

示例

在以下示例中，setDownloadInserts 方法使用 DownloadTableData.getUpsertPreparedStatement 来为要插入或更新的行获取预准备语句。java.sql.PreparedStatement.setInt 和 PreparedStatement.setString 方法会设置列值，PreparedStatement.executeUpdate 方法会设置下载的行值。

```
void setDownloadInserts(DownloadTableData td) {
    java.sql.PreparedStatement insert_ps = td.getUpsertPreparedStatement();

    // This is the same as executing the following SQL statement:
    // INSERT INTO remoteOrders(pk, col1) VALUES (2300, "truck");
    insert_ps.setInt(1, 2300);
    insert_ps.setString(2, "truck");

    int update_result = insert_ps.executeUpdate();
    if (update_result == 0) {
        // Insert was filtered because it was uploaded
        // in the same synchronization.
    }
    else {
        // Insert was not filtered.
    }
    insert_ps.close();
}
```

getName 方法

语法

public java.lang.String **getName( )**

注释

返回 DownloadTableData 实例的表名。您也可以使用由 DownloadTableData.getMetaData 方法返回的 java.sql.ResultSetMetaData 实例来访问表名。

返回值

DownloadTableData 实例的表名。

另请参见

- “DownloadTableData 接口”一节第 515 页
- DownloadTableData “getMetaData 方法”一节第 520 页
- “直接行处理”第 609 页

示例

以下示例显示如何输出 DownloadTableData 实例的表名。

> **注意**
> 此示例假定您拥有一个名为 _cc 的 DBConnectionContext 实例。

```
// The method used for the handle_DownloadData event
public void handleDownload() throws SQLException {
```

```
        // Get the DownloadData for the current synchronization.
        DownloadData my_dd = _cc.getDownloadData();

        // Get the DownloadTableData for the remoteOrders table.
        DownloadTableData td = my_dd.getDownloadTableByName("remoteOrders");

        // Print the table name to standard output (remoteOrders)
        System.out.println(td.getName());

        // User defined-methods to set download operations.
        setDownloadInserts(td);
        setDownloadDeletes(td);

        // ...
    }
```

getMetaData 方法

语法

public java.sql.ResultSetMetaData **getMetaData( )**

注释

获取 DownloadTableData 实例的元数据。元数据是标准的 java.sql.ResultSetMetaData 对象。

如果希望元数据中包含列名信息，可在客户端指定在上载时发送列名。

有关 java.sql.ResultSetMetaData 的详细信息，请参考 Java SDK 文档。

返回值

DownloadTableData 实例的元数据。

另请参见

- “DownloadTableData 接口”一节第 515 页
- “直接行处理”第 609 页
- SQL Anywhere 客户端： “SendColumnNames (scn) 扩展选项”一节 《MobiLink - 客户端管理》
- UltraLite： “Send Column Names 同步参数”一节 《UltraLite - 数据库管理和参考》

示例

以下示例显示如何获取用于 DownloadTableData 实例的查询的列数。

```
import java.sql.ResultSetMetaData;

// The method used to return the number of columns in a DownloadTableData
instance query
public int getNumColumns(DownloadTableData td) {
    ResultSetMetaData rsmd = td.getMetaData();
    return rsmd.getColumnCount();
}
```

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

getLastDownloadTime 方法

语法

public java.sql.Timestamp **getLastDownloadTime( )**

注释

返回此表的上次下载时间。传递给几个表格下载事件的上次下载时间是相同的。

上次下载时间对于生成用于某个特定同步的表下载数据很有用。

返回值

此下载表的上次下载时间。

另请参见

- “DownloadTableData 接口”一节第 515 页
- “直接行处理”第 609 页

示例

以下示例为代码片段，它以使用上次下载时间的插入项填充下载的表。请注意，此示例假定您拥有一个名为 _cc 的 DBConnectionContext 实例。

```
// The method used for the handle_DownloadData event
public void handleDownload() throws SQLException {

    // Get the DownloadData for the current synchronization.
    DownloadData my_dd = _cc.getDownloadData();

    // Get the DownloadTableData for the remoteOrders table.
    DownloadTableData td = my_dd.getDownloadTableByName("remoteOrders");

    // get the inserts given a last download time
    ResultSet inserts_rs  =
makeInsertsFromTimestamp(td.getLastDownloadTime());

    // fill the DownloadTableData using the inserts resultset.
    setDownloadInsertsFromRS(td, inserts_rs);
    inserts_rs.close();

    // ...
}
```

InOutInteger 接口

语法

public **ianywhere.ml.script.InOutInteger**

注释

被传递给方法，以使传递给 SQL 脚本的输入/输出参数生效。

成员

ianywhere.ml.script.InOutInteger 的所有成员，包括所有继承的成员。

- “getValue 方法”一节第 522 页
- “setValue 方法”一节第 523 页

示例

以下对 MobiLink 系统过程的调用在同步脚本版本 ver1 时将名为 handleError 的 Java 方法注册为 handle_error 连接事件的脚本。

```
CALL ml_add_java_connection_script(
    'ver1',
    'handle_error',
    'ExamplePackage.ExampleClass.handleError'
)
```

以下是 Java 方法 handleError 示例。它根据传入的数据处理错误。同时还确定结果错误代码。

```
public String handleError(
    ianywhere.ml.script.InOutInteger actionCode,
    int errorCode,
    String errorMessage,
    String user,
    String table)
{

    int new_ac;
    if (user == null) {
        new_ac = handleNonSyncError(errorCode, errorMessage);
    } else if (table == null) {
        new_ac = handleConnectionError(errorCode, errorMessage, user);
    }
    else {
        new_ac = handleTableError(errorCode, errorMessage, user, table);
    }

    // Keep the most serious action code.
    if (actionCode.getValue() < new_ac) {
        actionCode.setValue(new_ac);
    }
}
```

getValue 方法

语法

public int **getValue( )**

注释

返回此整型参数的值。

返回值

此整型参数的值。

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

示例

请参见： “InOutInteger 接口”一节第 521 页.

setValue 方法

语法

public void **setValue(** int *new_value* **)**

注释

设置此整型参数的值。

参数

- **new_value** 此整数接受的值。

示例

请参见： “InOutInteger 接口”一节第 521 页.

InOutString 接口

语法

public **ianywhere.ml.script.InOutString**

注释

被传递给方法，以使传递给 SQL 脚本的输入/输出参数生效。

成员

ianywhere.ml.script.InOutString 的所有成员，包括所有继承的成员。

- “getValue 方法”一节第 524 页
- “setValue 方法”一节第 524 页

示例

以下对 MobiLink 系统过程的调用在同步脚本版本 ver1 时将名为 modifyUser 的 Java 方法注册为 modify_user 连接事件的脚本。

```
CALL ml_add_java_connection_script(
    'ver1',
    'modify_user',
    'ExamplePackage.ExampleClass.modifyUser'
)
```

以下是 Java 方法 modifyUser 示例。它从数据库获取用户 ID，然后使用该 ID 设置用户名。

```
public String modifyUser(InOutString io_user_name) throws SQLException {
    Statement uid_select = curConn.createStatement();
```

```
        ResultSet uid_result = uid_select.executeQuery(
            "SELECT rep_id FROM SalesRep WHERE name = '"
                + io_user_name.getValue() + "' "
        );
        uid_result.next();
        io_user_name.setValue(java.lang.Integer.toString(uid_result.getInt(1));
        uid_result.close();
        uid_select.close();
        return (null);
    }
```

getValue 方法

语法

public java.lang.String **getValue( )**

注释

返回此字符串参数的值。

返回值

此字符串参数的值。

示例

请参见： “InOutString 接口”一节第 523 页

setValue 方法

语法

public void **setValue(** java.lang.String *new_value* **)**

注释

设置此字符串参数的值。

参数

- **new_value** 此字符串接受的值。

示例

请参见： “InOutString 接口”一节第 523 页

LogListener 接口

语法

public **ianywhere.ml.script.LogListener**

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

注释

监听器接口用来捕获输出到日志中的消息。

另请参见

- “使用 Java 处理 MobiLink 服务器错误”一节第 500 页

成员

ianywhere.ml.script.LogListener 的所有成员，包括所有继承的成员。

- “messageLogged 方法”一节第 525 页

示例

请参见： “使用 Java 处理 MobiLink 服务器错误”一节第 500 页

messageLogged 方法

语法

```
public void messageLogged(
  ServerContext sc,
  LogMessage message )
```

注释

将消息输出到日志中时调用。

参数

- **sc** 正在打印消息的服务器的上下文。
- **message** 已发送到 MobiLink 日志中的 LogMessage。

示例

请参见： “使用 Java 处理 MobiLink 服务器错误”一节第 500 页

LogMessage 类

语法

public **ianywhere.ml.script.LogMessage**

注释

保存与日志消息相关联的数据。

扩展 java.lang.Object。

另请参见

- “使用 Java 处理 MobiLink 服务器错误”一节第 500 页

成员

ianywhere.ml.script.LogMessage 的所有成员，包括所有继承的成员。

- “ERROR 变量”一节第 528 页
- “INFO 变量”一节第 528 页
- “WARNING 变量”一节第 528 页
- “getType 方法”一节第 529 页
- “getUser 方法”一节第 529 页
- “getText 方法”一节第 529 页

示例

下面的示例为所有警告、错误和信息消息安装 LogListener，然后将信息写入一个文件中。以下代码为所有警告消息安装 LogListener。

```
class WarningLogListener implements LogListener {
    FileOutputStream _outFile;

    public WarningLogListener(FileOutputStream outFile) {
        _outFile = outFile;
    }

    public void messageLogged(ServerContext sc, LogMessage msg) {
        String user;

        try {
            if (msg.getType() != LogMessage.WARNING) {
                //this class deals exclusively with warnings
                return;
            }
            user = msg.getUser();

            if (user == null) {
                user = "NULL";
            }

            _outFile.write(("Caught warning"
                + " user=" + user
                + " text=" + msg.getText()
                + "\n").getBytes()
            );
            _outFile.flush();
        }
        catch(Exception e) {
            // Print some error output to the MobiLink log.
            e.printStackTrace();
        }
    }
}
```

以下代码为所有错误消息安装 LogListener。

```
class ErrorLogListener implements LogListener {
    FileOutputStream _outFile;
```

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

```
    public ErrorLogListener(FileOutputStream outFile) {
        _outFile = outFile;
    }

    public void messageLogged(ServerContext sc, LogMessage msg) {
        String user;

        try {
            if (msg.getType() != LogMessage.ERROR) {
                //this class deals exclusively with errors
                return;
            }

            user = msg.getUser();
            if (user == null) {
                user = "NULL";
            }

            _outFile.write(("Caught error"
                    + " user=" + user
                    + " text=" + msg.getText()
                    + "\n").getBytes()
            );
            _outFile.flush();
        }
        catch(Exception e) {
            // Print some error output to the MobiLink log.
            e.printStackTrace();
        }
    }
}
```

以下代码为所有信息消息安装 LogListener。

```
class InfoLogListener implements LogListener {
    FileOutputStream _outFile;

    public InfoLogListener(FileOutputStream outFile) {
        _outFile = outFile;
    }

    public void messageLogged(ServerContext sc, LogMessage msg) {
        String user;

        try {
            if (msg.getType() != LogMessage.ERROR) {
                // this class deals exclusively with info
                return;
            }

            user = msg.getUser();
            if (user == null) {
                user = "NULL";
            }

            _outFile.write(("Caught info"
                    + " user=" + user
                    + " text=" + msg.getText()
                    + "\n").getBytes()
            );
            _outFile.flush();
        }
        catch(Exception e) {
            // Print some error output to the MobiLink log.
```

```
                e.printStackTrace();
            }
        }
    }
```

以下代码注册 WarningLogListener、ErrorLogListener 和 InfoLogListener，以分别接收警告、错误和信息消息。可以从有权访问 ServerContext 的任何位置（例如类构造函数或同步脚本）调用此代码。

```
// ServerContext serv_context;
// FileOutputStream outFile
serv_context.addWarningListener(new WarningLogListener(outFile));
serv_context.addErrorListener(new ErrorLogListener(outFile));
serv_context.addInfoListener(new InfoLogListener(outFile));
```

ERROR 变量

语法

int **ERROR**

注释

日志消息属于错误。

示例

请参见： “LogMessage 类”一节第 525 页.

INFO 变量

语法

int **INFO**

注释

消息日志显示信息。

示例

请参见： “addInfoListener 方法”一节第 530 页.

WARNING 变量

语法

int **WARNING**

注释

日志消息属于警告。

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

示例

请参见： “LogMessage 类”一节第 525 页.

getType 方法

语法

public int **getType( )**

注释

获取此消息的类型。

返回值

此消息的类型可以是 LogMessage.ERROR、LogMessage.INFO 或 LogMessage.WARNING。

示例

请参见： “LogMessage 类”一节第 525 页.

getUser 方法

语法

public java.lang.String **getUser( )**

注释

用于此消息用户的取值函数。如果此消息没有用户，则用户为空。

返回值

与此消息关联的用户。

示例

请参见： “LogMessage 类”一节第 525 页.

getText 方法

语法

public java.lang.String **getText( )**

注释

获取消息文本。

返回值

此消息的正文文本。

示例

请参见： “LogMessage 类”一节第 525 页.

ServerContext 接口

语法

public **ianywhere.ml.script.ServerContext**

注释

MobiLink 服务器工作期间出现的所有上下文的实例化。上下文可以作为静态数据保存，并在后台线程中使用。它在由 MobiLink 调用的 Java 虚拟机的工作期间有效。

要访问 ServerContext 实例，请使用 DBConnectionContext.getServerContext 方法。

成员

ianywhere.ml.script.ServerContext 的所有成员，包括所有继承的成员。

- “addInfoListener 方法”一节第 530 页
- “addErrorListener 方法”一节第 532 页
- “addShutdownListener 方法”一节第 532 页
- “addWarningListener 方法”一节第 532 页
- “getProperties 方法”一节第 533 页
- “getPropertiesByVersion 方法”一节第 534 页
- “getPropertySetNames 方法”一节第 534 页
- “getStartClassInstances 方法”一节第 535 页
- “makeConnection 方法”一节第 535 页
- “removeErrorListener 方法”一节第 536 页
- “removeInfoListener 方法”一节第 536 页
- “removeShutdownListener 方法”一节第 537 页
- “removeWarningListener 方法”一节第 537 页
- “shutdown 方法”一节第 537 页

addInfoListener 方法

语法

public void **addInfoListener(** LogListener *l* **)**

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

注释

从监听器列表添加指定的 LogListener，在输出信息时接收通知。将调用方法 LogListener.messageLogged (ianywhere.ml.script.ServerContext)。

参数

- **II** 要通知的 LogListener。

示例

下面代码注册了 MyLogListener 类型的监听器，以接收信息消息通知。

```
// ServerContext serv_context;
serv_context.addInfoListener(new MyLogListener(ll_out_file));

// The following code shows an example of processing those messages:
class MyLogListener implements LogListener {
    FileOutputStream _out_file;
    public TestLogListener(FileOutputStream out_file) {
        _out_file = out_file;
    }

    public void messageLogged(ServerContext sc, LogMessage msg) {
        String type;
        String user;

        try {
            if (msg.getType() == LogMessage.ERROR) {
                type = "ERROR";
            } else if (msg.getType() == LogMessage.WARNING) {
                type = "WARNING";
            } else if (msg.getType() == LogMessage.INFO) {
                type = "INFO";
            } else {
                type = "UNKNOWN!!!";
            }

            user = msg.getUser();
            if (user == null) {
                user = "NULL";
            }
            _out_file.write(("Caught msg type="
                + type
                + " user=" + user
                + " text=" +msg.getText()
                + "\n").getBytes()
            );
            _out_file.flush();
        }
        catch(Exception e) {

            // if we print the exception from processing an info message,
            // we may recurse indefinitely
            if (msg.getType() != LogMessage.ERROR) {
                // Print some error output to the MobiLink log.
                e.printStackTrace();
            }
        }
    }
}
```

addErrorListener 方法

语法

public void **addErrorListener(** LogListener *ll* **)**

注释

添加指定的 LogListener 以便在输出错误时接收通知。

输出错误时，将调用以下方法：LogListener.messageLogged(ianywhere.ml.script.ServerContext, ianywhere.ml.script.LogMessage)。

参数

- **ll** 要通知的 LogListener。

另请参见

- “messageLogged 方法”一节第 525 页

示例

请参见： “LogMessage 类”一节第 525 页.

addShutdownListener 方法

语法

public void **addShutdownListener(** ShutdownListener *sl* **)**

注释

添加将在服务器上下文被取消前接收通知的指定 ShutdownListener。服务器关闭时，将调用 ShutdownListener.shutdownPerformed (ianywhere.ml.script.ServerContext) 方法。

参数

- **sl** 服务器关闭时要收到通知的 ShutdownListener。

示例

请参见： “ShutdownListener 接口”一节第 539 页.

addWarningListener 方法

语法

public void **addWarningListener(** LogListener *ll* **)**

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

注释

添加指定的 LogListener 以便在输出警告时接收通知。

调用以下方法：LogListener.messageLogged(ianywhere.ml.script.ServerContext, ianywhere.ml.script.LogMessage)。

参数

- **ll** 要通知的 LogListener。

示例

请参见： “LogMessage 类”一节第 525 页.

getProperties 方法

语法

```
public java.util.Properties getProperties(
  java.lang.String component,
  java.lang.String set )
```

注释

返回与给定组件和属性集关联的一组属性。这些属性存储在 MobiLink 系统表 ml_property 中。

参数

- **component** 组件。
- **set** 属性集。

返回值

一组属性（可能为空）。

另请参见

- “ml_property”一节第 670 页
- “ml_add_property 系统过程”一节第 637 页

示例

以下代码列出了所有 ServerContext 的属性。

```
import java.util.*;
// ServerContext serverContext;
// PrintStream out
Properties prop = serverContext.getProperties();
prop.list(out);
```

getPropertiesByVersion 方法

语法

public java.util.Properties **getPropertiesByVersion(** java.lang.String *script_version* **)**

注释

返回与此脚本版本关联的一组属性。这些属性存储在 MobiLink 系统表 ml_property 中。当 component_name 为 ScriptVersion 时，脚本版本存储在 property_set_name 列中。

参数

- **script_version** 要为其返回关联属性的脚本版本。

返回值

与给定脚本版本关联的一组属性。

另请参见

- “ml_property”一节第 670 页
- “ml_add_property 系统过程”一节第 637 页

示例

以下代码列出了所有与给定脚本版本相关联的 ServerContext 的属性。

```
import java.util.*;
// ServerContext serverContext;
// PrintStream out
Properties prop = serverContext.getPropertiesByVersion("MyScriptVersion");
prop.list(out);
```

getPropertySetNames 方法

语法

public Iterator **getPropertySetNames(**
java.lang.String *component_name* **)**

注释

返回给定组件的属性集名称列表。这些属性存储在 MobiLink 系统表 ml_property 中。

参数

- **component_name** 要列出其属性名称的组件的名称。

返回值

给定组件的属性集名称列表。

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

另请参见

- “ml_property”一节第 670 页
- “ml_add_property 系统过程”一节第 637 页

示例

以下代码列出了所有与给定组件相关联的 ServerContext 的属性。

```
import java.util.*;
// ServerContext serverContext;
// PrintStream out
Properties prop = serverContext.getPropertySetNames("Component Name");
prop.list(out);
```

getStartClassInstances 方法

语法

public java.lang.Object[ ] **getStartClassInstances( )**
获取包含服务器启动时所构造的启动类的数组。如果没有启动类，此数组的长度为零。

返回值

返回包含服务器启动时所构造的启动类的数组，或者如果没有启动类，则返回长度为零的数组。

示例

以下是使用 getStartClassInstances() 的一个示例：

```
Object objs[] = sc.getStartClassInstances();
int i;
for (i=0; i < objs.length; i += 1) {
    if (objs[i] instanceof MyClass) {
        // Use class.
    }
}
```

另请参见

- “用户定义的启动类”一节第 501 页

makeConnection 方法

语法

public java.sql.Connection **makeConnection( )**
throws **java.sql.SQLException**

注释

创建新的服务器连接。如果在打开新连接时出现错误，则此方法将抛出 java.sql.SQLException。

返回值

新的服务器连接。

异常

- **SQLException** 如果在打开新连接时出错，则将其抛出。

removeErrorListener 方法

语法

public void **removeErrorListener(** LogListener *ll* **)**

注释

从在输出错误时接收通知的监听器的列表中删除指定的 LogListener。

参数

- **ll** 不再通知的 LogListener。

示例

以下代码从错误监听器列表中删除了 LogListener。

```
// ServerContext serverContext;
// LogListener myErrorListener
serverContext.removeErrorListener(myErrorListener);
```

removeInfoListener 方法

语法

public void **removeInfoListener(** LogListener *ll* **)**

注释

从监听器列表删除指定的 LogListener，在输出信息时接收通知。

参数

- **ll** 不再通知的监听器。

示例

以下代码从信息监听器列表中删除了 LogListener。

```
// ServerContext serverContext;
// LogListener myInfoListener
serverContext.removeInfoListener(myInfoListener);
```

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

removeShutdownListener 方法

语法

public void **removeShutdownListener(** ShutdownListener *sl* **)**

注释

从将在服务器上下文被取消前接收通知的监听器列表中删除指定的 ShutdownListener。

参数

- **sl** 服务器关闭时不再通知的 ShutdownListener。

示例

在服务器上下文被取消前，以下代码将 ShutdownListener 从接收通知的 监听器列表中删除。

```
// ServerContext serverContext;
// ShutdownListener myShutdownListener
serverContext.removeShutdownListener(myShutdownListener);
```

removeWarningListener 方法

语法

public void **removeWarningListener(** LogListener *ll* **)**

注释

从在输出警告时接收通知的监听器的列表中删除指定的 LogListener。

参数

- **ll** 不再通知的 LogListener。

示例

以下代码从警告监听器列表中删除了 LogListener。

```
// ServerContext serverContext;
// LogListener myWarningListener
serverContext.removeWarningListener(myWarningListener);
```

shutdown 方法

语法

public void **shutdown( )**

注释

强制服务器关闭。任何已注册的 ShutdownListener 实例的 shutdownPerformed 方法都将被调用。

示例

以下代码强制服务器关闭。

```
// ServerContext serverContext;
serverContext.shutdown();
```

ServerException 类

语法

public **ianywhere.ml.script.ServerException**

注释

抛出此异常说明出现使服务器上的同步无法进一步执行的错误状况。这将导致 MobiLink 服务器关闭。

成员

ianywhere.ml.script.ServerException 的所有成员，包括所有继承的成员。

- “ServerException 构造函数”一节第 539 页

示例

以下代码是一个函数，可在发生严重问题时抛出 ServerException，从而致使 MobiLink 服务器关闭。

```
public void handleUpload(UploadData ud)
    throws SQLException, IOException, ServerException
{

    UploadedTableData tables[] = ud.getUploadedTables();
    if (tables == null) {
        throw new ServerException("Failed to read uploaded tables");
    }

    for (int i = 0; i < tables.length; i++) {
        UploadedTableData currentTable = tables[i];
        println("table " + java.lang.Integer.toString(i)
            + " name: " + currentTable.getName());

        // Print out delete result set.
        println("Deletes");
        printRSInfo(currentTable.getDeletes());

        // Print out insert result set.
        println("Inserts");
        printRSInfo(currentTable.getInserts());

        // print out update result set
        println("Updates");
        printUpdateRSInfo(currentTable.getUpdates());
    }
}
```

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

ServerException 构造函数

语法

public **ServerException( )**

注释

构造一个不带任何细节消息的 ServerException。

语法

public **ServerException(** java.lang.String *s* **)**

注释

构造带有指定细节消息的 ServerException。

参数

- **s** 细节消息。

示例

请参见： “ServerException 类”一节第 538 页.

ShutdownListener 接口

语法

public **ianywhere.ml.script.ShutdownListener**

注释

用于捕获服务器关闭的监听器接口。使用此接口可确保在服务器退出前清理所有线程、连接及其它资源。

成员

ianywhere.ml.script.ShutdownListener 的所有成员，包括所有继承的成员。

- “shutdownPerformed 方法”一节第 540 页

示例

以下代码为 ServerContext 安装 ShutdownListener。

```
class MyShutdownListener implements ShutdownListener {
    FileOutputStream _outFile;
    public MySutdownListener(FileOutputStream outFile) {
        _outFile = outFile;
    }

    public void shutdownPerformed(ServerContext sc) {
        // Add shutdown code
        try {
```

```
                _outFile.write(("Shutting Down" + "\n").getBytes());
                _outFile.flush();
            }
            catch(Exception e) {
                // Print some error output to the MobiLink log.
                e.printStackTrace();
            }
            // ...
        }
    }
```

以下代码注册 MyShutdownListener。可以从有权访问 ServerContext 的任何位置（例如类构造函数或同步脚本）调用此代码。

```
// ServerContext serv_context;
// FileOutputStream outFile
serv_context.addShutdownListener(new MyShutdownListener(outFile));
```

shutdownPerformed 方法

语法

public void **shutdownPerformed(** ServerContext *sc***)**

注释

在 ServerContext 因服务器关闭而被取消之前，将调用此方法。

参数

- **sc** 正在关闭的服务器的上下文。

示例

请参见： “ShutdownListener 接口”一节第 539 页.

SynchronizationException 类

语法

public **ianywhere.ml.script.SynchronizationException**

注释

抛出此异常说明出现了使当前同步无法完成的错误状况。这强制 MobiLink 服务器回退。

成员

ianywhere.ml.script.SynchronizationException 的所有成员，包括所有继承的成员。

- “SynchronizationException 构造函数”一节第 541 页

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

示例

以下代码是一个函数，可在发生问题时抛出 SynchronizationException，从而致使 MobiLink 服务器回退。

```
public void handleUpload(UploadData ud)
    throws SQLException, IOException, SynchronizationException
{

    UploadedTableData tables[] = ud.getUploadedTables();

    for (int i = 0; i < tables.length; i++) {
        UploadedTableData currentTable = tables[i];
        println("table " + java.lang.Integer.toString(i)
            + " name: " + currentTable.getName());

        // Print out delete result set.
        println("Deletes");
        printRSInfo(currentTable.getDeletes());

        // Print out insert result set.
        println("Inserts");
        printRSInfo(currentTable.getInserts());

        // print out update result set
        println("Updates");
        printUpdateRSInfo(currentTable.getUpdates());

        if (/* Reason for Sync failure */) {
            throw new SynchronizationException("Sync Failed");
        }
    }
}
```

SynchronizationException 构造函数

语法

public **SynchronizationException( )**

注释

构造一个不带任何细节消息的 SynchronizationException。

语法

public **SynchronizationException(** java.lang.String *s* **)**

注释

用指定的细节消息构造 SynchronizationException。

参数

- **s** 细节消息。

UpdateResultSet

语法

public **ianywhere.ml.script.UpdateResultSet**

注释

一个结果集对象，其中包括用于访问指定行前映像（旧）和后映像（新）值的特殊方法。要获取 UpdateResultSet 实例，请使用 DownloadTableData.getUpdates 方法。

UpdateResultSet 是对 java.sql.ResultSet 的扩展，它添加了 setNewRowValues 和 setOldRowValues 方法。另外，它也可以用作常规的结果集。有关 java.sql.ResultSet 的详细信息，请参考 Java SDK 文档。

另请参见

- DownloadTableData “getUpdates 方法”一节第 548 页
- “直接上载的冲突处理”一节第 615 页
- “直接行处理”第 609 页

成员

ianywhere.ml.script.UpdateResultSet 的所有成员，包括所有继承的成员。

- “setNewRowValues 方法”一节第 542 页
- “setOldRowValues 方法”一节第 543 页

示例

以下代码显示如何从 DownloadTableData 实例中获取 UpdateResultSet 实例。

```
// DownloadTableData tableData
UpdateResultSet results = tableData.getUpdates();
```

setNewRowValues 方法

语法

public void **setNewRowValues( )**

注释

将此结果集的模式设置为返回新列值（更新后的行）。此结果集表示远程客户端数据库中最新更新的值。这是缺省模式。

另请参见

- “UpdateResultSet”一节第 542 页
- “直接上载的冲突处理”一节第 615 页
- “直接行处理”第 609 页

Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

示例

以下代码显示如何设置 UpdateResultSet 的模式以返回新的列值。

```
// UpdateResultSet results
results.setNewRowValues();
```

setOldRowValues 方法

语法

public void **setOldRowValues( )**

注释

将此结果集的模式设置为返回旧列值（更新前的行）。在此模式中，UpdateResultSet 表示上次同步中客户端所获得的旧列值。

另请参见

- “UpdateResultSet”一节第 542 页
- “直接上载的冲突处理”一节第 615 页
- “直接行处理”第 609 页

示例

以下代码显示如何设置 UpdateResultSet 的模式以返回旧的列值。

```
// UpdateResultSet results
results.setOldRowValues();
```

UploadData 接口

语法

public **ianywhere.ml.script.UploadData**

注释

封装用于直接行处理的上载操作。表示单个上载事务的 UploadData 实例被传递给 handle_UploadData 同步事件。

> **小心**
> 您必须在针对 handle_UploadData 事件注册的方法中处理直接行处理上载操作。UploadData 在每次调用注册的方法后被取消。不要创建 UploadData 的新实例以在后续事件中使用。

使用 UploadData.getUploadedTables 或 UploadData.getUploadedTableByName 方法来获取 UploadedTableData 实例。

除非远程数据库采用事务性上载，否则一个同步将有一个 UploadData。

另请参见

- “UploadedTableData 接口”一节第 545 页
- “handle_UploadData 连接事件”一节第 422 页
- “直接行处理”第 609 页
- “处理直接上载”一节第 614 页

成员

ianywhere.ml.script.UploadData 的所有成员，包括所有继承的成员。

- “getUploadedTableByName 方法”一节第 544 页
- “getUploadedTables 方法”一节第 545 页

示例

请参见“handle_UploadData 连接事件”一节第 422 页。

getUploadedTableByName 方法

语法

```
public UploadedTableData getUploadedTableByName(
  java.lang.String table_name
)
```

注释

返回表示所指定表的 UploadedTableData 实例。

参数

- **table_name** 想要获取其上载数据的所上载表的名称。

返回值

返回表示所指定表的 UploadedTableData 实例，或者如果当前同步中不存在具有所指定名称的表，则返回空值。

另请参见

- “UploadData 接口”一节第 543 页
- “UploadedTableData 接口”一节第 545 页
- “直接行处理”第 609 页

示例

假定对 handle_UploadData 同步事件使用称为 HandleUpload 的方法。以下示例使用 getUploadedTableByName 方法返回 remoteOrders 表的 UploadedTableData 实例。

```
// The method used for the handle_UploadData event.

public void handleUpload(UploadData ut)
    throws SQLException, IOException
{
```

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

```
    UploadedTableData uploaded_t1 =
ut.getUploadedTableByName("remoteOrders");
    // ...
}
```

getUploadedTables 方法

语法

public UploadedTableData[] **getUploadedTables( )**

注释

返回包含当前同步中 UploadedTableData 对象的数组。数组中表的顺序与 MobiLink 在进行 SQL 的行处理时所采用的顺序相同，这一顺序能最有效地防止参照完整性违规。如果数据源为关系数据库，请使用此表顺序。

返回值

包含当前同步中 UploadedTableData 对象的数组。数组中表的顺序与客户端的上载顺序相同。

另请参见

- “UploadData 接口”一节第 543 页
- “UploadedTableData 接口”一节第 545 页
- “直接行处理”第 609 页

示例

假定对 handle_UploadData 同步事件使用称为 HandleUpload 的方法。以下示例使用 getUploadedTables 方法为当前上载事务返回 UploadedTableData 实例。

```
// The method used for the handle_UploadData event.

public void handleUpload(UploadData ud)
    throws SQLException, IOException
{
    UploadedTableData tables[] = ud.getUploadedTables();
    //...
}
```

UploadedTableData 接口

语法

public **ianywhere.ml.script.UploadedTableData**

注释

封装用于直接行处理上载的表操作。可以使用 UploadedTableData 实例来获取单个上载事务的表插入、更新和删除操作。使用 UploadedTableData.getInserts、UploadedTableData.getUpdates 和 UploadedTableData.getDeletes 方法返回标准 JDBC java.sql.ResultSet 对象。

有关 java.sql.ResultSet 和 java.sql.ResultSetMetaData 的详细信息，请参考 Java SDK 文档。

可使用 UploadedTableData.getMetaData 方法或由 getInserts、getUpdates 和 getDeletes 返回的结果集来访问表元数据。删除结果集只包括表的主键列。

另请参见

- “UploadData 接口”一节第 543 页
- “handle_UploadData 连接事件”一节第 422 页
- “直接行处理”第 609 页

成员

ianywhere.ml.script.UploadedTableData 的所有成员，包括所有继承的成员。

- “getDeletes 方法”一节第 546 页
- “getInserts 方法”一节第 547 页
- “getUpdates 方法”一节第 548 页
- “getName 方法”一节第 549 页
- “getMetaData 方法”一节第 549 页

示例

请参见： “UploadData 接口”一节第 543 页.

getDeletes 方法

语法

public java.sql.ResultSet **getDeletes( )**

注释

返回表示 MobiLink 客户端所上载的删除操作的 java.sql.ResultSet 对象。结果集包含所删除行的主键值。

返回值

表示 MobiLink 客户端所上载的删除操作的 java.sql.ResultSet 对象。

另请参见

- “UploadedTableData 接口”一节第 545 页
- “handle_UploadData 连接事件”一节第 422 页
- “直接行处理”第 609 页

示例

假定远程客户端包含一个名为 remoteOrders 的表。以下示例使用 DownloadTableData.getDeletes 方法获取所删除行的结果集。在本例中，删除结果集包含单个主键列。

```
import ianywhere.ml.script.*;
import java.sql.*;
```

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

```
// The method used for the handle_UploadData event.
public void HandleUpload(UploadData ut)
    throws SQLException, IOException
{
    // Get an UploadedTableData for the remoteOrders table.
    UploadedTableData remoteOrdersTable =
ut.getUploadedTableByName("remoteOrders");

    // Get deletes uploaded by the MobiLink client.
    java.sql.ResultSet delete_rs = remoteOrdersTable.getDeletes();

     while (delete_rs.next()) {
         // Get primary key values for deleted rows.
         int deleted_id = delete_rs.getInt(1);

         // ...
    }
    delete_rs.close();
}
```

getInserts 方法

语法

public java.sql.ResultSet **getInserts( )**

注释

返回表示 MobiLink 客户端所上载的插入操作的 java.sql.ResultSet 对象。每个插入都由结果集中的一行来表示。

返回值

表示 MobiLink 客户端所上载的插入操作的 java.sql.ResultSet 对象。

另请参见

- “UploadedTableData 接口”一节第 545 页
- “直接行处理”第 609 页

示例

假定远程客户端包含一个名为 remoteOrders 的表。以下示例使用 DownloadTableData.getInserts 方法获取所插入行的结果集。此代码将获取当前上载事务中每行的订单额。

```
import ianywhere.ml.script.*;
import java.sql.*;

// The method used for the handle_UploadData event
public void HandleUpload(UploadData ut)
    throws SQLException, IOException
{
    // Get an UploadedTableData instance representing the remoteOrders table.
    UploadedTableData remoteOrdersTable =
ut.getUploadedTableByName("remoteOrders");

    // Get inserts uploaded by the MobiLink client.
```

```
        java.sql.ResultSet rs = remoteOrdersTable.getInserts();
        while (rs.next()) {
            // get the uploaded order_amount
            double order_amount = rs.getDouble("order_amount");

            // ...
        }
        rs.close();
    }
```

getUpdates 方法

语法

public UpdateResultSet **getUpdates( )**

注释

返回表示 MobiLink 客户端所上载的更新操作的 UpdateResultSet 对象。每个更新由包括所有列值的一行来表示。UpdateResultSet 是对 java.sql.ResultSet 的扩展，它加入了用于 MobiLink 冲突检测的特殊方法。

返回值

表示 MobiLink 客户端所上载的更新操作的 UpdateResultSet 对象。

另请参见

- “UploadedTableData 接口”一节第 545 页
- “UpdateResultSet”一节第 542 页
- “handle_UploadData 连接事件”一节第 422 页
- “直接上载的冲突处理”一节第 615 页
- “直接行处理”第 609 页

示例

假定远程客户端包含一个名为 remoteOrders 的表。以下示例使用 UploadedTableData.getUpdates 方法获取所更新行的结果集。此代码将获取每行的订单额。

```
import ianywhere.ml.script.*;
import java.sql.*;

// the method used for the handle_UploadData event
public void HandleUpload(UploadData ut)
    throws SQLException, IOException
{
    // Get an UploadedTableData instance representing the remoteOrders table.
    UploadedTableData remoteOrdersTable =
ut.getUploadedTableByName("remoteOrders");

    // Get inserts uploaded by the MobiLink client.
    java.sql.ResultSet rs = remoteOrdersTable.getUpdates();
    while (rs.next()) {
        // Get the uploaded order_amount.
        double order_amount = rs.getDouble("order_amount");

        // ...
```

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

```
        }
        rs.close();
    }
```

getName 方法

语法

public java.lang.String **getName( )**

注释

返回 UploadedTableData 实例的表名。也可以使用由 getMetaData 方法返回的 java.sql.ResultSetMetaData 实例来访问表名。

返回值

UploadedTableData 实例的表名。

另请参见

- “UploadedTableData 接口”一节第 545 页
- UploadedTableData “getMetaData 方法”一节第 549 页
- “handle_UploadData 连接事件”一节第 422 页
- “直接行处理”第 609 页

示例

以下示例将获取单个上载事务中每个所上载表的名称。

```
import ianywhere.ml.script.*;
import java.sql.*;

// The method used for the handle_UploadData event.
public void HandleUpload(UploadData ud) {
    throws SQLException, IOException
{
    int i;

    // Get UploadedTableData instances.
    UploadedTableData tables[] = ud.getUploadedTables();

    for (i=0; i<tables.length; i+=1) {
        // Get the table name.
        String table_name = tables[i].getName();

        // ...
    }
}
```

getMetaData 方法

语法

public java.sql.ResultSetMetaData **getMetaData( )**

注释

获取 UploadedTableData 实例的元数据。元数据是标准的 java.sql.ResultSetMetaData 实例。

如果想要 ResultSetMetaData 包含列名信息，则必须将客户端选项设置为发送列名。

有关 java.sql.ResultSetMetaData 的详细信息，请参考 Java SDK 文档。

返回值

UploadedTableData 实例的元数据。

另请参见

- dbmlsync： “SendColumnNames (scn) 扩展选项”一节 《MobiLink - 客户端管理》
- UltraLite： “Send Column Names 同步参数”一节 《UltraLite - 数据库管理和参考》

示例

以下示例将获取名为 remoteOrders 的上载表的 java.sql.ResultSetMetaData 实例。此代码使用 ResultSetMetaData.getColumnCount 和 getColumnLabel 方法来编译一组列名。

```
import ianywhere.ml.script.*;
import java.sql.*;

// The method used for the handle_UploadData event.
public void HandleUpload(UploadData ut) {
    throws SQLException, IOException
{
    // Get an UploadedTableData instance representing the remoteOrders table.
    UploadedTableData remoteOrdersTable =
ut.getUploadedTableByName("remoteOrders");

    // get inserts uploaded by the MobiLink client
    java.sql.ResultSet rs = remoteOrdersTable.getInserts();

    // Obtain the result set metadata.
    java.sql.ResultSetMetaData md = rs.getMetaData();
    String columnHeading = "";

    // Compile a list of column names.
    for (int c=1; c <= md.getColumnCount(); c += 1) {
        columnHeading += md.getColumnLabel();

        if (c < md.getColumnCount()) {
            columnHeading += ", ";
        }
    }
    //...
}
```

在本例中，使用名为 HandleUpload 的方法处理 handle_UploadData 同步事件。

另请参见

- “UploadedTableData 接口”一节第 545 页
- “直接行处理”第 609 页
- “SendColumnNames (scn) 扩展选项”一节 《MobiLink - 客户端管理》
- “handle_UploadData 连接事件”一节第 422 页

Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

使用 .NET 编写同步脚本

目录

.NET 同步逻辑简介 552
设置 .NET 同步逻辑 553
编写 .NET 同步逻辑 555
.NET 同步技术 562
装载共享程序集 563
.NET 同步示例 565
用于 .NET 参考的 MobiLink 服务器 API 567

.NET 同步逻辑简介

MobiLink 支持使用 Visual Studio 编程语言编写同步脚本。使用 .NET 编写 MobiLink 脚本时，可以使用任何能够创建有效的 .NET 程序集的语言。特别是，以下语言已经过测试，并且具有相关文档：

- C#
- Visual Basic .NET
- C++

.NET 同步逻辑与 SQL 逻辑的功能相同：正如 MobiLink 服务器可以在 MobiLink 事件发生时访问 SQL 脚本一样，它也可以在 MobiLink 事件发生时调用 .NET 方法。.NET 方法可将一个 SQL 字符串返回到 MobiLink。

本节讲解如何设置、开发和运行 C#、Visual Basic .NET 和 C++ 的 .NET 同步逻辑。其中包括一个示例应用程序和用于 .NET 参考的 MobiLink 服务器 API。

另请参见

- “教程：使用 .NET 同步逻辑” 《MobiLink - 入门》
- “用于编写服务器端同步逻辑的选项”一节 《MobiLink - 入门》
- “编写同步脚本”第 293 页

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

设置 .NET 同步逻辑

使用 .NET 实现同步脚本时，必须告知 MobiLink 在何处可以找到程序集中包含的程序包、类和方法。

♦ 使用 .NET 实施同步脚本

1. 创建自己的类。为每个所需的同步事件编写方法。这些方法必须是公共的。

   有关这些方法的详细信息，请参见“方法”一节第 557 页。

   每个包含非静态方法的类都应有一个公共的构造函数。在第一次调用每个类中的方法进行连接时，MobiLink 服务器将自动实例化每一个类。

   请参见“构造函数”一节第 556 页。

2. 创建一个或多个程序集。编译时可引用 *iAnywhere.MobiLink.Script.dll*，它包含将要在您自己的 .NET 方法中使用的 MobiLink 服务器 API 类存储库。*iAnywhere.MobiLink.Script.dll* 位于 *install-dir\Assembly\v2* 中。

   您可以在命令行中编译类，也可以使用 Visual Studio 或其它 .NET 开发环境编译类。

   请参见“用于 .NET 参考的 MobiLink 服务器 API”一节第 567 页。

3. 编译您的项目。

   例如，在 Visual Studio 中进行编译，如下所示：

   a. 在 [**VS.NET Project**] 菜单中，选择 [**Add Existing Item**]。

   b. 定位 *iAnywhere.MobiLink.Script.dll*。

      在 [**Open**] 列表中，选择 [**Link File**]。

      注意
      对于 Visual Studio，请始终使用 [Link File] 方法。请不要使用 [Add Reference] 选项来引用 *iAnywhere.MobiLink.Script.dll*。[Add Reference] 选项将复制与类程序集位于同一物理目录的 *iAnywhere.MobiLink.Script.dll*，这会导致 MobiLink 服务器出现问题。

   c. 使用 [**Build**] 菜单生成程序集。

   还可以从命令行进行编译，方式如下：

   将 *dll-path* 替换为 *iAnywhere.MobiLink.Script.dll* 的路径。例如，在 C# 中：

   ```
   csc /out:dll-pathout.dll /target:library /reference:dll-
   pathiAnywhere.MobiLink.Script.dll sync_v1.cs
   ```

4. 在统一数据库的 MobiLink 系统表中，为每个同步脚本指定要调用的程序包、类和方法的名称。每个脚本版本中只允许存在一个类。

   例如，可以通过使用 ml_add_dnet_connection_script 存储过程或 ml_add_dnet_table_script 存储过程，将此信息添加到 MobiLink 系统表中。如果在 SQL Anywhere 数据库中运行以下 SQL 语句，则这些语句将指定只要发生 authenticate_user 连接级别事件，即运行 myNamespace.myClass.myMethod。

```
CALL ml_add_dnet_connection_script(
    'version1',
    'authenicate_user',
    'myNamespace.myClass.myMethod'
)
```

注意
完全限定的方法名称区分大小写。

作为此过程调用的结果，ml_script 系统表的 script_language 列包含单词 **dnet**。脚本列包含公共 .NET 方法的限定名称。

请参见“ml_add_dnet_connection_script 系统过程”一节第 628 页和“ml_add_dnet_table_script 系统过程”一节第 629 页。

您也可以使用 Sybase Central 添加此信息。

请参见“添加和删除脚本”一节第 306 页。

5. 指示 MobiLink 服务器装载程序集并启动 CLR。可以使用 mlsrv11 命令行选项告知 MobiLink 在何处找到这些程序集。有两个选项可以选择：

   - **使用 -sl dnet (-MLAutoLoadPath)** 此选项将给定路径设置为应用程序基目录，并装载该目录下的所有专用程序集。在大多数情况下都应使用此选项。例如，要装载位于 *dll-path* 下的所有程序集，请输入：

     ```
     mlsrv11 -c "dsn=consolidated1" -sl dnet(-MLAutoLoadPath=dll-path)
     ```

     如果使用 -MLAutoLoadPath 选项，则在输入事件脚本的完全限定方法名时不能指定域。

     请参见“装载程序集”一节第 563 页和“-sl dnet 选项”一节第 86 页。

   - **使用 -sl dnet (-MLDomConfigFile)** 此选项需要一个包含域和程序集设置的配置文件。您可在以下情况下使用此选项：当您具有共享程序集时，当您不想装载目录中的所有程序集时，或者当您出于某种其它原因需要使用配置文件时。

     有关装载共享程序集的详细信息，请参见“装载程序集”一节第 563 页。有关 mlsrv11 选项 -sl dnet 的详细信息，请参见“-sl dnet 选项”一节第 86 页。

     注意
     可以使用 -MLAutoLoadPath 选项，也可以使用 -MLDomConfigFile 选项，但二者不可同时使用。

Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

编写 .NET 同步逻辑

要编写 .NET 同步逻辑，您需要熟知 MobiLink 事件，了解 .NET，并熟悉用于 .NET 的 MobiLink 服务器 API。

有关 API 的完整说明，请参见“用于 .NET 参考的 MobiLink 服务器 API”一节第 567 页。

.NET 同步逻辑可用于维护状态信息，以及实现上载和下载事件的逻辑。例如，使用 .NET 编写的 begin_synchronization 脚本能够将 MobiLink 用户名存储在变量中。在同步过程后期调用的脚本可以访问此变量。此外，您还可以在提交统一数据库中的行之前或之后，使用 .NET 访问这些行。

使用 .NET 还可以减少对统一数据库的依赖。通过将统一数据库升级到新版本或切换到另一数据库管理系统，可使行为受到的影响得以减少。

直接行处理

可以使用 MobiLink 直接行处理实现远程数据与任何中央数据源、应用程序或 Web 服务之间的通信。直接行处理使用用于 Java 或 .NET 的 MobiLink 服务器 API 中的特殊类来直接访问同步数据。

请参见“直接行处理”第 609 页。

类实例

MobiLink 服务器在数据库连接级别实例化类。当到达某个已经为其编写了非静态 .NET 方法的事件时，如果尚未在现有数据库连接上实例化类，则 MobiLink 服务器将自动实例化该类。

请参见“构造函数”一节第 556 页。

注意
对于一个脚本版本，所有与连接级别或表级别事件直接关联的方法必须属于同一个类。

对于每个数据库连接，类在实例化后会一直持续，直到连接关闭。因此，多个连续的同步会话可能使用同一个实例。除非将其显式地清除，公共或专用变量中的信息会跨同一连接上的多个同步而持续存在。

您也可以使用静态的类或变量。在这种情况下，这些值在相同域中的所有连接间都可用。

仅当与统一数据库的连接关闭时，MobiLink 服务器才会自动删除类实例。

事务

有关事务的一般规则适用于 .NET 方法。数据库事务的启动和运行时间对同步进程来说至关重要。事务只能由 MobiLink 服务器启动和结束。在 .NET 方法内的同步连接中显式提交或回退事务，将破坏同步进程的完整性且可能导致错误。

这些规则只适用于由 MobiLink 服务器创建的数据库连接，特别是由方法返回的 SQL 语句。

SQL-.NET 数据类型

下表显示 SQL 数据类型和用于 MobiLink 脚本参数的相应 .NET 数据类型。

SQL 数据类型	相应的 .NET 数据类型
VARCHAR	string
CHAR	string
INTEGER	int
BINARY	byte [ ]
TIMESTAMP	DateTime
INOUT INTEGER	ref int
INOUT VARCHAR	ref string
INOUT CHAR	ref string
INOUT BYTEARRAY	ref byte [ ]

构造函数

类的构造函数或者不带有任何参数，或者带一个 iAnywhere.MobiLink.Script.DBConnectionContext 参数。例如：

```
public ExampleClass(iAnywhere.MobiLink.Script.DBConnectionContext cc)
```

或者

```
public ExampleClass()
```

传递给您的同步上下文是针对某个连接的，MobiLink 服务器通过该连接与当前用户保持同步。

DBConnectionContext.GetConnection 方法返回的数据库连接与 MobiLink 用来与当前用户保持同步的数据库连接相同。您可以在此连接上执行语句，但是不能提交或回退事务。这些事务由 MobiLink 服务器进行管理。

MobiLink 服务器使用带有一个 iAnywhere.MobiLink.Script.DBConnectionContext 参数（如果存在）的构造函数。如果该参数不存在，则 MobiLink 服务器会使用空的构造函数。

请参见“DBConnectionContext 接口”一节第 570 页。

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

方法

一般说来，您为每个同步事件实现一种方法。这些方法必须是公共的。如果这些方法是专用的，则 MobiLink 服务器无法使用它们且无法识别它们的存在。

方法的名称并不重要，只要与统一数据库 ml_script 表中所指定的名称匹配即可。不过，在本文档包括的示例中，方法的名称与 MobiLink 事件的名称相同。该命名约定使 .NET 代码更易于阅读。

方法的签名应与该事件脚本的签名相匹配，不过如果参数列表尾的参数值不再需要，可以将参数列表截断。您应该只接受所需的参数，因为开销的大小与传递参数的多少有关。

但该方法不能重载。在 ml_script 系统表中，每个类只能有一个方法原型。

注册方法

创建方法之后，必须将其注册。注册方法会在统一数据库的 MobiLink 系统表中创建对该方法的引用，以使该方法在事件发生时被调用。注册方法的方式与添加同步脚本的方式几乎完全相同，其唯一的差别是：您仅添加限定的方法名，而不会将整个 SQL 脚本添加到 MobiLink 系统表中。

请参见“添加和删除脚本”一节第 306 页。

返回值

为进行基于 SQL 的上载或下载所调用的方法必须返回有效的 SQL 语言语句。这些方法的返回类型必须是 String。而不允许是其它返回类型。

所有其它脚本的返回类型必须是 string 或 void。而不允许是其它类型。如果返回类型是 string 并且非空，MobiLink 服务器将假设该字符串中包含有效的 SQL 语句，并将在统一数据库中执行该语句，如同执行正常的 SQL 语言同步脚本一样。如果一个方法正常返回了一个字符串，但并不想基于其返回的内容对数据库执行 SQL 语句，则它可以返回空值。

用户定义的启动类

可以定义在服务器启动时自动装载的启动类。利用这一功能，就可以编写当 MobiLink 服务器启动 CLR 时（在第一次同步之前）执行的 .NET 代码。这意味着您可以在服务器实例中的第一个用户同步请求之前，创建连接或高速缓存数据。

此功能是通过使用 mlsrv11 -sl dnet 选项的 MLStartClasses 选项来实现的。例如，以下是 mlsrv11 命令行的部分内容。它将使 mycl1 和 mycl2 作为启动类装载。

```
-sl dnet(-MLStartClasses=MyNameSpace.MyClass.mycl1,MyNameSpace.MyClass.mycl2)
```

将按所列顺序装载各个类。如果多次列出同一个类，则会创建多个实例。

所有启动类必须是公共类，并且必须具有一个不接受任何参数，或接受一个类型为 MobiLink.Script.ServerContext 的参数的公共构造函数。

装载的启动类的名称输出到 MobiLink 日志，显示以下消息："已装载 .NET 启动类 *classname*"。

有关 .NET CLR 的详细信息，请参见“-sl dnet 选项”一节第 86 页。

要查看服务器启动时所构造的启动类，请参见“GetStartClassInstances 方法”一节第 590 页。

示例

下面是一个启动类模板。它启动一个处理事件并创建数据库连接的守护程序线程。（并非所有启动类都需要创建线程，但如果生成线程，则应是守护程序线程。）

```
using System;
using System.IO;
using System.Threading;
using iAnywhere.MobiLink.Script;

namespace TestScripts {
    public class MyStartClass {
        ServerContext    _sc;
        bool             _exit_loop;
        Thread           _thread;
        OdbcConnection   _conn;

        public MyStartClass(ServerContext sc) {

            // Perform setup first so that an exception
            // causes MobiLink startup to fail.
            _sc = sc;

            // Create connection for use later.
            _conn = _sc.makeConnection();
            _exit_loop = false;
            _thread = new Thread(new ThreadStart(run)) ;
            _thread.IsBackground = true;
            _thread.Start();
        }

        public void run() {
            ShutdownCallback callback = new
ShutdownCallback(shutdownPerformed);
            _sc.ShutdownListener += callback;

            // run() can't throw exceptions.
            try {
                handlerLoop();
                _conn.close();
                _conn = null;
            }
            catch(Exception e) {
                // Print some error output to the MobiLink log.
                Console.Error.Write(e.ToString());

                // There is no need to be notified of shutdown.
                _sc.ShutdownListener -= callback;

                // Ask server to shut down so this fatal error can be fixed.
                _sc.Shutdown();
            }

            // Shortly after return, this thread no longer exists.
            return;
        }

        public void shutdownPerformed(ServerContext sc) {
            // Stop the event handler loop.
            try {
                _exit_loop = true;

                // Wait a maximum of 10 seconds for thread to die.
```

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

```
                _thread.Join(10*1000);
            }
            catch(Exception e) {
                // Print some error output to the MobiLink log.
                Console.Error.Write(e.ToString());
            }
        }

        private void handlerLoop() {
            while (!_exit_loop) {
                // Handle events in this loop.
                Thread.Sleep(1*1000);
            }
        }
    }
}
```

从 .NET 打印信息

也可以选择使用 System.Console. Doing 向 .NET 方法中添加语句，将信息输出到 MobiLink 日志中。这样做可以帮助跟踪类的进程和行为。

性能提示
通过此种方法将信息输出到 MobiLink 日志是一种非常有用的监控手段，但是不建议在生产方案中使用。

同样的技术也可以用于记录任意同步信息或收集脚本使用情况的统计信息。

使用 .NET 处理 MobiLink 服务器错误

如果扫描日志无法获取足够的信息，可以通过编程方式监控应用程序。例如，可以通过电子邮件发送某类消息。

您可以编写方法，用于接收代表输出到日志中每条错误或警告消息的类。这样可以帮助您监控和审计 MobiLink 服务器。

以下代码为所有错误消息安装监听器，并将信息输出到 StreamWriter。

```
class TestLogListener {
    public TestLogListener(StreamWriter output_file) {
        _output_file    = output_file;
    }

    public void errCallback(ServerContext sc, LogMessage lm) {
        string type;
        string user;

        if (lm.Type == LogMessage.MessageType.ERROR) {
            type = "ERROR";
        } else if (lm.Type==LogMessage.MessageType.WARNING) {
            type = "WARNING";
        }
        else {
            type = "INVALID TYPE!!";
```

```
            }
            if (lm.User == null) {
                user = "null";
            }
            else {
                user = lm.User;
            }

            _output_file.WriteLine("Caught msg type=" + type
                + " user=" + user
                + " text=" + lm.Text);
            _output_file.Flush();
        }
        StreamWriter _output_file;
}
```

以下代码可注册 TestLogListener。从有权访问 ServerContext 的位置（例如类构造函数或同步脚本）调用此代码。

```
// ServerContext serv_context;
TestLogListener etll = new TestLogListener(log_listener_file);
serv_context.ErrorListener += new LogCallback(etll.errCallback);
```

另请参见

- “LogCallback 委派”一节第 587 页
- “ServerContext 接口”一节第 589 页中的 ErrorListener 和 WarningListener
- “LogMessage 类”一节第 587 页
- “MessageType 枚举”一节第 588 页

调试 .NET 同步逻辑

下面的过程介绍了一种可以使用 Visual Studio 调试 .NET 脚本的方法。

♦ **调试 .NET 脚本**

1. 在打开调试信息的情况下使用以下方法之一编译代码：
   - 在 csc 命令行上，设置 /debug+ 选项。
   - 使用 Microsoft Visual Studio 设置来设置调试输出。
     - 选择 [**File**] » [**Build**] » [**Configuration Manager**]。

       在 [**Active Solution Configuration**] 列表中，选择 [**Debug**]。
     - 构建您的程序集。
2. 关闭包含您源文件的 Visual Studio 运行实例。
3. 在这一步，启动了一个新的 Visual Studio 实例来调试 MobiLink 服务器和 .NET 同步脚本。使用命令行选项启动 Visual Studio 来调试 MobiLink 服务器。
   - 在命令提示符下，导航到安装 Visual Studio 的 *Common7\IDE* 子目录。
   - 使用 /debugexe 选项启动 devenv (Visual Studio IDE)。

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

例如，可运行以下命令来调试 MobiLink 服务器。请记住要指定 mlsrv11 选项，其中包括连接字符串和用于装载 .NET 程序集的选项。

对于 32 位 Windows 环境：

```
devenv /debugexe %sqlany11%\bin32\mlsrv11.exe -c ...
```

对于 64 位 Windows 环境：

```
devenv /debugexe %sqlany11%\bin64\mlsrv11.exe -c ...
```

Visual Studio 启动，然后 *mlsrv11.exe* 会出现在 [Solution Explorer] 窗口中。

4. 设置 Microsoft Visual Studio 以调试 .NET 代码：
   - 在 Visual Studio 的 [Solution Explorer] 窗口中，右击 [*mlsrv11.exe*] 并选择 [Properties]。
   - 将 [调试器类型] 由 [Auto] 更改为 [Mixed] 或 [Managed Only]，以确保 Visual Studio 只调试您的 .NET 同步脚本。
5. 打开关联的 .NET 源文件并设置断点。

   注意：在 mlsrv11 解决方案中分别打开源文件。不要打开原始解决方案或项目文件。
6. 从 [调试] 菜单或按 F5 键启动 MobiLink。

   如果出现提示，保存 *mlsrv11.sln*。

   如果出现 [No Symbolic Information] 窗口，请务必单击 [**OK**] 进行调试。您正在调试的是 MobiLink 调用的托管 .NET 同步脚本，而不是 MobiLink 服务器本身。
7. 执行一个使 MobiLink 执行含断点的代码的同步。

.NET 同步技术

本节介绍的技术可用于处理常见的 .NET 同步任务。

上载行或下载行

有关如何通过 .NET 上载或下载行的信息，请参见“直接行处理”第 609 页。

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

装载共享程序集

本节详细介绍装载 .NET 程序集的选项及装载共享程序集的过程。

装载程序集

.NET 程序集是一个由类型、元数据和可执行代码组成的程序包。在 .NET 应用程序中，所有代码必须都在程序集中。程序集文件的扩展名为 *.dll* 或 *.exe*。

存在以下类型的程序集：

- **专用程序集** 专用程序集是文件系统中的一个文件。
- **共享程序集** 共享程序集是安装在全局程序集高速缓存中的程序集。

MobiLink 必须先找到一个包含某个类的程序集，然后才能装载该类并调用该类的方法。MobiLink 只需要找到它直接调用的程序集。然后，该程序集便可调用它所需要的任何其它程序集。

例如，MobiLink 调用 MyAssembly，而 MyAssembly 调用 UtilityAssembly 和 NetworkingUtilsAssembly。在这种情况下，只需要通过配置使 MobiLink 能够找到 MyAssembly。

MobiLink 提供了以下装载程序集的方法：

- **使用 -sl dnet (-MLAutoLoadPath)** 此选项只适用于专用程序集。此选项将路径设置为应用程序基目录，并装载该目录下的所有程序集。

  如果使用 -MLAutoLoadPath 选项，则在输入事件脚本的完全限定方法名时不能指定域。

  通过 -MLAutoLoadPath 选项指定了路径和目录后，MobiLink 将执行以下操作：

  - 将该目录设置为应用程序基路径
  - 装载指定目录中所有扩展名为 *.dll* 或 *.exe* 的文件内的所有类
  - 创建一个应用程序域，并将所有未指定域的用户类载入该域

  不能通过此选项直接调用全局程序集高速缓存中的程序集。要调用这些共享程序集，请使用 -MLDomConfigFile。

- **使用 -sl dnet (-MLDomConfigFile)** 此选项既适用于专用程序集，也适用于共享程序集。它需要一个包含域和程序集设置的配置文件。您可在以下情况下使用此选项：当您具有共享程序集时，当您不想装载应用程序基路径中的所有程序集时，或者当您出于某种其它原因需要使用配置文件时。

  如果使用此选项，MobiLink 将读取指定域配置文件中的设置。域配置文件包含一个或多个 .NET 域的配置设置。如果该文件中指定了多个域，将使用指定的第一个域作为缺省域。（对于未指定域的脚本，将使用缺省域。）

  装载程序集时，MobiLink 首先尝试装载专用程序集，然后尝试从全局程序集高速缓存中装载程序集。专用程序集必须位于应用程序基目录中。共享程序集从全局程序集高速缓存中装载。

使用 -MLDomConfigFile 选项时，只能从事件脚本直接调用域配置文件中指定的程序集。

示例域配置文件

随 MobiLink 一起安装了一个名为 *mlDomConfig.xml* 的示例域配置文件。您可以从头编写您自己的文件，也可以根据您的需要对示例文件进行修改。示例文件位于 SQL Anywhere 路径中，其位置是：

MobiLink\setup\dnet\mlDomConfig.xml

以下是示例域配置文件 *mlDomConfig.xml* 的内容：

```
<?xml version="1.0" encoding="utf-8"?>
<config xmlns="iAnywhere.MobiLink.mlDomConfig"
xsi:schemaLocation='iAnywhere.MobiLink.mlDomConfig mlDomConfig.xsd'
xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance' >
    <domain>
        <name>SampleDomain1</name>
        <appBase>C:\scriptsDir</appBase>
        <configFile></configFile>
        <assembly name="Assembly1" />
        <assembly name="Assembly2" />
    </domain>

    <domain>
        <name>SampleDomain2</name>
        <appBase>\Dom2assembly</appBase>
        <configFile>\Dom2assembly\AssemblyRedirects.config</configFile>
        <assembly name="Assembly3" />
        <assembly name="Assembly4" />
    </domain>
</config>
```

以下是对 *mlDomConfig.xml* 内容的解释：

- **name** 域名，在事件脚本中指定域时使用。格式为 ["DomainName:Namespace.Class.Method"] 的事件脚本要求域配置文件中具有名为 DomainName 的域。

  必须指定至少一个域名。

- **appBase** 该域用作其应用程序基目录的目录。所有专用程序集都由 .NET CLR 根据该目录装载。必须指定 appBase。

- **configFile** 应该用于该域的 .NET 应用程序配置文件。此元素可以为空。它通常用于修改缺省程序集捆绑和装载行为。有关应用程序配置文件的详细信息，请参见 .NET 文档。

- **assembly** MobiLink 在解析事件脚本中的类型引用时应装载并搜索的程序集的名称。必须指定至少一个程序集。如果多个域中都使用某个程序集，则必须分别在各个域中将其指定为程序集。如果该程序集是专用程序集，则其必须位于该域的应用程序基目录中。

有关 mlsrv11 选项 -sl dnet 的详细信息，请参见“-sl dnet 选项”一节第 86 页。

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

.NET 同步示例

此示例通过修改一个现有应用程序，介绍如何使用 .NET 同步逻辑来处理 authenticate_user 事件。它为 authenticate_user 创建一个名为 *AuthUser.cs* 的 C# 脚本。该脚本在名为 user_pwd_table 的表中查找用户的口令，并根据该口令验证用户的身份。

♦ 创建 .NET 同步脚本

1. 将 user_pwd_table 表添加到数据库中。在 Interactive SQL 中执行以下 SQL 语句：

```
CREATE TABLE user_pwd_table (
  user_name  varchar(128) PRIMARY KEY NOT NULL,
  pwd        varchar(128)
)
```

2. 向该表中添加一个用户及其口令：

```
INSERT INTO user_pwd_table VALUES('user1', 'myPwd')
```

3. 为您的 .NET 程序集创建一个目录。例如，*c:\mlexample*。
4. 创建一个名为 *AuthUser.cs* 的文件，并使该文件具有以下内容：

   请参见“authenticate_user 连接事件”一节第 335 页。

```
using System;
using iAnywhere.MobiLink.Script;
namespace MLExample {

public class AuthClass {
    private DBConnection  _conn;

    /// AuthClass constructor.
    public AuthClass(DBConnectionContext cc) {
        _conn   = cc.GetConnection();
    }

    /// The DoAuthenticate method handles the 'authenticate_user'
    /// event.
    /// Note: This method does not handle password changes for
    /// advanced authorization status codes.
    public void DoAuthenticate(
        ref int authStatus,
        string user,
        string pwd,
        string newPwd)
    {
        DBCommand   pwd_command = _conn.CreateCommand();
        pwd_command.CommandText = "select pwd from user_pwd_table"
            + " where user_name = ? ";
        pwd_command.Prepare();

        // Add a parameter for the user name.
        DBParameter user_param  = new DBParameter();
        user_param.DbType       = SQLType.SQL_CHAR;

        // Set the size for SQL_VARCHAR.
        user_param.Size     = (uint) user.Length;
        user_param.Value    = user;
        pwd_command.Parameters.Add(user_param);
```

```
            // Fetch the password for this user.
            DBRowReader rr      = pwd_command.ExecuteReader();
            object[] pwd_row    = rr.NextRow();

            if (pwd_row == null) {
                // User is unknown.
                authStatus  = 4000;
            }
            else {
                if (((string) pwd_row[0]) == pwd) {
                    // Password matched.
                    authStatus = 1000;
                }
                else {
                    // Password did not match.
                    authStatus = 4000;
                }
            }
            pwd_command.Close();
            rr.Close();
            return;
        }
    }
```

MLExample.AuthClass.DoAuthenticate 方法可处理 authenticate_user 事件。它会接受用户名和口令，并返回指示校验成功或失败的授权状态码。

5. 编译文件 *AuthUser.cs*。可以使用命令行或在 Visual Studio 中进行编译。

   例如，以下命令行会编译文件 *AuthUser.cs*，并在 *c:\mlexample* 中生成名为 *example.dll* 的程序集。

   ```
   csc /out:c:\mlexample\example.dll /target:library /reference:"%SQLANY11%
   \Assembly\v2\iAnywhere.MobiLink.Script.dll" AuthUser.cs
   ```

6. 为 authenticate_user 事件注册 .NET 代码。需要执行的方法 (DoAuthenticate) 位于 MLExample 命名空间和 AuthClass 类中。执行以下 SQL 命令：

   ```
   CALL ml_add_dnet_connection_script('ex_version', 'authenticate_user',
   'MLExample.AuthClass.DoAuthenticate');
   COMMIT
   ```

7. 使用以下选项运行 MobiLink 服务器。使用此选项时，MobiLink 将装载 *c:\myexample* 中的所有程序集：

   ```
   -sl dnet (-MLAutoLoadPath=c:\mlexample)
   ```

现在，当用户与版本 ex_version 同步时，将使用 user_pwd_table 表中的口令对其进行验证。

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

用于 .NET 参考的 MobiLink 服务器 API

本节讲解 MobiLink .NET 接口和类，及其相关方法、属性和构造函数。要使用这些类，可引用 *iAnywhere.MobiLink.Script.dll* 程序集，它位于 *install-dir\Assembly\v2* 中。

本节以 C# 为例进行讲解，但在 Visual Basic.NET 和 C++ 中也有等效项。

DBCommand 接口

语法

interface **DBCommand**
Member of **iAnywhere.MobiLink.Script**

注释

代表一个 SQL 语句或数据库命令。DBCommand 可以代表一个更新或查询。

示例

例如，以下 C# 代码使用 DBCommand 接口执行两个查询：

```
DBCommand stmt = conn.CreateCommand();
stmt.CommandText = "SELECT t1a1, t1a2 FROM table1 ";

DBRowReader rs = stmt.ExecuteReader();
printResultSet(rs);
rs.Close();

stmt.CommandText = "SELECT t2a1 FROM table2 ";
rs = stmt.ExecuteReader();
printResultSet(rs);
rs.Close();
stmt.Close();
```

以下 C# 示例使用 DBCommand 执行带有参数的更新：

```
public void prepare_for_download(
    DateTime last_download,
    String ml_username)
{
    DBCommand cstmt     = conn.CreateCommand();
    cstmt.CommandText   = "CALL myProc(?,?,?,?)";
    cstmt.Prepare();

    DBParameter param   = new DBParameter();
    param.DbType        = SQLType.SQL_CHAR;
    param.Value         = "10000";
    cstmt.Parameters.Add(param);

    param               = new DBParameter();
    param.DbType        = SQLType.SQL_INTEGER;
    param.Value         = 20000;
    cstmt.Parameters.Add(param);

    param               = new DBParameter();
```

```
        param.DbType        = SQLType.SQL_DECIMAL;
        param.Precision     = 5;
        param.Value         = new Decimal(30000);
        cstmt.Parameters.Add(param);

        param               = new DBParameter();
        param.DbType        = SQLType.SQL_TIMESTAMP;
        param.Precision     = 19;
        param.Value         = last_download;
        cstmt.Parameters.Add(param);

        // Execute update
        DBRowReader rset    = cstmt.ExecuteNonQuery();
        cstmt.Close();
    }
```

Prepare 方法

语法

void **Prepare( )**

注释

为执行存储在 CommandText 中的 SQL 语句进行准备。

ExecuteNonQuery 方法

语法

int **ExecuteNonQuery( )**

注释

执行非查询语句。返回数据库中受该 SQL 语句影响的行数。

ExecuteReader 方法

语法

DBRowReader **ExecuteReader( )**

注释

执行一条查询语句，返回结果集。返回一个 DBRowReader，用于检索由该 SQL 语句返回的结果。

Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

Close 方法

语法

void **Close( )**

注释

关闭当前 SQL 语句或命令。

CommandText 属性

语法

string **CommandText**

注释

其值是要执行的 SQL 语句。

Parameters 属性

语法

DBParameterCollection **Parameters**

注释

获取该 DBCommand 的 iAnywhere.MobiLink.Script.DBParameterCollection。

DBConnection 接口

语法

interface **DBConnection**
Member of **iAnywhere.MobiLink.Script**

注释

代表 MobiLink ODBC 连接。

用户编写的同步逻辑可以通过此接口访问由 MobiLink 创建的 ODBC 连接。

Commit 方法

语法

void **Commit( )**

注释

提交当前事务。

Rollback 方法

语法

void **Rollback( )**

注释

回退当前事务。

Close 方法

语法

void **Close( )**

注释

关闭当前连接。

CreateCommand 方法

语法

DBCommand **CreateCommand( )**

注释

在该连接上创建一条 SQL 语句或命令。返回新生成的 DBCommand。

DBConnectionContext 接口

语法

interface **DBConnectionContext**
Member of **iAnywhere.MobiLink.Script**

注释

用于获取和访问当前数据库连接相关信息的接口。此接口将被传递到包含脚本的类的构造函数。如果后台线程或在连接生存期以外要求使用上下文，请使用 ServerContext。

有关构造函数的详细信息，请参见“构造函数”一节第 556 页。

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

小心
DBConnectionContext 实例不能在调用 .NET 代码的线程之外使用。

GetConnection 方法

语法

iAnywhere.MobiLink.Script.DBConnection GetConnection( )
Member of **iAnywhere.MobiLink.Script.DBConnectionContext**

注释

返回与 MobiLink 统一数据库之间的现有连接。该连接与 MobiLink 用以执行 SQL 脚本的连接相同。

不能以任何将会影响到 MobiLink 服务器对该连接的使用的方式提交、关闭或变更该连接。返回的连接仅在基础 MobiLink 连接生存期内有效。在为该连接调用 end_connection 事件了之后，不要使用该连接。

如果需要使用可执行完全访问的服务器连接，请使用 ServerContext.makeConnection()。

GetDownloadData 方法

语法

DownloadData GetDownloadData();

注释

返回当前同步的 DownloadData 实例。使用 DownloadData 实例为直接行处理创建下载。

返回值

当前同步的 DownloadData 实例。

示例

以下示例假定您拥有一个名为 _cc 的 DBConnectionContext 实例。

```
// The method used for the handle_DownloadData event.
public void HandleDownload() {
    // Get the DownloadData for the current synchronization.
    DownloadData my_dd = _cc.GetDownloadData();

    // Get an array of tables to set download operations.
    DownloadTableData[] download_tables = my_dd.GetDownloadTables();

    // Get the first table in the DownloadTableData array.
    DownloadTableData my_download_table = download_tables[0];

     // ...
}
```

GetServerContext 方法

语法

public **iAnywhere.MobiLink.Script.ServerContext.GetServerContext( )**
Member of **iAnywhere.MobiLink.Script.DBConnectionContext**

注释

为 MobiLink 服务器返回 ServerContext。

GetProperties 方法

语法

NameValueCollection **getProperties( )**

注释

基于此连接的脚本版本返回此连接的属性。属性存储在 ml_property 表中。

有关详细信息，请参见“ml_property”一节第 670 页和“ml_add_property 系统过程”一节第 637 页。

GetRemoteID 方法

语法

string **GetRemoteID( )**

注释

返回当前正在此连接上同步的数据库的远程 ID。如果您的远程数据库为版本 10 之前的版本，则其将返回 MobiLink 用户名。

返回值

远程 ID。

另请参见

- “远程 ID”一节 《MobiLink - 客户端管理》

GetVersion 方法

语法

string **getVersion( )**

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

注释

返回该脚本版本的名称。

另请参见

- “ml_property”一节第 670 页
- “ml_add_property 系统过程”一节第 637 页

DBParameter 类

语法

class **DBParameter**
Member of **iAnywhere.MobiLink.Script**

注释

代表捆绑的 ODBC 参数。

要执行带有参数的命令，需要 DBParameter。执行该命令前，必须准备好所有参数。

示例

例如，以下 C# 代码使用 DBCommand 执行带有参数的更新：

```
public void handleUpload(UploadData ud) {
    UploadedTableData UTDAdmin = ud.GetUploadedTableByName("Admin");
    IDataReader       AdminIns = UTDAdmin.GetInserts();
    DBCommand         stmt1    = _conn.CreateCommand();
    DBParameter       p_id     = new DBParameter();
    DBParameter       p_data   = new DBParameter();

    stmt1.CommandText          = "INSERT INTO Admin(admin_id,data) VALUES
(?,?)";
    p_id.DbType                = SQLType.SQL_BIGINT;
    stmt1.Parameters.Add(p_id);
    p_data.DbType              = SQLType.SQL_VARCHAR;
    p_data.Size                = 30;
    stmt1.Parameters.Add(p_data);

    stmt1.Prepare();

    while (AdminIns.Read()) {
        p_id.Value = AdminIns.GetInt64(0);
        p_data.Value = AdminIns.GetString(1);
        stmt1.ExecuteNonQuery();
    }
    stmt1.Close();
}
```

DbType 属性

语法

SQLTYPE DbType

注释

其值是该参数的 SQLType。

缺省值为 SQLType.SQL_TYPE_NULL。

Direction 属性

语法

System.Data.ParameterDirection Direction

注释

其值是该参数的输入/输出方向。

缺省值为 ParameterDirection.Input。

IsNullable 属性

语法

bool **IsNullable**

注释

其值表示该参数是否可以为空。

缺省值为 false。

ParameterName 属性

语法

string **ParameterName**

注释

其值是该参数的名称。

缺省值为空。

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

Precision 属性

语法

uint **Precision**

注释

其值是该参数的小数精度。仅用于 SQLType.SQL_NUMERIC 和 SQLType.SQL_DECIMAL 参数。

缺省值是 0。

Scale 属性

语法

short **Scale**

注释

其值是该参数的可解析位数。仅用于 SQLType.SQL_NUMERIC 和 SQLType.SQL_DECIMAL 参数。

缺省值是 0。

Size 属性

语法

uint **Size**

注释

其值是该参数的大小（以字节为单位）。

缺省值是从 DbType 推导出来的。

Value 属性

语法

object **Value**

注释

其值是该参数的值。

缺省值为空。

DBParameterCollection 类

语法

class **DBParameterCollection**
inherits from **IDataParameterCollection**, **IList**, **ICollection**, **IEnumerable**
Member of **iAnywhere.MobiLink.Script**

注释

DBParameters 的集合。当 DBCommand 创建 DBParameterCollection 时，它是一个空集合，且必须在执行 DBCommand 前，用适当的参数填充该集合。

DBParameterCollection 方法

语法

DBParameterCollection( )

注释

创建 DBParameters 的空列表。

Contains(string parameterName) 方法

语法

bool **Contains(string** *parameterName* **)**

注释

如果该集合包含具有指定名称的参数，此方法返回 True。

参数

- **parameterName** 要检查的参数的名称。

IndexOf(string parameterName) 方法

语法

int **IndexOf(string** *parameterName* **)**

注释

返回该参数的索引，或者，如果没有任何参数具有给定的名称，则返回 -1。

参数

- **parameterName** 要查找的参数的名称。

Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

RemoveAt(string parameterName) 方法

语法

void **RemoveAt(string** *parameterName* **)**

注释

从该集合中删除具有给定名称的参数。

参数

- **parameterName** 要删除的参数的名称。

Add(object value) 方法

语法

int **Add(object** *value* **)**

注释

将给定参数添加到该集合中。

参数

- **value** 要添加到该集合中的 iAnywhere.MobiLink.Script.DBParameter 实例。

返回值

添加的参数在该集合中的索引。

Clear 方法

语法

void **Clear( )**

注释

删除该集合中的所有参数。

Contains(object value) 方法

语法

bool **Contains(object** *value* **)**

注释

如果该集合包含给定的 iAnywhere.MobiLink.Script.DBParameter，则会返回 True。

Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

参数

- **value** 要检查的 iAnywhere.MobiLink.Script.DBParameter。

IndexOf(object value) 方法

语法

int **IndexOf(object** *value* **)**

注释

返回给定 iAnywhere.MobiLink.Script.DBParameter 在该集合中的索引。

参数

- **value** 要查找的 iAnywhere.MobiLink.Script.DBParameter。

Insert(int index, object value) 方法

语法

void **Insert(int** *index*, **object** *value* **)**

注释

将给定的 iAnywhere.MobiLink.Script.DBParameter 插入到该集合中指定的索引位置。

参数

- **value** 要插入的 iAnywhere.MobiLink.Script.DBParameter。
- **index** 插入 DBParameter 的索引。

Remove(object value) 方法

语法

void **Remove(object** *value* **)**

注释

从该集合中删除给定的 iAnywhere.MobiLink.Script.DBParameter。

参数

- **value** 要删除的 iAnywhere.MobiLink.Script.DBParameter。

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

RemoveAt(int index) 方法

语法

void **RemoveAt(int index)**

注释

删除该集合中位于给定索引位置的 iAnywhere.MobiLink.Script.DBParameter。

参数

- **index** 要删除的 iAnywhere.MobiLink.Script.DBParameter 的索引。

CopyTo(Array array, int index) 方法

语法

void **CopyTo(Array** *array*, **int** *index* **)**

注释

从指定的索引位置开始，将该集合的内容复制到给定的数组中。

参数

- **array** 将该集合的内容复制到其中的数组。
- **index** 开始复制该集合内容的数组中的索引。

GetEnumerator 方法

语法

IEnumerator **GetEnumerator( )**

注释

返回该集合的枚举器。

IsFixedSize 属性

语法

bool **IsFixedSize**

注释

返回 False。

IsReadOnly property

语法

bool **IsReadOnly**

注释

返回 False。

Count 属性

语法

int **Count**

注释

该集合中参数的数量。

IsSynchronized 属性

语法

bool **IsSynchronized**

注释

返回 False。

SyncRoot 属性

语法

object **SyncRoot**

注释

可用于同步该集合的对象。

this[string parameterName] 属性

语法

object **this[string** *parameterName* **]**

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

注释

获取或设置该集合中具有给定名称的 iAnywhere.MobiLink.Script.DBParameter。

参数

- **parameterName** 要获取或设置的 iAnywhere.MobiLink.Script.DBParameter 的名称。

this[int index] 属性

语法

object **this[int** *index* **]**

注释

获取或设置该集合中位于给定索引位置的 iAnywhere.MobiLink.Script.DBParameter。

参数

- **index** 要获取或设置的 iAnywhere.MobiLink.Script.DBParameter 的索引。

DBRowReader 接口

语法

interface **DBRowReader**
Member of **iAnywhere.MobiLink.Script**

注释

代表正从数据库中读取的一组行。执行 DBCommand.executeReader( ) 方法可创建一个 DBRowReader。

以下示例是一段 C# 代码。它用给定 DBRowReader 所代表的结果集中的行调用一个函数。

```
DBCommand stmt = conn.CreateCommand();
stmt.CommandText = "select intCol, strCol from table1 ";
DBRowReader rset = stmt.ExecuteReader();
object[] values = rset.NextRow();

while (values != null) {
    handleRow((int) values[0], (String) values[1]);
    values = rset.NextRow();
}

rset.Close();
stmt.Close();
```

NextRow 方法

语法

object[ ] **NextRow( )**

注释

检索并返回结果集中下一行的值。如果已到达结果集的最后一行，它将返回空值。

请参见“SQLType 枚举”一节第 593 页。

Close 方法

语法

void **Close( )**

注释

清除该 MLDBRowReader 使用的资源。调用 Close( ) 后，就不能再使用该 MLDBRowReader。

ColumnNames 属性

语法

string[ ] **ColumnNames**

注释

获取结果集中所有列的名称。其值是对应于结果集中各列的名称的字符串数组。

ColumnTypes 属性

语法

SQLType[ ] **ColumnTypes**

注释

获取结果集中所有列的类型。其值是对应于结果集各列的类型的 SQLTypes 数组。

DownloadData 接口

语法

interface **DownloadData**

Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

注释

封装用于直接行处理的下载数据操作。要获取 DownloadData 实例，请使用 DBConnectionContext GetDownloadData 方法。使用 DownloadData.GetDownloadTables 和 GetDownloadTableByName 方法返回 DownloadTableData 实例。

可通过 DBConnectionContext 使用此下载数据。在 begin_synchronization 事件之前或 end_download 事件之后无法访问下载数据。无法访问仅上载同步中的 DownloadData。

另请参见

- “DownloadTableData 接口”一节第 584 页
- “handle_DownloadData 连接事件”一节第 410 页
- DBConnectionContext “GetDownloadData 方法”一节第 571 页
- “直接行处理”第 609 页

GetDownloadTables 方法

语法

DownloadTableData[ ] **GetDownloadTables( );**

注释

获取此同步中要下载数据的所有表的数组。对此表执行的操作会发送到远程数据库。

返回值

下载表数据的数组。数组中表的顺序与远程的上载顺序相同。

示例

以下示例使用 DownloadData.GetDownloadTables 方法为当前同步获取 DownloadTableData 对象的一个数组。本示例假定您拥有一个名为 _cc 的 DBConnectionContext 实例。

```
// The method used for the handle_DownloadData event.
public void HandleDownload() {
    // Get the DownloadData for the current synchronization.
    DownloadData my_dd = _cc.GetDownloadData();

    // Get an array of tables to set download operations.
    DownloadTableData[] download_tables = my_dd.GetDownloadTables();

    // Get the first table in the DownloadTableData array.
    DownloadTableData my_download_table = download_tables[0];

    // ...
}
```

GetDownloadTableByName 方法

语法

DownloadTableData GetDownloadTableByName(
string *table-name*);

注释

获取此同步的指定下载表。如果此同步中不存在具有指定名称的表，则会返回空值。

返回值

给定表名的下载数据或为空值（如果未找到）。

参数

- **table-name** 您所需要的下载数据所在的表的名称。

DownloadTableData 接口

语法

interface DownloadTableData

注释

为一个同步封装一个下载表的信息。使用此接口设置将被下载到同步客户端站点的数据操作。

示例

例如，假设您具有以下表：

```
CREATE TABLE remoteOrders (
    pk INT NOT NULL,
    col1 VARCHAR(200),
    PRIMARY KEY (pk)
);
```

以下示例使用 DownloadData.GetDownloadTableByName 方法返回表示 remoteOrders 表的 DownloadTableData 实例。

```
// The method used for the handle_DownloadData event
public void HandleDownload() {
    // _cc is a DBConnectionContext instance.

    // Get the DownloadData for the current synchronization.
    DownloadData my_dd = _cc.GetDownloadData();

    // Get the DownloadTableData for the remoteOrders table.
    DownloadTableData td = my_dd.GetDownloadTableByName("remoteOrders");

    // User defined-methods to set download operations.
    SetDownloadUpserts(td);
    SetDownloadDeletes(td);
```

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

```
    // ...
}
```

在本例中，SetDownloadInserts 方法使用 DownloadTableData.GetUpsertCommand 为您想要插入或更新的行获取命令。IDbCommand 将会保存一些参数，这些参数将会被设置为您要插入到远程数据库中的值。

```
void SetDownloadInserts(DownloadTableData td) {
    IDbCommand  upsert_cmd = td.GetUpsertCommand();
    IDataParameterCollection parameters = upsert_cmd.Parameters;

    // The following method calls are the same as the following SQL
statement:
    // INSERT INTO remoteOrders(pk, col1) values(2300, "truck");
    ((IDataParameter) (parameters[0])).Value = (Int32) 2300;
    ((IDataParameter) (parameters[1])).Value = (String) "truck";

    if (upsert_cmd.ExecuteNonQuery() > 0) {
        // Insert was not filtered.
    }
    else {
        // Insert was filtered because it was uploaded
        // in the same synchronization.
    }
 }
```

SetDownloadDeletes 方法使用 DownloadTableData.GetDeleteCommand 为您想要删除的行获取命令。

```
void SetDownloadDeletes(DownloadTableData td) {
    IDbCommand  delete_cmd = t2_download_dd.GetDeleteCommand();

    // The following method calls are the same as the following SQL
statement:
    // DELETE FROM remoteOrders  where pk = 2300;
    IDataParameterCollection parameters = delete_cmd.Parameters;
    ((IDataParameter) (parameters[0])).Value = (Int32) 2300;
    delete_cmd.ExecuteNonQuery();
}
```

GetDeleteCommand 方法

语法

IDbCommand **GetDeleteCommand();**

注释

获取一条命令，它允许用户将删除操作添加到下载数据操作中。返回命令的参数数量与该表主键列的参数数量相同。对于包括在下载中的删除操作，必须设置主键列的列值，并且必须使用 ExecuteNonQuery() 来执行语句。

> **注意**
> 您必须为下载删除操作设置所有主键值。

返回值

下载中用于删除操作的命令。

示例

请参见“DownloadTableData 接口”一节第 584 页。

GetLastDownloadTime 方法

语法

DateTime **GetLastDownloadTime();**

注释

返回此表的上次下载时间。传递给几个表格下载事件的上次下载时间是相同的。

上次下载时间对于生成用于某个特定同步的表下载数据很有用。

返回值

此下载表的上次下载时间。

GetName 方法

语法

string **GetName();**

注释

获取该实例的表名。这是一个实用程序函数。还可通过该实例的模式来访问表名。

返回值

该实例的表名。

GetSchemaTable 方法

语法

DataTable **GetSchemaTable();**

注释

获取一个描述该下载表元数据的 DataTable 实例。

如果希望 DataTable 包含列名信息，则必须指定客户端选项以发送列名。

返回值

描述列元数据的 DataTable。

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

另请参见

- dbmlsync： “SendColumnNames (scn) 扩展选项”一节 《MobiLink - 客户端管理》
- UltraLite： “Send Column Names 同步参数”一节 《UltraLite - 数据库管理和参考》

GetUpsertCommand 方法

语法

IDbCommand **GetUpsertCommand();**

注释

获取一条命令，该命令允许您将 upsert（插入/更新）操作添加到直接下载数据操作中。返回命令的参数数量与该表中的列的参数数量相同。必须设置插入操作的列值，并且必须使用 ExecuteNonQuery() 执行语句以将插入/更新包括在下载中。如果插入/更新操作被过滤，则命令中的 ExecuteNonQuery() 将返回 0；如果插入/更新操作未被过滤，则将返回 1。

不能向该命令添加或删除参数；只能设置它们的值。

返回值

用于下载的插入/更新命令。

示例

请参见“DownloadTableData 接口”一节第 584 页。

LogCallback 委派

语法

delegate void **LogCallback(**
  ServerContext *sc*
  LogMessage *message*
)
Member of **iAnywhere.MobiLink.Script**

注释

当 MobiLink 服务器输出消息时被调用。

LogMessage 类

语法

class **LogMessage** : **iAnywhere.MobiLink.Script.LogMessage**
Member of **iAnywhere.MobiLink.Script**

注释

包含有关输出到日志的消息的信息。

MessageType 枚举

语法

enum **MessageType**
Member of **iAnywhere.MobiLink.Script.LogMessage**

注释

LogMessage 的可能的类型的枚举。

ERROR 字段

语法

ERROR

注释

日志消息是属于错误。

INFO 字段

语法

INFO

注释

日志信息消息。

WARNING 字段

语法

WARNING

注释

日志消息属于警告。

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

Type 属性

语法

LogMessage.MessageType **Type**

注释

此实例代表的日志消息的类型。

User 属性

语法

string **User**

注释

为之记录消息的用户。此属性可以为空。

Text 属性

语法

string **Text**

注释

该消息的正文文本。

ServerContext 接口

语法

interface **ServerContext**
Member of **iAnywhere.MobiLink.Script**

注释

MobiLink 服务器工作期间出现的所有上下文的实例化。上下文可以作为静态数据保存，并在后台线程中使用。它在 MobiLink 调用的 .NET CLR 期间有效。

要访问 ServerContext 实例，请使用 DBConnectionContext.getServerContext 方法。

GetStartClassInstances 方法

语法

object[ ] GetStartClassInstances( )
Member of **iAnywhere.MobiLink.Script.ServerContext**

注释

获取包含服务器启动时所构造的启动类的数组。如果没有启动类，此数组的长度为零。

有关用户定义的启动类的详细信息，请参见“用户定义的启动类”一节第 557 页。

以下是 getStartClassInstances() 方法的示例：

```
void FindStartClass(ServerContext sc, string name) {
    object[] startClasses = sc.GetStartClassInstances();
    foreach (object obj in startClasses) {
        if (obj is MyClass) {
           // Execute some code.
        }
    }
}
```

LogCallback ErrorListener 事件

MobiLink 服务器输出错误时将触发此事件。

LogCallback InfoListener 事件

MobiLink 服务器输出信息时将触发此事件。

LogCallback WarningListener 事件

MobiLink 服务器输出警告时将触发此事件。

MakeConnection 方法

语法

iAnywhere.MobiLink.Script.DBConnection makeConnection( )
Member of **iAnywhere.MobiLink.Script.ServerContext**

注释

创建新的服务器连接。

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

ShutDown 方法

语法

void **Shutdown( )**
Member of **iAnywhere.MobiLink.Script.ServerContext**

注释

强制服务器关闭。

ShutdownListener 方法

语法

event iAnywhere.MobiLink.Script.ShutdownCallback
ShutdownListener(
iAnwyhere.MobiLink.Script.ServerContext *sc*)
Member of **iAnywhere.MobiLink.Script.ServerContext**

注释

服务器关闭时将触发此事件。以下代码示例说明如何使用此事件：

```
ShutdownCallback callback = new ShutdownCallback(shutdownHandler);
_sc.ShutdownListener += callback;

public void shutdownHandler(ServerContext sc) {
    _test_out_file.WriteLine("shutdownPerformed");
}
```

getProperties 方法

语法

NameValueCollection **getProperties(**
string *component_name*
string *prop_set_name* **)**

注释

返回与此脚本版本关联的一组属性。这些属性存储在 MobiLink 系统表 ml_property 中。

另请参见

- “ml_property”一节第 670 页
- “ml_add_property 系统过程”一节第 637 页

Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

getPropertiesByVersion 方法

语法

NameValueCollection **getPropertiesByVersion(** string *script_version* **)**

注释

返回与此脚本版本关联的一组属性。这些属性存储在 MobiLink 系统表 ml_property 中。当 component_name 为 ScriptVersion 时，脚本版本存储在 property_set_name 列中。

另请参见

- “ml_property”一节第 670 页
- “ml_add_property 系统过程”一节第 637 页

getPropertySetNames 方法

语法

StringCollection **getPropertySetNames(** string *component_name* **)**

注释

返回给定组件的属性集名称列表。这些属性存储在 MobiLink 系统表 ml_property 中。

另请参见

- “ml_property”一节第 670 页
- “ml_add_property 系统过程”一节第 637 页

ServerException 类

语法

public class **ServerException**
Member of **iAnywhere.MobiLink.Script**

注释

用于通知 MobiLink 服务器发生错误，应立即将其关闭。

ServerException 构造函数

语法

public **ServerException( )**
Member of **iAnywhere.MobiLink.Script.ServerException**

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

注释

构造一个不带任何细节消息的 ServerException。

语法

public **ServerException(string** *message* **)**
Member of **iAnywhere.MobiLink.Script.ServerException**

注释

创建一个带有给定消息的新 ServerException。

参数

- **message** 此 ServerException 的消息。

语法

public **ServerException(string** *message*, **SystemException** *ie* **)**
Member of **iAnywhere.MobiLink.Script.ServerException**

注释

创建一个带有给定消息并包含导致该异常的给定内部异常的新 ServerException。

参数

- **message** 此 ServerException 的消息。
- **ie** 导致该 ServerException 的异常。

ShutdownCallback 委派

语法

sealed delegate **ShutdownCallback : System.MulticastDelegate**
Member of **iAnywhere.MobiLink.Script**

注释

当 MobiLink 服务器关闭时，将调用此委派。可通过在 MobiLink 服务器关闭时调用的 ServerContext.ShutdownListener 事件，来注册此委派的实现。

SQLType 枚举

语法

enum **SQLType**
Member of **iAnywhere.MobiLink.Script**

注释

所有可能的 ODBC 数据类型的枚举。

SQL_TYPE_NULL 字段

语法

SQL_TYPE_NULL

注释

Null 数据类型。

SQL_UNKNOWN_TYPE 字段

语法

SQL_UNKNOWN_TYPE

注释

Unknown 数据类型。

SQL_CHAR 字段

语法

SQL_CHAR

注释

单字节字符串。类型为 .NET String。

SQL_NUMERIC 字段

语法

SQL_NUMERIC

注释

固定大小和精度的数字值。类型为 .NET Decimal。

SQL_DECIMAL 字段

语法

SQL_DECIMAL

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

注释

固定大小和精度的十进制数。类型为 .NET Decimal。

SQL_INTEGER 字段

语法

SQL_INTEGER

注释

32 位整数。类型为 .NET Int32。

SQL_SMALLINT 字段

语法

SQL_SMALLINT

注释

16 位整数。类型为 .NET Int16。

SQL_FLOAT 字段

语法

SQL_FLOAT

注释

精度由 ODBC 驱动程序定义的浮点数。类型为 .NET Double。

SQL_REAL 字段

语法

SQL_REAL

注释

单精度浮点数。类型为 .NET Single。

SQL_DOUBLE 字段

语法

SQL_DOUBLE

注释

双精度浮点数。类型为 .NET Double。

SQL_DATE 字段

语法

SQL_DATE

注释

日期。类型为 .NET DateTime。

SQL_DATETIME 字段

语法

SQL_DATETIME

注释

日期和时间。类型为 .NET DateTime。

SQL_TIME 字段

语法

SQL_TIME

注释

时间。类型为 .NET DateTime。

SQL_INTERVAL 字段

语法

SQL_INTERVAL

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

注释

时间间隔。类型为 .NET TimeSpan。

SQL_TIMESTAMP 字段

语法

SQL_TIMESTAMP

注释

时间戳。类型为 .NET DateTime。

SQL_VARCHAR 字段

语法

SQL_VARCHAR

注释

单字节字符串。类型为 .NET String。

SQL_TYPE_DATE 字段

语法

SQL_TYPE_DATE

注释

日期。类型为 .NET DateTime。

SQL_TYPE_TIME 字段

语法

SQL_TYPE_TIME

注释

时间。类型为 .NET DateTime。

Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

SQL_TYPE_TIMESTAMP 字段

语法

SQL_TYPE_TIMESTAMP

注释

时间戳。类型为 .NET DateTime。

SQL_DEFAULT 字段

语法

SQL_DEFAULT

注释

缺省类型。没有类型。

SQL_ARD_TYPE 字段

语法

SQL_ARD_TYPE

注释

ARD 对象。没有类型。

SQL_BIT 字段

语法

SQL_BIT

注释

一个数位。类型为 .NET Boolean。

SQL_TINYINT 字段

语法

SQL_TINYINT

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

注释

8 位整数。类型为 .NET SByte。

SQL_BIGINT 字段

语法

SQL_BIGINT

注释

64 位整数。类型为 .NET Int64。

SQL_LONGVARBINARY 字段

语法

SQL_LONGVARBINARY

注释

可变长度二进制数据，其最大长度取决于驱动程序。类型为 .NET byte[ ]。

SQL_VARBINARY 字段

语法

SQL_VARBINARY

注释

可变长度二进制数据，其最大长度由用户指定。类型为 .NET byte[ ]。

SQL_BINARY 字段

语法

SQL_BINARY

注释

固定长度的二进制数据。类型为 .NET byte[ ]。

Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

SQL_LONGVARCHAR 字段

语法

SQL_LONGVARCHAR

注释

单字节字符串。类型为 .NET String。

SQL_GUID 字段

语法

SQL_GUID

注释

全局唯一 ID（也称作 UUID）。类型为 .NET Guid。

SQL_WCHAR 字段

语法

SQL_WCHAR

注释

固定大小的 Unicode 字符数组。类型为 .NET String。

SQL_WVARCHAR 字段

语法

SQL_WVARCHAR

注释

以空值结尾的 Unicode 字符串，其最大长度由用户定义。类型为 .NET String。

SQL_WLONGVARCHAR 字段

语法

SQL_WLONGVARCHAR

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

注释

以空值结尾的 Unicode 字符串，其最大长度取决于驱动程序。类型为 .NET String。

SynchronizationException 类

语法

class **SynchronizationException**
Member of **iAnywhere.MobiLink.Script**

注释

用于通知发生同步异常，应回退并重新启动当前同步。

SynchronizationException 构造函数

语法

SynchronizationException( )
Member of **iAnywhere.MobiLink.Script.SynchronizationException**

注释

构造不带有任何详细消息的 SynchronizationException。

语法

public **SynchronizationException(string** *message* **)**
Member of **iAnywhere.MobiLink.Script.SynchronizationException**

注释

创建一个带有给定消息的新 SynchronizationException。

参数

- **message** 此 ServerException 的消息。

语法

SynchronizationException(string *message*, **SystemException** *ie* **)**
Member of **iAnywhere.MobiLink.Script.SynchronizationException**

注释

创建一个带有给定消息并包含导致该异常的内部异常的新 SynchronizationException。

参数

- **message** 此 ServerException 的消息。
- **ie** 导致该 ServerException 的异常。

UploadData 接口

语法

public interface **UploadData**

注释

封装用于直接行处理的上载操作。上载事务包含一组包含行操作的表。表示单个上载事务的 UploadData 实例被传递给 handle_UploadData 同步事件。

> **小心**
> 您必须在针对 handle_UploadData 事件注册的方法中处理直接行处理上载操作。UploadData 在每次调用注册的方法后即被取消。不要创建 UploadData 的新实例以在后续事件中使用。

使用 UploadData.GetUploadedTables 或 UploadData.GetUploadedTableByName 方法来获取 UploadedTableData 实例。

除非远程数据库采用事务性上载，否则一个同步将有一个 UploadData。

另请参见

- “直接行处理”第 609 页
- “UploadedTableData 接口”一节第 604 页
- “handle_UploadData 连接事件”一节第 422 页
- “处理直接上载”一节第 614 页

示例

请参见“handle_UploadData 连接事件”一节第 422 页。

GetUploadedTableByName 方法

语法

UploadedTableData **GetUploadedTableByName(**
  string *table-name*);

注释

在此上载事务中获取指定的上载表数据。如果此事务中不存在具有给定名称的表，则会返回空值。

参数

- **table-name**　您所需要的上载数据所在的表的名称。

返回值

给定表名的上载数据或空值（如果未找到）。

Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

示例

假定您将一个名为 HandleUpload 的方法用于 handle_UploadData 同步事件。以下示例使用 GetUploadedTableByName 方法返回 remoteOrders 表的 UploadedTableData 实例。

```
// The method used for the handle_UploadData event.
public void HandleUpload(UploadData ut) {
    UploadedTableData uploaded_t1 =
ut.GetUploadedTableByName("remoteOrders");
    // ...
}
```

GetUploadedTables 方法

语法

UploadedTableData[ ] **GetUploadedTables();**

注释

在此上载事务中获取所有上载表数据的数组。数组中表的顺序与 MobiLink 在进行 SQL 的行处理时所采用的顺序相同，这一顺序能最有效地防止参照完整性违规。如果您的数据源是关系数据库，请使用此表顺序。

返回值

上载表数据的数组。数组中表的顺序与客户端的上载顺序相同。

示例

假定您将一个名为 HandleUpload 的方法用于 handle_UploadData 同步事件。以下示例使用 GetUploadedTables 方法为当前上载事务返回 UploadedTableData 实例。

```
// The method used for the handle_UploadData event.
public void HandleUpload(UploadData ud) {
    UploadedTableData[] tables = ud.GetUploadedTables();
    //...
}
```

UpdateDataReader 接口

语法

interface **UpdateDataReader**

注释

为一个表保存一个上载事务的更新操作。通过将 DataReader 的模式更改为旧或新，可以对新行和旧行进行访问。否则，可将其用作常规 DataReader。

SetNewRowValues 方法

语法

void **SetNewRowValues();**

注释

将此 DataReader 的模式设置为返回新列值（更新后的行）。这是缺省模式。

SetOldRowValues 方法

语法

void **SetOldRowValues();**

注释

将此 DataReader 的模式设置为返回旧列值（更新前的行）。

UploadedTableData 接口

语法

public interface **UploadedTableData**

注释

为一个同步封装一个上载表的信息。

插入、更新和删除操作均可通过标准 ADO.NET IDataReader 进行访问。表的元数据可通过 GetSchemaTable() 调用或插入和删除数据读取器进行访问。删除数据读取器只包括该表的主键列。

GetDeletes 方法

语法

IDataReader **GetDeletes();**

注释

为此上载表数据获取带有删除操作的 DataReader。每个删除操作均以那些需要唯一代表本实例表中的行的主键值来表示。

注意：各列的索引和顺序与用于该表模式的 DataTable.PrimaryKey 属性数组相匹配。

返回值

带有删除行的主键列的 DataReader。

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

示例

假定远程客户端包含一个名为 sparse_pk 的表。以下示例使用 DownloadTableData.GetDeletes 方法来获取删除行的数据读取器。在此情况下，删除 datareader 包括两个主键列。注意每个主键列的索引。

```
CREATE TABLE sparse_pk (
    pcol1  INT NOT NULL,
    col2 VARCHAR(200),
    pcol3 INT NOT NULL,
    PRIMARY KEY (pcol1, pcol3)
);

using iAnywhere.MobiLink.Script;
using System;
using System.IO;
using System.Data;
using System.Text;
...

// The method used for the handle_UploadData event.
public void HandleUpload(UploadData ut) {

    // Get an UploadedTableData for the sparse_pk table.
    UploadedTableData sparse_pk_table =
ut.GetUploadedTableByName("sparse_pk");

    // Get deletes uploaded by the MobiLink client.
    IDataReader data_reader = sparse_pk_table.GetDeletes();

    while (data_reader.Read()) {
        StringBuilder row_str = new StringBuilder("( ");
        row_str.Append(data_reader.GetString(0)); // pcol1
        row_str.Append(", ");
        row_str.Append(data_reader.GetString(1)); // pcol3
        row_str.Append(" )");
        writer.WriteLine(row_str);
    }
    data_reader.Close();
}
```

GetInserts 方法

语法

IDataReader **GetInserts();**

注释

为此上载表数据获取带有插入操作的 DataReader。每个插入都以读取器返回的一个行来表示。

返回值

用于此表数据的带有插入操作的 DataReader。

示例

```
CREATE TABLE sparse_pk (
    pcol1  INT NOT NULL,
```

```
    col2 VARCHAR(200),
    pcol3 INT NOT NULL,
    PRIMARY KEY (pcol1, pcol3)
);

using iAnywhere.MobiLink.Script;
using System;
using System.IO;
using System.Data;
using System.Text;
...

// The method used for the handle_UploadData event.
public void HandleUpload(UploadData ut) {

    // Get an UploadedTableData for the sparse_pk table.
    UploadedTableData sparse_pk_table =
ut.GetUploadedTableByName("sparse_pk");

    // Get deletes uploaded by the MobiLink client.
    IDataReader data_reader = sparse_pk_table.GetInserts();

    while (data_reader.Read()) {
        StringBuilder row_str = new StringBuilder("( ");
        row_str.Append(data_reader.GetString(0)); // pcol1
        row_str.Append(", ");
        if (data_reader.IsDBNull(1)) {
            row_str.Append("<NULL>");
        }
        else {
            row_str.Append(data_reader.GetString(1)); // col2
        }
        row_str.Append(", ");
        row_str.Append(data_reader.GetString(2)); // pcol3
        row_str.Append(" )");
        writer.WriteLine(row_str);
    }
    data_reader.Close();
}
```

GetName 方法

语法

string **GetName();**

注释

获取该实例的表名。这是一个实用程序函数。还可通过该实例的模式来访问表名。

返回值

该实例的表名。

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

GetSchemaTable 方法

语法

DataTable **GetSchemaTable();**

注释

获取一个描述该下载表元数据的 DataTable。

如果希望 DataTable 包含列名信息，则必须指定客户端选项以发送列名。

返回值

描述列元数据的 DataTable。

另请参见

- dbmlsync： “SendColumnNames (scn) 扩展选项”一节 《MobiLink - 客户端管理》
- UltraLite： “Send Column Names 同步参数”一节 《UltraLite - 数据库管理和参考》

GetUpdates 方法

语法

UpdateDataReader **GetUpdates();**

注释

为此上载表数据获取带有更新操作的 DataReader。结果集中的每个行代表一个更新。结果集的模式可以在新旧列值之间进行切换。

返回值

用于此表数据的带有更新操作的 DataReader。

示例

以下示例说明了如何使用 GetUpdates 方法。

```
CREATE TABLE sparse_pk (
    pcol1  INT NOT NULL,
    col2 VARCHAR(200),
    pcol3 INT NOT NULL,
    PRIMARY KEY (pcol1, pcol3)
);

using iAnywhere.MobiLink.Script;
using System;
using System.IO;
using System.Data;
using System.Text;
...

// The method used for the handle_UploadData event.
public void HandleUpload(UploadData ut) {
```

```
        // Get an UploadedTableData for the sparse_pk table.
        UploadedTableData sparse_pk_table =
    ut.GetUploadedTableByName("sparse_pk");

        // Get deletes uploaded by the MobiLink client.
        UpdateDataReader data_reader = sparse_pk_table.GetInserts();

        while (data_reader.Read()) {
            data_reader.SetNewRowValues();
            StringBuilder row_str = new StringBuilder("New values ( ");
            row_str.Append(data_reader.GetString(0)); // pcol1
            row_str.Append(", ");
            if (data_reader.IsDBNull(1)) {
                row_str.Append("<NULL>");
            }
            else {
                row_str.Append(data_reader.GetString(1)); // col2
            }
            row_str.Append(", ");
            row_str.Append(data_reader.GetString(2)); // pcol3
            row_str.Append(" )");
            data_reader.SetOldRowValues();
            row_str.Append(" Old Values (");
            row_str.Append(data_reader.GetString(0)); // pcol1
            row_str.Append(", ");
            if (data_reader.IsDBNull(1)) {
                row_str.Append("<NULL>");
            }
            else {
                row_str.Append(data_reader.GetString(1)); // col2
            }
            row_str.Append(", ");
            row_str.Append(data_reader.GetString(2)); // pcol3
            row_str.Append(" )");
            writer.WriteLine(row_str);
        }
        data_reader.Close();
    }
```

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

直接行处理

目录

直接行处理简介 610
处理直接上载 614
处理直接下载 620

直接行处理简介

注意
直接行处理是一种高级 MobiLink 功能。要使用该功能，您必须对如何创建 MobiLink 应用程序以及如何使用 MobiLink API 有透彻的了解。请参见：

- MobiLink - 入门
- MobiLink - 服务器管理第 1 页
- MobiLink - 客户端管理

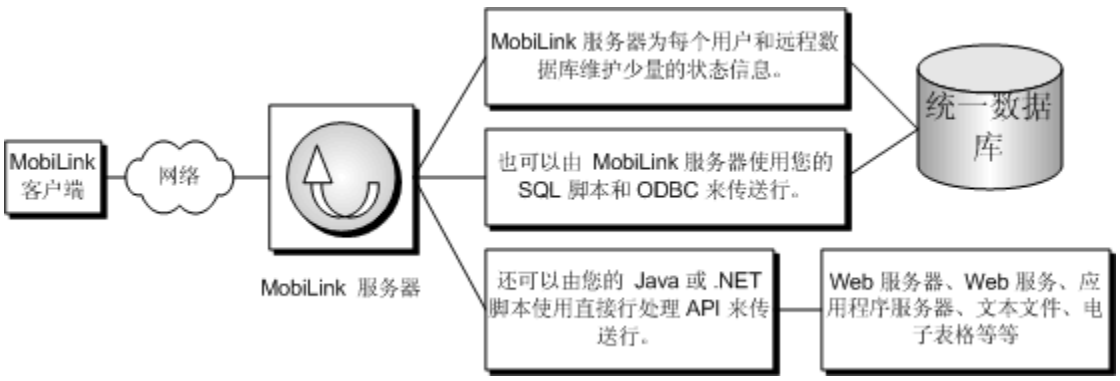
MobiLink 支持以下两种处理行的方式：SQL 行处理和直接行处理。两种方式可以单独使用，也可以一起使用。

- **SQL 行处理** 允许您将远程数据同步到受支持的统一数据库。基于 SQL 的事件为冲突解决和其它同步任务提供了一个可靠的接口。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。
- **直接行处理** 可以将远程数据与任何中央数据源同步。直接行处理允许您使用特殊 MobiLink 事件和面向 Java 和 .NET 的 MobiLink 服务器 API 来访问原始同步数据。

  实际上，任何类型的数据源都可进行同步，包括应用程序、Web 服务器、Web 服务、应用程序服务器、文本文件、电子表格、非关系数据库或不能用作统一数据库的 RDBMS。您仍需要一个统一数据库来存储 MobiLink 系统表，并且直接行处理的许多实现将同步到统一数据库和另一数据源。

  要使用直接行处理，您需要熟悉如何创建 MobiLink 统一数据库、添加同步脚本以及创建 Mobilink 远程用户。

下图显示了基本的 MobiLink 体系结构：



直接行处理的组成部分

要实现直接行处理，可以使用两个同步事件以及面向 Java 和 .NET 的 MobiLink 服务器 API 中的多个接口和方法。

Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

直接同步事件

直接行处理允许您直接访问上载流和下载流。为此，需要为 handle_UploadData 和 handle_DownloadData 同步事件编写 Java 或 .NET 方法。

- **handle_UploadData** 接受一个 UploadData 参数，该参数用来封装 MobiLink 客户端为单个上载事务而上载的操作。请参见：
  - “处理直接上载”一节第 614 页
  - “handle_UploadData 连接事件”一节第 422 页
- **handle_DownloadData** 允许您使用 DownloadData 接口设置下载操作。请参见：
  - “处理直接下载”一节第 620 页
  - “handle_DownloadData 连接事件”一节第 410 页

用于直接行处理的 MobiLink 服务器 API 的组成部分

对于 Java API：

- DBConnectionContext “getDownloadData 方法”一节第 509 页
- “DownloadData 接口”一节第 513 页
- “DownloadTableData 接口”一节第 515 页
- “UpdateResultSet”一节第 542 页
- “UploadData 接口”一节第 543 页
- “UploadedTableData 接口”一节第 545 页

对于 .NET API：

- DBConnectionContext “GetDownloadData 方法”一节第 571 页
- “DownloadData 接口”一节第 582 页
- “DownloadTableData 接口”一节第 584 页
- “UpdateDataReader 接口”一节第 603 页
- “UploadedTableData 接口”一节第 604 页
- “UploadData 接口”一节第 602 页

快速入门

要使用直接行处理，您需要熟悉如何创建 MobiLink 统一数据库、添加同步脚本以及创建 Mobilink 远程用户。

要与除统一数据库之外的数据源进行同步，请完成以下步骤。

♦ 设置直接行处理概述

1. 如果尚不具备统一数据库，请先行设置。

   无论是否要同步到统一数据库，都需要一个统一数据库来保存 MobiLink 系统表。

   请参见“MobiLink 统一数据库”第 3 页。

2. 如果要处理上载，请使用 UploadData 接口来编写一个公共方法，并将其注册给 handle_UploadData 连接事件。

   请参见“处理直接上载”一节第 614 页。

3. 如果要处理下载，请使用 DownloadData 接口来编写一个公共方法，并将其注册给 handle_DownloadData 连接事件（或另一事件）。

   请参见“处理直接下载”一节第 620 页。

4. 如果希望使用行处理 API 来按名称（而非索引）引用列，请在您的客户端中指定列名应与上载一起发送。请参见：

   - SQL Anywhere 客户端： “SendColumnNames (scn) 扩展选项”一节 《MobiLink - 客户端管理》
   - UltraLite： “Send Column Names 同步参数”一节 《UltraLite - 数据库管理和参考》

其它入门资源

- “教程：直接行处理简介” 《MobiLink - 入门》
- http://www.sybase.com/detail?id=1058600#319
- “设置 Java 同步逻辑”一节第 495 页
- “设置 .NET 同步逻辑”一节第 553 页

可在 MobiLink 新闻组上发布问题：sybase.public.sqlanywhere.mobilink。

直接行处理的开发提示

唯一主键

对于 MobiLink 同步（包括直接行处理），您的数据源必须具有不更新的唯一主键。在非关系数据源（如电子表格或文本文件）中，这意味着有一列必须包含标识行的唯一不变的值。

请参见“维护唯一主键”一节第 131 页。

列名称

在使用直接行处理时，仅当 Mobilink 客户端被配置为发送列名时，表的列名才可用。或者，您也可以使用列索引访问行信息。

要使用列名称，请参见：

- SQL Anywhere 远程： “SendColumnNames (scn) 扩展选项”一节 《MobiLink - 客户端管理》
- UltraLite 远程： “Send Column Names 同步参数”一节 《UltraLite - 数据库管理和参考》

使用下载的上次下载时间

如果可能，则像设置基于时间戳的 SQL 应用程序一样来设置您的直接行处理应用程序；维护 last_modified 列并下载基于该列的数据。此方法可以避免您使用其它下载方法时可能出现的无法预料的问题。

请参见“基于时间戳的下载”一节第 121 页。

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

上载的事务管理

您无法提交针对 Mobilink 统一数据库的事务。但是，可以提交针对直接行处理数据源的事务。在设置事务管理时，请记住以下提示：

- **在 MobiLink 提交之前提交上载** 在应用上载时，MobiLink 会在 end_upload 事件结束时提交更改。应确保您想要保留的所有上载更改已在 end_upload 脚本结束之前提交。否则，如果出现错误或故障，您可能会进入到一种状态：您的应用程序认为已应用上载，但 MobiLink 却尚未应用数据，这会导致数据丢失。
- **处理冗余上载** 如果在应用程序提交一个已上载的行之后并且在 MobiLink 服务器提交该行之前出现错误或故障，则 MobiLink 服务器和您的数据源可能会处于不一致状态。可以通过允许冗余上载并采用适当的处理逻辑来解决此问题，以确保能够正确地应用冗余上载。特别地，当您的应用程序再次发送上载时，不应再次应用该上载。

处理错误

要处理错误，请确保采用恰当的事务管理（如上所述）。此外，用于处理行的 Java 或 .NET 代码必须发送 MobiLink 服务器出现的任何异常。如果在 MobiLink 服务器或您的应用程序提交更改之前发生错误，则 MobiLink 会回退事务并保持与应用程序一致的状态。

类实例

对于直接行处理，MobiLink 为每个数据库连接创建一个类实例。该类实例在同步结束时不被销毁：它是在数据库连接关闭时被销毁。类级别变量保留先前同步的值。

处理直接上载

要处理直接上载，请完成以下步骤：

♦ **处理直接上载**

1. 为“handle_UploadData 连接事件”一节第 422 页注册一个 Java 或 .NET 方法。
2. 为 handle_UploadData 同步事件编写一个方法。此事件接受一个 UploadData 参数。请参见：
   - Java 服务器 API： “UploadData 接口”一节第 543 页
   - .NET 服务器 API： “UploadData 接口”一节第 602 页

handle_UploadData 事件通常在每个同步过程中被调用一次。不过，对于使用事务级上载的 SQL Anywhere 客户端，每个同步可能有多个上载，这时 handle_UploadData 在每个事务中被调用一次。

有关 dbmlsync 事务级上载的详细信息，请参见“-tu 选项”一节 《MobiLink - 客户端管理》。

有关编写 Java 或 .NET 同步脚本的一般信息，请参见：

- “使用 Java 语言编写同步脚本”第 493 页
- “使用 .NET 编写同步脚本”第 551 页

有关注册连接级事件的信息，请参见：

- “ml_add_java_connection_script 系统过程”一节第 630 页
- “ml_add_dnet_connection_script 系统过程”一节第 628 页

用于直接上载的类

面向 Java 和 .NET 的 MobiLink 服务器 API 提供了以下用于处理直接上载的接口：

- **UploadData** 封装单个上载事务。上载事务包含一组包含行操作的表。请参见：
  - Java API： “UploadData 接口”一节第 543 页
  - .NET API： “UploadData 接口”一节第 602 页
- **UploadedTableData** 封装由 MobiLink 客户端上载的表的插入、更新和删除操作。对于 Java，UploadedTableData 方法返回一个 UpdateResultSet 的实例。对于 .NET，UploadedTableData 方法返回一个 UpdateDataReader 接口的实例。您遍历结果集 IDataReaders 以处理上载行操作。请参见：
  - Java API： “UploadedTableData 接口”一节第 545 页
  - .NET API： “UploadedTableData 接口”一节第 604 页
- **UpdateResultSet** 对于 Java，此类代表由 UploadedTableData getUpdates 方法返回的一个更新结果集。其扩展了 java.sql.ResultSet 以包括用于检索更新行的新旧版本的特殊方法。

  请参见“UpdateResultSet”一节第 542 页。

  对于 .NET，UpdateDataReader 接口代表一组由 UploadedTableData GetUpdates 方法返回的行。其扩展了 IDataReader 以包括用于检索更新行的新旧版本的特殊方法。

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

请参见“UpdateDataReader 接口”一节第 603 页。

示例

请参见“handle_UploadData 连接事件”一节第 422 页。

直接上载的冲突处理

在 MobiLink 客户端向 MobiLink 服务器发送一个更新后的行时，发送的数据中不仅包含更新后的值（后映像或新行），而且还将包含那些上次与 MobiLink 服务器同步时获得的旧行值的副本（前映像或旧行）。如果前映像行与中央数据源中的当前值不匹配，表示检测到了冲突。

基于 SQL 的冲突解决

对于基于 SQL 的上载，MobiLink 统一数据库是您的中央数据源，并且 MobiLink 为冲突的检测和解决提供了特殊事件。

请参见“冲突处理”一节第 137 页。

使用直接行处理解决冲突

对于直接上载，可以编程方式访问新旧行以便冲突的检测和解决。

UpdateResultSet（由 UploadedTableData.getUpdates 方法返回）扩展标准 Java 或 .NET 结果集以包括处理冲突的特殊方法。setNewRowValues 设置 UpdateResultSet 以从远程客户端返回新的更新后的值（缺省模式）。setOldRowValues 设置 UpdateResultSet 以返回旧行值。

使用直接行处理检测冲突

通过使用 UpdateResultSet 方法 .setOldRowValues，在远程某行的值被更改之前获取该行的值。将返回的行值与数据源中的现有行值进行比较。如果比较的行不相等，则存在冲突。

使用直接行处理解决冲突

一旦在上载期间检测到冲突，您可以使用自定义业务逻辑来解决冲突。冲突的解决通过 Java 或 .NET 代码进行处理。

示例

假定您在 XML 文档中跟踪库存并想要将其用作中央数据源。User1 使用您的远程数据库中名为 Remote1 的数据库。User2 使用名为 Remote2 的另一远程数据库。

您的 XML 文档、User1 和 User2 开始时的库存都为 10 件。User1 卖出 3 件并将 Remote1 的库存值更新为 7 件。User2 卖出 4 件并将 Remote2 的库存更新为 6 件。当 Remote1 同步时，中央数据库将被更新为 7 件。Remote2 同步时，由于库存值不再是 10 个项目，因此会检测到冲突。若要以编程方式解决这一冲突，需要三个行值：

- 中央数据源中的当前值。
- Remote2 上载的新行值。
- Remote2 在上一次同步期间获取的旧行值。

在这种情况下，业务逻辑将使用以下公式来计算新的库存值，并解决冲突：

```
current data source - (old remote - new remote)
-> 7 - (10-6) = 3
```

以下 Java 和 .NET 过程演示了如何解决此直接上载冲突，以下表为例：

```
CREATE TABLE remoteOrders
(
    pk integer primary key not null,
    inventory integer not null
);
```

♦ 处理直接冲突 (Java)

1. 为 handle_UploadData 连接事件注册一个 Java 或 .NET 方法。

   请参见“handle_UploadData 连接事件”一节第 422 页。

   例如，以下存储过程调用在同步脚本版本 ver1 时，为 handle_UploadData 连接事件注册一个称为 HandleUpload 的 Java 方法。针对您的 MobiLink 统一数据库运行此存储过程。

   ```
   call ml_add_java_connection_script( 'ver1',
      'handle_UploadData',
      'OrderProcessor.HandleUpload' )
   ```

   有关为同步事件注册方法的详细信息，请参见：

   - “添加和删除脚本”一节第 306 页
   - “ml_add_java_connection_script 系统过程”一节第 630 页

2. 获取上载中表的 UpdateResultSet。

   OrderProcessor.HandleUpload 方法获取 remoteOrders 表的 UpdateResultSet：

   ```
   // method for handle_UploadData event
    public void HandleUpload( UploadData u_data )
    {

       // Get UploadedTableData for the remoteOrders table.
         UploadedTableData u_table =
   u_data.getUploadedTableByName("remoteOrders");

       // Get an UpdateResultSet for the remoteOrders table.
       UpdateResultSet update_rs = u_table.getUpdates();

       // (Continued...)
   ```

3. 针对每次更新，获取中央数据源中的当前值。

   在本例中，UpdateResultSet getInt 方法返回主键列（第一列）的整数值。您可以实现 getMyCentralData 方法，然后使用该方法从中央数据源获取数据。

   ```
   while( update_rs.next() )
   {
      // Get central data source values.

      // Get the primary key value.
      int pk_value = update_rs.getInt(1);

      // Get central data source values.
   ```

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

```
    int central_value = getMyCentralData(pk_value);

    // (Continued...)
```

4. 针对每次更新，获取由 MobiLink 客户端上载的旧值和新值。

   该示例分别将 UpdateResultSet setOldRowValues 和 UpdateResultSet setNewRowValues 用于旧值和新值。

```
// Set mode for old row values.
update_rs.setOldRowValues();

// Get the _old_ stored value on the remote.
int old_value = update_rs.getInt(2);

// Set mode for new row values.
update_rs.setNewRowValues();

// Get the _new_ updated value on the remote.
int new_value = update_rs.getInt(2);

// (Continued...)
```

5. 针对每次更新，检查冲突。

   旧行值与中央数据源中的当前值不匹配时，将发生冲突。要解决冲突，通过使用业务逻辑来计算解决值。如果未发生冲突，将用新的远程值对中央数据源进行更新。您可以实现 setMyCentralData 方法，然后使用该方法来执行更新。

```
// Check if there is a conflict.

if(old_value == central_value)
{
  // No conflict.
  setMyCentralData(pk_value, new_value);

}
else
{
  // Handle the conflict.
  int inventory = old_value - new_value;
  int resolved_value = central_value - inventory;

  setMyCentralData(pk_value, resolved_value);

}
}
```

♦ 处理直接冲突 (.NET)

1. 为 handle_UploadData 连接事件注册一个方法。

   例如，以下存储过程调用在同步脚本版本 ver1 时，为 handle_UploadData 连接事件注册一个称为 HandleUpload 的 .NET 方法。针对您的 MobiLink 统一数据库运行此存储过程。

```
call ml_add_dnet_connection_script( 'ver1',
   'handle_UploadData',
   'MyScripts.OrderProcessor.HandleUpload' )
```

   有关为同步事件注册方法的详细信息，请参见：

- “添加和删除脚本”一节第 306 页
- “ml_add_dnet_connection_script 系统过程”一节第 628 页

2. 获取上载中表的 UpdateDataReader。

   MyScripts.OrderProcessor.HandleUpload 方法获取 remoteOrders 表的 UpdateResultSet：

```
// method for handle_UploadData event
 public void HandleUpload( UploadData u_data )
 {

    // Get UploadedTableData for the remoteOrders table.
      UploadedTableData u_table =
u_data.GetUploadedTableByName("remoteOrders");

    // Get an UpdateDataReader for the remoteOrders table.
    UpdateDataReader update_dr = u_table.GetUpdates();

    // (Continued...)
```

3. 针对每次更新，获取中央数据源中的当前值。

   在本例中，UpdateDataReader GetInt32 方法返回主键列（第一列）的整数值。您可以实现 getMyCentralData 方法，然后使用该方法从中央数据源获取数据。

```
while( update_dr.Read() )
{
   // Get central data source values.

   // Get the primary key value.
   int pk_value = update_dr.GetInt32(0);

   // Get central data source values.
   int central_value = getMyCentralData(pk_value);

   // (Continued...)
```

4. 针对每次更新，获取由 MobiLink 客户端上载的旧值和新值。

   该示例分别将 UpdateResultSet setOldRowValues 和 UpdateResultSet setNewRowValues 用于旧值和新值。

```
   // Set mode for old row values.
   update_dr.SetOldRowValues();

   // Get an _old_ value.
   int old_value = update_dr.GetInt32(1);

   // Set mode for new row values.
   update_dr.SetNewRowValues();

   // Get the _new_ updated value.
   int new_value = update_dr.GetInt32(1);

   // (Continued...)
```

5. 针对每次更新，检查冲突。

   旧行值与中央数据源中的当前值不匹配时，将发生冲突。要解决冲突，通过使用业务逻辑来计算解决值。如果未发生冲突，将用新的远程值对中央数据源进行更新。您可以实现 setMyCentralData 方法，然后使用该方法来执行更新。

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

```
  // Check if there is a conflict.

  if(old_value == central_value)
  {
    // No conflict.
    setMyCentralData(pk_value, new_value);

  }
  else
  {
    // Handle the conflict.
    int inventory = old_value - new_value;
    int resolved_value = central_value - inventory;

    setMyCentralData(pk_value, resolved_value);

  }
}
```

处理直接下载

要处理直接下载，请完成以下步骤：

♦ **处理直接下载**

1. 为“handle_DownloadData 连接事件”一节第 410 页注册一个 Java 或 .NET 方法。
2. 为 handle_DownloadData 同步事件编写一个方法。在此事件中，您使用 DBConnectionContext 的实例来获取当前同步的 DowloadData 实例。请参见：
   - Java： “DBConnectionContext 接口”一节第 508 页
   - Java： “DownloadData 接口”一节第 513 页
   - .NET： “DBConnectionContext 接口”一节第 570 页
   - .NET： “DownloadData 接口”一节第 582 页

可以在 handle_DownloadData 同步事件中创建整个直接下载。还可以使用其它同步事件来设置直接下载操作。不过，必须创建一个 handle_DownloadData 脚本，即使其方法不执行任何操作。如果在除 handle_DownloadData 之外的事件中处理直接下载，该事件必须在 begin_synchronization 和 end_download 之间。

有关事件顺序的信息，请参见“完整的 MobiLink 事件模型”一节第 324 页。

用于直接下载的类

面向 Java 和 .NET 的 MobiLink 服务器 API 提供了以下用于创建直接下载的类：

- **DownloadData** 封装下载表，这些表包含同步期间发送给远程客户端的操作。请参见：
  - Java： “DownloadData 接口”一节第 513 页
  - .NET： “DownloadData 接口”一节第 582 页
- **DownloadTableData** 封装那些下载到 MobiLink 客户端的 upsert（更新和插入）和删除操作。

  对于 Java，DownloadTableData 方法返回 JDBC PreparedStatement 的一个实例。在 Java 中，通过以下方法将行添加到下载：设置预准备语句的列值，然后执行该预准备语句。

  对于 .NET，DownloadTableData 方法返回 .NET IDbCommand 的一个实例。在 .NET 中，通过以下方法将行添加到下载：设置命令的列值，然后执行该命令。

  请参见：

  - Java： “DownloadTableData 接口”一节第 515 页
  - .NET： “DownloadTableData 接口”一节第 584 页

示例

请参见“handle_DownloadData 连接事件”一节第 410 页。

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

MobiLink 参考

本节包含 MobiLink 参考资料。

MobiLink 服务器系统过程 623
MobiLink 实用程序 647
MobiLink 服务器系统表 653
远程数据库和统一数据库之间的 MobiLink 数据映射 695
字符集注意事项 739
用于 MobiLink 的 iAnywhere Solutions ODBC 驱动程序 743
部署 MobiLink 应用程序 749

MobiLink 服务器系统过程

目录

MobiLink 系统过程 624

MobiLink 系统过程

MobiLink 提供了以下存储过程来帮助您创建应用程序。

用于添加或删除脚本的系统过程

必须将同步脚本添加到统一数据库的系统表中，才能使用它们。以下系统过程会在统一数据库中添加同步脚本或删除其中的同步脚本：

- “ml_add_connection_script 系统过程”一节第 627 页
- “ml_add_table_script 系统过程”一节第 639 页
- “ml_add_dnet_connection_script 系统过程”一节第 628 页
- “ml_add_dnet_table_script 系统过程”一节第 629 页
- “ml_add_java_connection_script 系统过程”一节第 630 页
- “ml_add_java_table_script 系统过程”一节第 631 页

使用面向 Java 或 .NET 的 MobiLink 服务器 API 时，可以使用这些存储过程将方法注册为对应于某个事件的脚本，这样事件发生时便会运行该方法。也可以使用这些存储过程来注销方法。

使用系统过程添加脚本时，该脚本便是字符串。脚本内部的任何字符串都需要使用转义字符。对于 SQL Anywhere，每个引号 (') 都需要是双引号，这样就不会使字符串终止。

您无法使用系统过程将长度超过 255 个字节的脚本添加到 Adaptive Server Enterprise 11.5 或更早版本中。较长的脚本需要使用 Sybase Central 或直接插入进行定义。

DB2 主机 8.1 版本支持向后兼容模式，其中列名和其它的标识符最多可以使用 18 个字符。要支持此环境，所有 DB2 主机中的 MobiLink 系统对象的名称都不能超过 18 个字符。请参见“IBM DB2 主机系统过程名称转换”一节第 624 页。

版本 6 之前的 IBM DB2 LUW 仅支持不超过 18 个字符的列名和其它标识符，因此这些名称会被截断。例如，ml_add_connection_script 会被截断成 ml_add_connection_。

其它系统过程

- “ml_add_property 系统过程”一节第 637 页
- “ml_delete_sync_state_before 系统过程”一节第 644 页
- “ml_reset_sync_state 系统过程”一节第 645 页

IBM DB2 主机系统过程名称转换

IBM DB2 主机统一数据库仅支持不超过 18 个字符的列名及其它标识符。下表列出了 DB2 主机统一数据库的系统过程名与所有其它统一数据库类型的系统过程名之间的映射关系。

如果系统过程名未显示在下表中，则不需要转换。

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

系统过程名	DB2 主机统一数据库的系统过程名
“ml_add_connection_script 系统过程”一节 第 627 页	ml_add_cs
“ml_add_dnet_connection_script 系统过程”一节第 628 页	ml_add_dcs
“ml_add_dnet_table_script 系统过程”一节 第 629 页	ml_add_dts
“ml_add_java_connection_script 系统过程”一节 第 630 页	ml_add_jcs
“ml_add_java_table_script 系统过程”一节 第 631 页	ml_add_jts
“ml_add_lang_connection_script 系统过程”一节第 633 页	ml_add_lcs
“ml_add_lang_connection_script_chk 系统过程”一节第 633 页	ml_add_lcs_chk
“ml_add_lang_table_script 系统过程”一节 第 633 页	ml_add_lts
“ml_add_lang_table_script_chk 系统过程”一节 第 633 页	ml_add_lts_chk
“ml_add_passthrough 系统过程”一节 第 633 页	ml_add_pt
“ml_add_passthrough_repair 系统过程”一节 第 634 页	ml_add_pt_repair
“ml_add_passthrough_script 系统过程”一节 第 635 页	ml_add_pt_script
“ml_add_table_script 系统过程”一节 第 639 页	ml_add_ts
“ml_delete_passthrough 系统过程”一节 第 641 页	ml_del_pt
“ml_delete_passthrough_repair 系统过程”一节 第 641 页	ml_del_pt_repair

系统过程名	DB2 主机统一数据库的系统过程名
“ml_delete_passthrough_script 系统过程”一节 第 642 页	ml_del_pt_script
“ml_delete_sync_state 系统过程”一节 第 643 页	ml_del_sstate
“ml_delete_sync_state_before 系统过程”一节 第 644 页	ml_del_sstate_b4
“ml_reset_sync_state 系统过程”一节 第 645 页	ml_reset_sstate

ml_add_column 系统过程

在远程数据库中注册供命名列参数使用的列信息。

语法

ml_add_column (
'*version***',**
'*table***',**
'*column***',**
'*type***'**
)

参数

语法	说明
version	VARCHAR(128)。版本名称。
table	VARCHAR(128)。表名。
column	VARCHAR(128)。列名称。
类型	VARCHAR(128)。留作将来使用。设为空值。

注释

此过程使用远程数据库中列的相关信息填充 ml_column MobiLink 系统表。该信息由命名行参数使用。

小心
必须按照列在远程数据库表中的存在顺序来执行 ml_add_column 调用。否则可能导致数据不正确。

以下两个条件都得到满足时，才需要运行此系统过程：

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

- SQL 脚本中包含列的命名参数（例如，o.column-name 和 r.column-name）。
- 未使用 [**创建同步模式向导**]。

即便您使用 [**创建同步模式向导**]，但如果在 [模型] 模式之外修改远程模式，仍需要使用此存储过程发送未在 ml_column 中注册的列的相关信息。

要删除给定脚本版本中表名的所有条目，请将列名设置为空。

另请参见

- “ml_column”一节第 658 页
- “脚本参数”一节第 299 页

示例

以下存储过程调用会以脚本版本 Version1 的 MyTable 中的 col1 填充 ml_column MobiLink 系统表。通过此调用，可以在 Version1 脚本版本 MyTable1 的表脚本内使用命名行参数 r.col1 和 o.col1。

```
CALL ml_add_column( 'Version1', 'MyTable1', 'col1', NULL )
```

以下存储过程调用会删除脚本版本 Version1 中 MyTable1 的 ml_column MobiLink 系统表内的所有条目：

```
CALL ml_add_column( 'Version1', 'MyTable1', NULL, NULL )
```

ml_add_connection_script 系统过程

使用此系统过程在统一数据库中添加 SQL 连接脚本或删除其中的 SQL 连接脚本。

语法

ml_add_connection_script (
'*version*',
'*event*',
'*script*'
)

参数

语法	说明
version	VARCHAR(128)。版本名称。
event	VARCHAR(128)。事件名称。
script	TEXT。脚本内容。如果是 Adaptive Server Enterprise，此参数为 VARCHAR(16384)。如果是 DB2 LUW，此参数为 VARCHAR(4000)。如果是 Oracle，此参数为 CLOB。

注释

要删除连接脚本，将脚本内容参数设置为空值。

当您添加脚本时，脚本将插入到 ml_script 表中，系统会定义相应的引用，将脚本与您指定的事件和脚本版本关联起来。如果版本名是新的，则会自动插入到 ml_version 表中。

对于 DB2 主机统一数据库类型，此过程称为 ml_add_cs。请参见“IBM DB2 主机系统过程名称转换”一节第 624 页。

另请参见

- “用于添加或删除脚本的系统过程”一节第 624 页
- “添加和删除脚本”一节第 306 页
- “ml_add_table_script 系统过程”一节第 639 页
- “ml_add_dnet_connection_script 系统过程”一节第 628 页
- “ml_add_dnet_table_script 系统过程”一节第 629 页
- “ml_add_java_connection_script 系统过程”一节第 630 页
- “ml_add_java_table_script 系统过程”一节第 631 页

示例

以下语句将与 begin_synchronization 事件关联的连接脚本添加到 SQL Anywhere 统一数据库的脚本版本 custdb 中。脚本本身是一条设置 @EmployeeID 变量的语句。

```
call ml_add_connection_script( 'custdb',
 'begin_synchronization',
 'set @EmployeeID = {ml s.username}' )
```

ml_add_dnet_connection_script 系统过程

使用此系统过程将 .NET 方法注册为连接事件的脚本，或将 .NET 方法注销，不再作为连接事件的脚本。

语法

ml_add_dnet_connection_script (
'*version***',**
'*event***',**
'*script***'**
)

参数

语法	说明
version	VARCHAR(128)。版本名称。
事件	VARCHAR(128)。事件名称。
script	TEXT。脚本内容。如果是 Adaptive Server Enterprise，此参数为 VARCHAR(16384)。如果是 DB2 LUW，此参数为 VARCHAR(4000)。如果是 Oracle，此参数为 CLOB。

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

注释

要注销某个方法，请将脚本内容参数设置为空值。

脚本内容值是 .NET 程序集某个类中的公共方法（例如，MyClass.MyMethod）。

调用 ml_add_dnet_connection_script 时，该方法将与您指定的事件和脚本版本关联。如果版本名是新的，则会自动插入到 ml_version 表中。

对于 DB2 主机统一数据库类型，此过程称为 ml_add_dcs。请参见“IBM DB2 主机系统过程名称转换”一节第 624 页。

另请参见

- “用于添加或删除脚本的系统过程”一节第 624 页
- “添加和删除脚本”一节第 306 页
- “IBM DB2 主机系统过程名称转换”一节第 624 页
- “ml_add_dnet_table_script 系统过程”一节第 629 页
- “ml_add_connection_script 系统过程”一节第 627 页
- “ml_add_table_script 系统过程”一节第 639 页
- “ml_add_java_table_script 系统过程”一节第 631 页
- “方法”一节第 557 页
- “使用 .NET 编写同步脚本”第 551 页

示例

以下示例为 begin_download 事件注册 ExampleClass 类的 beginDownloadConnection 方法。

```
call ml_add_dnet_connection_script( 'ver1',
'begin_download',
'ExamplePackage.ExampleClass.beginDownloadConnection' );
```

ml_add_dnet_table_script 系统过程

使用此系统过程将 .NET 方法注册为表事件的脚本，或将 .NET 方法注销，不再作为表事件的脚本。

语法

ml_add_dnet_table_script (
'*version*',
'*table*',
'*event*',
'*script*'
)

参数

语法	说明
version	VARCHAR(128)。版本名称。
table	VARCHAR(128)。表名。

语法	说明
事件	VARCHAR(128)。事件名称。
script	TEXT。脚本内容。如果是 Adaptive Server Enterprise，此参数为 VARCHAR(16384)。如果是 DB2 LUW，此参数为 VARCHAR(4000)。如果是 Oracle，此参数为 CLOB。

注释

要注销某个方法，请将脚本内容参数设置为空值。

script 值是 .NET 程序集某个类中的公共方法（例如，MyClass.MyMethod）。

调用 ml_add_dnet_table_script 时，该方法将与您指定的表、事件和脚本版本关联。如果版本名是新的，则会自动插入到 ml_version 表中。

对于 DB2 主机统一数据库类型，此过程称为 ml_add_dts。请参见“IBM DB2 主机系统过程名称转换”一节第 624 页。

另请参见

- “用于添加或删除脚本的系统过程”一节第 624 页
- “添加和删除脚本”一节第 306 页
- “ml_add_dnet_connection_script 系统过程”一节第 628 页
- “ml_add_connection_script 系统过程”一节第 627 页
- “ml_add_table_script 系统过程”一节第 639 页
- “ml_add_java_connection_script 系统过程”一节第 630 页
- “方法”一节第 557 页
- “使用 .NET 编写同步脚本”第 551 页

示例

以下示例将 EgClass 类的 empDownloadCursor 方法指派给 emp 表的 download_cursor 事件。

```
call ml_add_dnet_table_script( 'ver1', 'emp',
'download_cursor','EgPackage.EgClass.empDownloadCursor' )
```

ml_add_java_connection_script 系统过程

使用此系统过程将 Java 方法注册为连接事件的脚本，或将 Java 方法注销，不再作为连接事件的脚本。

语法

ml_add_java_connection_script (
'*version*',
'*event*',
'*script*'
)

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

参数

语法	说明
version	VARCHAR(128)。版本名称。
事件	VARCHAR(128)。事件名称。
script	TEXT。脚本内容。如果是 Adaptive Server Enterprise，此参数为 VARCHAR(16384)。如果是 DB2 LUW，此参数为 VARCHAR(4000)。如果是 Oracle，此参数为 CLOB。

注释

要注销某个方法，请将脚本内容参数设置为空值。

script 值是 MobiLink 服务器类路径的类之中的公共方法（例如，MyClass.MyMethod）。

调用 ml_add_java_connection_script 时，该方法将与您指定的事件和脚本版本关联。如果版本名是新的，则会自动插入到 ml_version 表中。

对于 DB2 主机统一数据库类型，此过程称为 ml_add_jcs。请参见“IBM DB2 主机系统过程名称转换”一节第 624 页。

另请参见

- “用于添加或删除脚本的系统过程”一节第 624 页
- “添加和删除脚本”一节第 306 页
- “ml_add_connection_script 系统过程”一节第 627 页
- “ml_add_table_script 系统过程”一节第 639 页
- “ml_add_dnet_connection_script 系统过程”一节第 628 页
- “ml_add_dnet_table_script 系统过程”一节第 629 页
- “ml_add_java_table_script 系统过程”一节第 631 页
- “方法”一节第 499 页
- “使用 Java 语言编写同步脚本”第 493 页

示例

以下示例为 end_connection 事件注册 CustEmpScripts 类的 endConnection 方法。

```
call ml_add_java_connection_script( 'ver1',
'end_connection',
'CustEmpScripts.endConnection' )
```

ml_add_java_table_script 系统过程

使用此系统过程将 Java 方法注册为表事件的脚本，或将 Java 方法注销，不再作为表事件的脚本。

语法

ml_add_java_table_script (
'*version*',

'*table*',
'*event*',
'*script*'
)

参数

语法	说明
version	VARCHAR(128)。版本名称。
table	VARCHAR(128)。表名。
事件	VARCHAR(128)。事件名称。
script	TEXT。脚本内容。如果是 Adaptive Server Enterprise，此参数为 VARCHAR(16384)。如果是 DB2 LUW，此参数为 VARCHAR(4000)。如果是 Oracle，此参数为 CLOB。

注释

要注销某个方法，请将脚本内容参数设置为空值。

script 值是 MobiLink 服务器类路径的类中的公共方法（例如，MyClass.MyMethod）。

调用 ml_add_java_table_script 时，该方法将与您指定的表、事件和脚本版本关联。如果版本名是新的，则会自动插入到 ml_version 表中。

对于 DB2 主机统一数据库类型，此过程称为 ml_add_jts。请参见“IBM DB2 主机系统过程名称转换”一节第 624 页。

另请参见

- “用于添加或删除脚本的系统过程”一节第 624 页
- “添加和删除脚本”一节第 306 页
- “ml_add_connection_script 系统过程”一节第 627 页
- “ml_add_table_script 系统过程”一节第 639 页
- “ml_add_dnet_connection_script 系统过程”一节第 628 页
- “ml_add_dnet_table_script 系统过程”一节第 629 页
- “ml_add_java_connection_script 系统过程”一节第 630 页
- “方法”一节第 499 页
- “使用 Java 语言编写同步脚本”第 493 页

示例

以下示例为 emp 表的 download_cursor 事件注册 CustEmpScripts 类的 empDownloadCursor 方法。

```
call ml_add_java_table_script( 'ver1', 'emp',
'download_cursor','CustEmpScripts.empDownloadCursor' )
```

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

ml_add_lang_connection_script 系统过程

此过程仅供内部使用。

ml_add_lang_connection_script_chk 系统过程

此过程仅供内部使用。

ml_add_lang_table_script 系统过程

此过程仅供内部使用。

ml_add_lang_table_script_chk 系统过程

此过程仅供内部使用。

ml_add_passthrough 系统过程

使用此系统过程来标识应执行脚本的远程数据库。此过程将条目添加到 ml_passthrough 系统表。如果表中已存在具有给定 remote_id 和 run_order 的条目，此过程将更新该条目。

语法

ml_add_passthrough (
'*remote_id***',**
'*script_name***',**
run_order
)

参数

语法	说明
remote_id	VARCHAR(128)。应执行该脚本的数据库的远程 ID。此值可以是 ml_database 表中的有效远程 ID 以应用于特定客户端，也可以为空值以应用于 ml_database 表中列出的所有脚本客户端。 **小心** 将脚本应用到所有或者多个远程数据库时，请务必谨慎。编写差的脚本可能使您的多数甚至所有远程数据库受损或禁用。
script_name	VARCHAR(128)。所预订的脚本的名称。此值必须为在 ml_passthrough_script 表中定义的有效脚本名称。

语法	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。每个远程数据库会存储它试图应用的上个脚本的 run_order，而不下载或执行 run_order 小于该值的任何脚本。 该值必须为非负整数或空值。

注释

如果定义 run_order 为空，该过程将基于 remote_id 的值分配一个整数。如果 remote_id 为空，该过程将分配一个等于 ml_passthrough 中 run_order 值加上 10 的值。如果 remote_id 不为空，该过程分配 ml_passthrough 表中 remote_id 的 run_order 列的最大值加 10。

对于 DB2 主机统一数据库类型，此过程称为 ml_add_pt。请参见“IBM DB2 主机系统过程名称转换”一节第 624 页。

另请参见

- “ml_database”一节第 660 页
- “ml_passthrough”一节第 665 页
- “ml_passthrough_script”一节第 667 页

ml_add_passthrough_repair 系统过程

使用此系统过程来定义处理脚本错误的规则。每条规则定义客户端在特定脚本生成给定错误模式时应执行的操作。此过程将条目添加到 ml_passthrough_repair 系统表。如果表中已存在具有给定 failed_script_name 和 error_code 的条目，此过程将更新该条目。

语法

ml_add_passthrough_repair (
'*failed_script_name***',**
***error_code*,**
'*new_script_name***',**
'*action***'**
)

参数

语法	说明
failed_script_name	VARCHAR(128)。此规则适用的失败脚本的名称。此值必须为 ml_passthrough_script 表中的有效脚本名称。
error_code	INTEGER。此规则处理的 SQL Anywhere 错误代码。

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

语法	说明
new_script_name	VARCHAR(128)。当操作为 R 时，要替换失败脚本的脚本的名称。如果操作为 S、P 或 H，此值必须为空。如果操作是 R，此值必须为 ml_passthrough_script 表中的有效脚本名称，可以与 failed_script_name 相同。
action	CHAR(1)。生成 failed_script_name 的 error_code 时，客户端应执行的操作。该值必须是下列项之一： ● **R** （替换）表示应该用由 **new script name** 指定的脚本替换失败脚本，并尝试运行新脚本。要重新运行失败脚本，请选择 **new script name** 与 **failed script name** 相同。 ● **P** （清除）表示远程数据库应该放弃它收到的所有脚本，在此之后继续正常地执行脚本。 ● **S** （跳过）表示远程数据库应该忽略失败的脚本，并像失败脚本已经成功一样继续执行脚本。 ● **H** （暂停）表示远程数据库不应该再执行任何脚本，直到收到进一步的指示。

注释

应通过彻底测试脚本来尽最大努力避免失败的 SQL 直通脚本。

对于 DB2 主机统一数据库类型，此过程称为 ml_add_pt_repair。请参见“IBM DB2 主机系统过程名称转换”一节第 624 页。

另请参见

- “ml_passthrough_repair”一节第 666 页
- “ml_passthrough_script”一节第 667 页

ml_add_passthrough_script 系统过程

使用此系统过程来创建直通脚本。此过程将条目添加到 ml_passthrough_script 系统表。

语法

ml_add_passthrough_script (
'*script_name*',
'*flags*',
'*affected_pubs*',
'*script*',
'*description*'
)

参数

语法	说明
script_name	VARCHAR(128)。脚本名称。此值必须唯一。
flags	VARCHAR(256)。该值指示客户端如何运行脚本。该值可以为空，也可以包含以分号分隔的列表中的以下关键字的组合： ● **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。 ● **独占** 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。在此模式下，对数据库模式进行更改以匹配该脚本所描述的模式。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。 例如： `'manual;exclusive;schema_diff'`
affected_pubs	TEXT。脚本运行前必须进行同步的发布列表。空字符串或空值表示不需要同步。此值仅对 SQL Anywhere 客户端有意义。如果是 Adaptive Server Enterprise，此参数为 VARCHAR(16384)。如果是 DB2 LUW，此参数为 VARCHAR(4000)。如果是 Oracle，此参数为 CLOB。

script	TEXT。直通脚本的内容。此值不能为空。如果是 Adaptive Server Enterprise，此参数为 VARCHAR(16384)。如果是 DB2 LUW，此参数为 VARCHAR(4000)。如果是 Oracle，此参数为 CLOB。
	脚本内容必须非空。对于 UltraLite 远程数据库，**script** 内容应为以单词 **go** 分隔的 SQL 语句的集合。请注意，单词 **go** 必须显示在单独行上。对于 SQL Anywhere 远程数据库，**script** 内容可以为包括在 **begin…end** 块中时有效的 SQL 语句的任意集合。
	SQL Anywhere 远程数据库上 **script** 内容的示例：
	`DECLARE val INTEGER;`
	`SELECT c1 INTO val FROM t1 WHERE pk = 5;`
	`IF val > 100 THEN`
	`  INSERT INTO t2 VALUES ('c1 is big');`
	`ENDIF`
	UltraLite 远程数据库上 **script** 内容的示例：
	`CREATE TABLE myScript (c1 INT NOT NULL PRIMARY KEY)`
	`GO`
	`INSERT INTO myScript VALUES (1)`
	`GO`

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

语法	说明
description	VARCHAR(2000)。脚本的注释或说明。此值可以为空。

注释

如果 ml_passthrough_script 中已存在指定的 script_name，则此过程生成错误。

对于 DB2 主机统一数据库类型，此过程称为 ml_add_pt_script。请参见“IBM DB2 主机系统过程名称转换”一节第 624 页。

另请参见

- “ml_passthrough_script”一节第 667 页

ml_add_property 系统过程

使用此系统过程添加或删除 MobiLink 属性。此系统过程会更改 ml_property 系统表中的行。

语法

ml_add_property (
'*comp_name*',
'*prop_set_name*',
'*prop_name*',
'*prop_value*'
)

参数

语法	说明
comp_name	VARCHAR(128)。组件名。要按脚本版本保存属性，请将此参数设置为 ScriptVersion。对于 MobiLink 服务器属性，请将此参数设置为 MLS。对于服务器启动的同步属性，请将此参数设置为 SIS。
prop_set_name	VARCHAR(128)。属性集名称。 如果组件名是 ScriptVersion，则此参数是脚本版本的名称。 如果组件名是 MLS，则此参数可以是 ml_user_log_verbosity（用于为 MobiLink 用户指定详细程度）或 ml_remote_id_log_verbosity（用于为远程 ID 指定详细程度）。 如果组件名是 SIS，则此参数是您设置的属性所属的通告程序、网关或运营公司的名称。

语法	说明
prop_name	VARCHAR(128)。属性名称。 如果组件名是 ScriptVersion，则此参数是您定义的属性。可以使用 DBConnectionContext 来引用以下属性：getVersion 和 getProperties，或使用 ServerContext 引用以下属性：getPropertiesByVersion、getProperties 和 getPropertySetNames。 如果组件名是 MLS，则此属性可以是 MobiLink 用户名或定义的远程 ID。
prop_value	TEXT。属性值。 如果 prop_set_name 是 ml_user_log_verbosity 或 ml_remote_id_log_verbosity，则其必须是有效的 mlsrv -v 选项。 如果是 Adaptive Server Enterprise，此参数为 VARCHAR(16384)。如果是 DB2 LUW，此参数为 VARCHAR(4000)。如果是 Oracle，此参数为 CLOB。要删除属性，请设置为空。

目标 MobiLink 用户和远程 ID 的日志详细程度

可将 MobiLink 服务器设置为对目标 MobiLink 用户或远程 ID 使用不同的日志详细程度。MobiLink 服务器每五分钟检查一次 ml_property 表，为 MobiLink 用户或远程 ID 查找详细程度设置。如果存在详细程度设置，则它将使用新设置为给定 MobiLink 用户或远程 ID 记录输出消息。这样您就可以查看特定用户或远程 ID 的详细信息而无需高详细程度设置（会对服务器群产生负面影响），且无需重新启动群中的每一台服务器。

要为目标 MobiLink 用户（如 *ml_user1*）设置最高详细程度，请登录统一数据库并发出以下 SQL 命令：

```
call ml_add_property( 'MLS', 'ml_user_log_verbosity', 'ml_user1', '-v+' )
```

要为目标远程 ID（如 *rid_1*）设置最高详细程度，请登录统一数据库并发出以下 SQL 命令：

```
call ml_add_property( 'MLS', 'ml_remote_id_log_versity', 'rid_1', '-v+' )
```

请注意，*verbose_setting* 必须为有效的 MobiLink 服务器 -v 选项。例如，用于记录行数据和未定义的表脚本时，verbose_setting 可以是 -vru 或 vru。5 分钟后，MobiLink 服务器将为 ml_user1 或 rid_1 使用此详细程度设置。请参见“-v 选项”一节第 97 页。

要禁用 MobiLink 用户的日志详细程度，请登录统一数据库并发出以下 SQL 命令：

```
call ml_add_property( 'MLS', 'ml_user_log_verbosity', 'ml_user', NULL )
```

要禁用远程 ID 的日志详细程度，请登录统一数据库并发出以下 SQL 命令：

```
call ml_add_property( 'MLS', 'ml_remote_id_log_verbosity', 'rid_1', NULL )
```

五分钟后，MobiLink 服务器将为 *ml_user* 或 *rid_1* 停止使用先前的详细程度设置。

如果**同时**为给定 MobiLink 用户和远程 ID 设置了 **ml_user_log_verbosity** 和 **ml_remote_id_log_verbosity**，且如果同步过程中的 MobiLink 用户名和远程 ID 与给定的目标 MobiLink 用户和远程 ID 相同，则 MobiLink 服务器将使用 **ml_remote_id_log_verbosity** 设置记录输出消息。

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

服务器启动的同步

如果是服务器启动的同步，使用 ml_add_property 系统过程可以为通告程序、网关和运营公司设置属性。

例如，为称作 x 的 SMTP 网关添加属性 server=mailserver1：

```
ml_add_property( 'SIS','SMTP(x)','server','mailserver1' );
```

verbosity 属性适用于所有通告程序和网关，因此无法指定特定的属性集名称。要更改 verbosity 设置，请将属性集名称留空：

```
ml_add_property( 'SIS','','verbosity',2 );
```

脚本版本

如果是常规 MobiLink 同步，可以使用此系统过程将属性与脚本版本关联起来。这种情况下，请将 component_name 设置为 ScriptVersion。可以指定任何属性，并可使用 Java 和 .NET 类来访问它们。

例如，要将 LDAP 服务器与名为 MyVersion 的脚本版本进行关联：

```
ml_add_property( 'ScriptVersion','MyVersion','ldap-server','MyServer' )
```

另请参见

- “ml_property”一节第 670 页
- “用于服务器启动的同步的 MobiLink 服务器设置” 《MobiLink - 服务器启动的同步》
- “用于服务器启动的同步的 MobiLink 服务器设置” 《MobiLink - 服务器启动的同步》
- Java API DBConnectionContext：“getProperties 方法”一节第 510 页 和 “getVersion 方法”一节第 512 页
- .NET API DBConnectionContext：“GetProperties 方法”一节第 572 页 和 “GetVersion 方法”一节第 572 页
- Java API ServerContext：“getPropertiesByVersion 方法”一节第 534 页，“getProperties 方法”一节第 533 页，“getPropertySetNames 方法”一节第 534 页
- .NET ServerContext：“getPropertiesByVersion 方法”一节第 592 页，“getProperties 方法”一节第 591 页，“getPropertySetNames 方法”一节第 592 页

ml_add_table_script 系统过程

使用此系统过程在统一数据库中添加 SQL 表脚本，或删除其中的 SQL 表脚本。

语法

ml_add_table_script (
'*version*',
'*table*',
'*event*',
'*script*'
)

参数

语法	说明
version	VARCHAR(128)。版本名称。
table	VARCHAR(128)。表名。
事件	VARCHAR(128)。事件名称。
script	TEXT。脚本内容。如果是 Adaptive Server Enterprise，此参数为 VARCHAR(16384)。如果是 DB2 LUW，此参数为 VARCHAR(4000)。如果是 Oracle，此参数为 CLOB。

注释

要删除表脚本，将脚本内容参数设置为空值。

当您添加脚本时，脚本将插入到 ml_script 表中，系统会定义相应的引用，将脚本与您指定的表、事件和脚本版本关联起来。如果版本名是新的，则会自动插入到 ml_version 表中。

对于 DB2 主机统一数据库类型，此过程称为 ml_add_ts。请参见“IBM DB2 主机系统过程名称转换”一节第 624 页。

另请参见

- “用于添加或删除脚本的系统过程”一节第 624 页
- “添加和删除脚本”一节第 306 页
- “ml_add_connection_script 系统过程”一节第 627 页
- “ml_add_dnet_connection_script 系统过程”一节第 628 页
- “ml_add_dnet_table_script 系统过程”一节第 629 页
- “ml_add_java_connection_script 系统过程”一节第 630 页
- “ml_add_java_table_script 系统过程”一节第 631 页

示例

下面的命令在 Customer 表上添加与 upload_insert 事件关联的表脚本。

```
call ml_add_table_script( 'default', 'Customer', 'upload_insert',
   'INSERT INTO Customer( cust_id, name, rep_id, active )
      VALUES ( {ml r.cust_id}, {ml r.name}, {ml r.rep_id}, 1 )' )
```

ml_add_user 系统过程

此过程仅供内部使用。

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

ml_delete_passthrough 系统过程

此存储过程删除 ml_passthrough 表中的行，该行将导致指定的脚本下载到具有指定运行顺序的指定远程数据库。如果脚本在被删除前已下载到远程数据库，则脚本不从远程数据库中删除且照常执行。

语法

ml_delete_passthrough (
'*remote_id***',**
'*script_name***',**
'*run_order***'**
)

参数

语法	说明
remote_id	VARCHAR(128)。远程 ID。如果 **remote_id** 为空，则删除 ml_passthrough 表中针对指定脚本名和运行顺序的所有行。
script_name	VARCHAR(128)。脚本名称。
run_order	INTEGER。应用到远程数据库上的脚本的运行顺序。如果 **run_order** 为空，则从 ml_passthrough 表中删除针对指定 **remote_id** 和 **script_name** 的所有行，无论其运行顺序如何。

注释

MobiLink 服务器不会从 ml_passthrough 表自动删除条目。必须使用此过程来删除过期的直通脚本。

对于 DB2 主机统一数据库类型，此过程称为 ml_del_pt。请参见“IBM DB2 主机系统过程名称转换”一节第 624 页。

另请参见

- “ml_passthrough”一节第 665 页

ml_delete_passthrough_repair 系统过程

使用此系统过程来删除 ml_passthrough_repair 系统表中的修复规则。

语法

ml_delete_passthrough_repair (
'*failed_script_name***',**
error_code
)

参数

语法	说明
failed_script_name	VARCHAR(128)。规则适用的脚本的名称。
error_code	INTEGER。该规则适用的错误代码。

注释

MobiLink 服务器不会从 ml_passthrough_repair 表自动删除条目。必须使用此过程来删除过期的直通修复脚本。

对于 DB2 主机统一数据库类型，此过程称为 ml_del_pt_repair。请参见“IBM DB2 主机系统过程名称转换”一节第 624 页。

另请参见

- “ml_passthrough_repair”一节第 666 页

ml_delete_passthrough_script 系统过程

使用此系统过程来删除 ml_passthrough_script 系统表中的直通脚本。

语法

ml_delete_passthrough_script (
'*script_name***'**
)

参数

语法	说明
script_name	VARCHAR(128)。要删除的脚本的名称。

注释

如果脚本在 ml_passthrough 或 ml_passthrough_repair 系统表中被引用，则不能将其删除。

MobiLink 服务器不会从 ml_passthrough_script 表自动删除条目。必须使用此过程来删除过期的直通脚本。

对于 DB2 主机统一数据库类型，此过程称为 ml_del_pt_script。请参见“IBM DB2 主机系统过程名称转换”一节第 624 页。

另请参见

- “ml_passthrough”一节第 665 页
- “ml_passthrough_repair”一节第 666 页
- “ml_passthrough_script”一节第 667 页

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

ml_delete_sync_state 系统过程

使用此过程删除未使用或不需要的同步状态。

语法

ml_delete_sync_state (
'*user*',
'*remote_id*'
)

参数

语法	说明
user	VARCHAR(128)。MobiLink 用户名。
remote_id	VARCHAR(128)。远程 ID。

注释

这些参数可以为空。如果所有参数都为空，则该过程不执行任何操作。

此存储过程删除给定 MobiLink 用户名和远程 ID 的 ml_subscription 表中的所有行。如果 ml_subscription 表中的任何行都不再引用远程 ID，则此存储过程还会从 ml_database 表中删除此远程 ID。

如果远程 ID 为空，而 MobiLink 用户名不为空，则此存储过程会从 ml_subscription 表中删除给定 MobiLink 用户名引用的所有行；如果 ml_subscription 表中的任何行都不再引用这些远程 ID，则此存储过程会从 ml_database 表中删除这些远程 ID。

如果 MobiLink 用户名为空而远程 ID 不为空，此存储过程将删除 ml_subscription 表以及 ml_database 表中针对给定远程 ID 的所有行。

此存储过程不会删除 MobiLink 用户，即使已删除 ml_database 表中所有远程 ID 且 ml_subscription 表中无任何行引用此用户。如果需要删除此 MobiLink 用户，可以通过发出如下命令来将其删除

```
delete from ml_user where name = 'user_name'
```

其中 *user_name* 是您要删除的 MobiLink 用户。

使用此存储过程时要极其谨慎，因为 MobiLink 服务器会在 ml_database 和 ml_subscription 表中自动添加此远程 ID，而下次 MobiLink 客户端为此远程 ID 请求同步时 MobiLink 服务器并不会检查它的同步状态。如果远程 ID 的上次同步尝试不成功，则删除它的同步状态可能会导致数据不一致。

对于 DB2 主机统一数据库类型，此过程称为 ml_del_sstate。请参见“IBM DB2 主机系统过程名称转换”一节第 624 页。

另请参见

- “ml_subscription”一节第 689 页
- “ml_database”一节第 660 页

Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

示例

以下示例为 MobiLink 用户 John 清理远程 ID 为 remote_db_for_John 的远程数据库的相关 MobiLink 系统表信息：

```
CALL ml_delete_sync_state( 'John', 'remote_db_for_John' )
```

ml_delete_sync_state_before 系统过程

使用此过程在删除远程数据库后清理 MobiLink 系统表。

语法

ml_delete_sync_state_before (
'*ts*'
)

参数

语法	说明
ts	TIMESTAMP。日期时间的显示顺序必须与统一数据库中指定的顺序完全一致。如果统一数据库中的日期时间格式设置为 'yyyy/mm/dd hh:mm:ss.ssss'，则时间戳必须以年、月、日、小时、分钟、秒、分秒的顺序显示。

注释

此存储过程从 MobiLink 系统表中删除与不再使用的远程数据库有关的行。具体地讲，它执行以下操作：

- 从 ml_subscription 系统表中删除所有 last_upload_time 和 last_download_time 都早于给定时间戳的行。
- 如果 ml_subscription 表中的任何行都不再引用远程 ID，则从 ml_database 系统表中删除这些远程 ID。

不应将此系统过程用于过近的时间段，因为它删除的行所属的远程数据库可能实际上尚未被删除。否则，删除 ml_subscription 和 ml_database 中的行可能会导致因上载不成功而处于 "未知状态" 的远程数据库出现问题；处于该未知状态时，远程数据库依赖 MobiLink 系统表重新发送数据。

为此过程提供的时间戳必须具有正确的日期时间格式，因为该过程不会对参数的日期时间格式进行校验。

对于 DB2 主机统一数据库类型，此过程称为 ml_del_sstate_b4。请参见“IBM DB2 主机系统过程名称转换”一节第 624 页。

另请参见

- “ml_subscription”一节第 689 页
- “ml_database”一节第 660 页

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

示例

以下示例清理自 2004 年 1 月 10 日以来未曾同步的远程数据库的相关 MobiLink 系统表信息。它对日期时间格式为 yyyy/mm/dd hh:mm:ss.ssss 的 SQL Anywhere 统一数据库有效。

```
CALL ml_delete_sync_state_before( '2004/01/10 00:00:00' )
```

ml_delete_user 系统过程

此过程仅供内部使用。

ml_reset_sync_state 系统过程

使用此过程在 MobiLink 系统表中重置同步状态信息。

语法

ml_reset_sync_state (
'*user***',**
'*remote_id***'**
)

参数

语法	说明
user	VARCHAR(128)。MobiLink 用户名。
remote_id	VARCHAR(128)。远程 ID。

注释

参数可以为空。如果两个参数都为空，则此过程不执行任何操作。

此存储过程为给定用户名和远程 ID 将 ml_subscription 表中的 progress、last_upload_time 和 last_download_time 列设置为它们的缺省值。progress 列的缺省值是 0，last_upload_time 和 last_download_time 列的缺省值是 '1900/01/01 00:00:00'。

如果远程 ID 为空，而 MobiLink 用户名不为空，则此过程为由给定 MobiLink 用户名引用的 ml_subscription 表中的行将这些列设置为缺省值。如果 MobiLink 用户名为空，而远程 ID 不为空，则此过程为 ml_subscription 表中具有给定远程 ID 的行将这些列设置为缺省值。

使用此存储过程时要极其谨慎。MobiLink 客户端下次为此远程 ID 请求同步时，MobiLink 服务器不会为该远程 ID 执行任何同步状态检查。重置上次同步不成功的远程 ID 会导致数据不一致。

对于 DB2 主机统一数据库类型，此过程称为 ml_reset_sstate。请参见“IBM DB2 主机系统过程名称转换”一节第 624 页。

Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

ml_server_delete 系统过程

此过程仅供内部使用。

ml_server_update 系统过程

此过程仅供内部使用。

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

MobiLink 实用程序

目录

MobiLink 实用程序简介 648
MobiLink 停止实用程序 (mlstop) 649
MobiLink 用户验证实用程序 (mluser) 650

MobiLink 实用程序简介

有两种 MobiLink 服务器实用程序：

- “MobiLink 停止实用程序 (mlstop)”一节第 649 页
- “MobiLink 用户验证实用程序 (mluser)”一节第 650 页

此外，请参见：

- MobiLink 客户端实用程序： “MobiLink 客户端实用程序” 《MobiLink - 客户端管理》
- UltraLite 实用程序： “UltraLite 实用程序” 《UltraLite - 数据库管理和参考》
- 适用于使用 TLS 证书的实用程序： “证书实用程序”一节 《SQL Anywhere 服务器 - 数据库管理》
- 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》

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

MobiLink 停止实用程序 (mlstop)

停止本地计算机上的 MobiLink 服务器。

语法

mlstop [*options*] [*name*]

选项	说明
@data	此选项用于从指定的环境变量或配置文件中读入选项。如果存在具有相同名称的环境变量和配置文件，则使用环境变量。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》。
-f	强制关机。在硬关机不起作用时使用。
-h	硬关机。MobiLink 停止所有同步，然后退出。某些远程数据库可能会报告一个错误。
-q	安静模式。此选项会取消标题。
-t *time*	软关机，在指定的时间后硬关机。*time* 是一个后面带有 D、H、M 或 S（分别表示日、时、分、秒）的数字。例如，`-t 10m` 指定服务器在 10 分钟后或当前同步完成后（按两者中较早的时间执行）关机。D、H、M 和 S 不区分大小写。
-w	等待 MobiLink 服务器关闭，然后再从命令返回。
name	如果 MobiLink 服务器是使用 -zs 选项启动的，则关闭时必须指定同一服务器名。请参见“-zs 选项”一节第 112 页。

说明

在缺省情况下（如果 -f、-h 或 -t 均未指定），mlstop 执行软关机。

- **软关机** 意味着 MobiLink 服务器在完成当前的同步后停止接受新连接并退出。
- **硬关机** 意味着 MobiLink 服务器停止所有同步并退出。某些远程数据库可能会报告一个错误。

MobiLink 用户验证实用程序 (mluser)

在统一数据库中注册 MobiLink 用户。对于 SQL Anywhere 远程数据库，先前必须已经使用 CREATE SYNCHRONIZATION USER 语句在远程数据库上创建了用户。

语法

mluser [*options*] **-c** "*connection-string*"
{ **-f** *file* | **-u** *user* [**-p** *password*] }

选项	说明
@data	此选项用于从指定的环境变量或配置文件中读入选项。如果存在具有相同名称的环境变量和配置文件，则使用环境变量。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》。
-c "*keyword=value*;..."	此选项用于提供数据库连接参数。连接字符串必须授予实用程序使用 ODBC 数据源连接到统一数据库的权限。该参数是必需的。
-d	删除由 -f 或 -u 指定的用户名。
-f *filename*	从指定的文件读取用户名和口令。该文件应为一个文本文件，每行包含一个用户名和口令对，用户名和口令使用空格分隔。必须指定 -f 或 -u。
-fips	设置此选项时，如果未安装 FIPS 支持，mluser 将失败。
-o *filename*	将输出消息记录到指定文件。
-ot *filename*	截断日志文件，然后将输出消息附加到该文件中。缺省情况下这些消息将输出到屏幕。
-p *password*	要与用户关联的口令。此选项只能与 -u 一起使用。
-pc *collation-id*	为用户名和口令的字符集转换提供数据库归类 ID。它应该是 SQL Anywhere 归类标签之一，如在“支持的归类和替代归类”一节 《SQL Anywhere 服务器 - 数据库管理》中所列。如果是从使用其它字符集（而非由地区确定的缺省字符集）编码的文件中读取用户名和口令，必须使用此选项。

Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

选项	说明
-u *ml_username*	指定要添加（如果与 -d 一起使用则为删除）的用户名。在单个命令行只能指定一个用户。如果在使用口令，则此选项与 -p 结合使用。必须指定 -f 或 -u。
-v	指定详细记录。

注释

获得 "用户/口令" 对之后，mluser 实用程序首先尝试添加用户。如果用户已经添加到统一数据库中，它会尝试为该用户更新口令。

还可以使用以下变通方法在统一数据库中注册用户名：

- 使用 Sybase Central。
- 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。

远程数据库中必须已经存在 MobiLink 用户。要在远程添加用户，可以使用以下方法：

- 对于 SQL Anywhere 远程数据库，使用 CREATE SYNCHRONIZATION USER 设置名称并与该用户名同步。
- 对于 UltraLite 远程，可以使用 ul_synch_info 结构的 user_name 字段；或者在 Java 中，在进行同步前使用 ULSynchInfo 类的 SetUserName() 方法。

另请参见

- “MobiLink 用户” 《MobiLink - 客户端管理》
- “-zu 选项”一节第 114 页
- “CREATE SYNCHRONIZATION USER 语句 [MobiLink]”一节 《SQL Anywhere 服务器 - SQL 参考》
- “传送层安全” 《SQL Anywhere 服务器 - 数据库管理》

Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

MobiLink 服务器系统表

目录

MobiLink 系统表简介 655
IBM DB2 主机系统表名称转换 656
ml_active_remote_id 657
ml_column 658
ml_connection_script 659
ml_database 660
ml_device 661
ml_device_address 662
ml_listening 663
ml_passthrough 665
ml_passthrough_repair 666
ml_passthrough_script 667
ml_passthrough_status 669
ml_property 670
ml_qa_clients 671
ml_qa_delivery 672
ml_qa_delivery_archive 673
ml_qa_global_props 674
ml_qa_notifications 675
ml_qa_repository 676
ml_qa_repository_archive 677
ml_qa_repository_props 678
ml_qa_repository_props_archive 679
ml_qa_repository_staging 680
ml_qa_status_history 681
ml_qa_status_history_archive 682
ml_qa_status_staging 683
ml_script 684
ml_script_version 685
ml_scripts_modified 686
ml_server 687

ml_sis_sync_state 688
ml_subscription 689
ml_table 691
ml_table_script 692
ml_user 693

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

MobiLink 系统表简介

MobiLink 系统表存储有关 MobiLink 用户、预订、表、脚本、脚本版本和其它信息的信息。它们是进行 MobiLink 同步所必需的。与其它系统表不同，您可以修改 MobiLink 系统表，尽管在大多数情况下并不需要这样做。

MobiLink 系统表是在您为统一数据库运行 MobiLink 安装脚本时创建的。它们必须存储在统一数据库上。运行安装脚本的数据库用户是该脚本创建的 MobiLink 系统表的所有者。

请参见“建立统一数据库”一节第 6 页。

注意

- 本章介绍 SQL Anywhere 统一数据库中 MobiLink 系统表的数据类型。某些 RDBMS 中的数据类型稍有不同。
- DB2 主机 8.1 版本支持向后兼容模式，其中列名和其它的标识符最多可以使用 18 个字符。要支持此环境，所有 DB2 主机中的 MobiLink 系统对象的名称都不能超过 18 个字符。请参见“IBM DB2 主机系统表名称转换”一节第 656 页。
- IBM DB2 LUW 5.2 版仅支持不超过 18 个字符的列名及其它标识符。在 DB2 LUW 5.2 统一数据库中，必要时会截断 MobiLink 系统表名。

Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

IBM DB2 主机系统表名称转换

IBM DB2 主机统一数据库仅支持不超过 18 个字符的列名及其它标识符。下表列出了 DB2 主机统一数据库的系统表名与所有其它统一数据库类型的系统表名之间的映射关系。

如果系统表名未出现在下表中，则不需要转换。

系统表名	DB2 主机统一数据库的系统表名
“ml_active_remote_id”一节第 657 页	ml_active_rid
“ml_connection_script”一节第 659 页	ml_conn_script
“ml_passthrough”一节第 665 页	ml_pt
“ml_passthrough_repair”一节第 666 页	ml_pt_repair
“ml_passthrough_script”一节第 667 页	ml_pt_script
“ml_passthrough_status”一节第 669 页	ml_pt_status
“ml_scripts_modified”一节第 686 页	ml_script_modified

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

ml_active_remote_id

将每个远程数据库的同步状态存储在服务器群中。

列	说明
remote_id	VARCHAR(128)。标识当前正在同步的远程数据库的 ID 的唯一整数。
server_id	INTEGER。在 ml_server 表中引用 server_id 的值。
status	CHAR(1)。远程数据库的同步状态。值 O 表示正在进行的同步，C 表示取消的同步。

注释

除非有服务器群在运行，否则该表不包含数据。

对于 DB2 主机统一数据库类型，该表称为 ml_active_rid。请参见“IBM DB2 主机系统表名称转换”一节第 656 页。

约束

PRIMARY KEY(remote_id)

FOREIGN KEY(server_id) REFERENCES ml_server(server_id)

ml_column

以特定脚本版本存储特定表的列名。

列	说明
version_id	INTEGER。标识脚本版本的数字。
table_id	INTEGER。标识表的数字。
idx	INTEGER。表中此列的索引 origin 1。列顺序必须为在远程数据库中创建列时的顺序。
name	VARCHAR(128)。列名称。
类型	VARCHAR(128)。当前未使用。

仅在 SQL 脚本中包含列的命名参数（例如，o.column-name 和 r.column-name）时，才需要此表。（列索引例外，即使此 MobiLink 系统表尚未填充，该索引仍然可用；例如，o.column-index 和 r.column-index 就属例外。）

该表在部署 MobiLink 模型时由 [**创建同步模型向导**] 填充。如果未使用 [**创建同步模型向导**]，或如果确实使用了，但后来在 Sybase Central 的 [模型] 模式以外更改了远程数据库中已同步列的模式，则可以使用 ml_add_column 存储过程来填充该表。

注意：dbmlsync 扩展选项 SendColumnNames 和 UltraLite 同步参数 Send Column Names 由直接行处理使用，但不用于命名的行参数。

注释

有一个系统视图 ml_columns，更便于查看表的内容。

约束

PRIMARY KEY(idx, version_id, table_id)

UNIQUE(version_id, table_id, name)

FOREIGN KEY(version_id) REFERENCES ml_script_version(version_id)

FOREIGN KEY(table_id) REFERENCES ml_table(table_id)

另请参见

- “ml_add_column 系统过程”一节第 626 页
- “脚本参数”一节第 299 页

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

ml_connection_script

对于给定脚本版本，此表将脚本与给定事件关联起来。

列	说明
version_id	INTEGER。标识脚本版本的数字。
event	VARCHAR(128)。触发连接脚本的事件的名称。
script_id	INTEGER。标识脚本的数字。连接脚本的文本存储在 ml_script 系统表中。

注释

有一个系统视图 ml_connection_scripts，更便于查看表的内容。

对于 DB2 主机统一数据库类型，该表称为 ml_conn_script。请参见“IBM DB2 主机系统表名称转换”一节第 656 页。

约束

PRIMARY KEY(version_id, event)

FOREIGN KEY(version_id) REFERENCES ml_script_version(version_id)

FOREIGN KEY(script_id) REFERENCES ml_script(script_id)

ml_database

存储每个已同步远程数据库的唯一 ID。

小心
不要改动此表。

列	说明
rid	INTEGER。标识远程 ID 的唯一整数。此值在内部使用。
remote_id	VARCHAR(128)。远程 ID 可唯一地标识每个远程数据库。
script_ldt	TIMESTAMP。上次下载直通脚本的时间。
description	VARCHAR(128)。保留。

注释

远程 ID 在每次同步时由客户端发送。MobiLink 服务器使用此远程 ID 跟踪每个远程数据库的状态信息。

约束

PRIMARY KEY(rid)

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

ml_device

此表仅用于服务器启动的同步。它存储设备跟踪所需的设备名。

列	说明
device_name	VARCHAR(255)。分配给设备的名称.除非您使用 dblsn -e 选项指定名称，否则将从操作系统抽取此名称。
listener_version	VARCHAR(128)。非空。设备上已安装软件的 SQL Anywhere 版本号。更改此值不会影响软件的运行，但可能对诊断有用处。
listener_protocol	INTEGER。非空。此列为 0、1 或 2： ● **0** 适用于 9.0.1 之前版本的 SQL Anywhere 的监听器 ● **1** 适用于 9.0.0 之后版本的 Palm 监听器 ● **2** 适用于 9.0.0 之后版本的 Windows 监听器
info	VARCHAR(255)。非空。有关监听设备的操作系统信息。可以使用 dblsn -f 选项提供的信息替换此信息。
ignore_tracking	VARCHAR(1)。非空。如果是 **y**，则不会将跟踪信息写入行。如果是 **n**，则会将跟踪信息写入行。
source	VARCHAR(255)。非空。如果该行是由自动设备跟踪创建的，则为 **tracking**。否则为空，除非您更改它，使用存储过程添加有关此行中数据来源的信息。除非它设置为 **tracking**，否则此列中的值不影响软件的操作。

注释

MobiLink 系统表 ml_device、ml_device_address 和 ml_listening 包含有关用于服务器启动同步的设备的信息。DeviceTracker 网关使用此信息来按 MobiLink 用户名寻址目标设备。

在大多数情况下，您应该不需要改动这些表。不过，如果您的设备不支持设备跟踪或为了进行故障排除而想要忽略设备跟踪，则可以使用预定义存储过程在此系统表中添加或删除行。请参见“添加设备跟踪支持”一节 《MobiLink - 服务器启动的同步》 。

如果要停止自动跟踪，请将 ignore_tracking 设置为 **y**。在这种情况下，还建议您使用 **tracking** 以外的源名。

约束

PRIMARY KEY(device_name)

另请参见

- “ml_set_device 系统过程”一节 《MobiLink - 服务器启动的同步》
- “ml_delete_device 系统过程”一节 《MobiLink - 服务器启动的同步》

ml_device_address

此表仅用于服务器启动的同步。它存储设备跟踪所需的寻址信息。

列	说明
device_name	VARCHAR(255)。非空。设备的名称。除非您使用 dblsn -e 选项指定名称，否则将从操作系统抽取此名称。
medium	VARCHAR(255)。非空。对于 UDP，为 **_UDP_**。否则为网络提供商 ID。
address	VARCHAR(255)。非空。对于 UDP，为 *ip*:*port-number*，其中 *ip* 为 IP 地址或主机名。对于 SMS，此为电话号码。
active	VARCHAR(1)。非空。活动时为 **y**，否则为 **n**。如果 UDP 通道不响应并且存在回退 SMS 传送路径，DeviceTracker 网关可能会停用 UDP 通道。
last_modified	Timestamp。非空。缺省时间戳。上次修改此行的日期时间。
ignore_tracking	VARCHAR(1)。非空。如果是 **y**，则不会将跟踪信息写入行。如果是 **n**，则会将跟踪信息写入行。
source	VARCHAR(255)。非空。如果该行是由自动设备跟踪创建的，则为 **tracking**。否则为空，除非您更改它，使用存储过程添加有关此行中数据来源的信息。除非它设置为 **tracking**，否则此列中的值不影响软件的操作。

注释

MobiLink 系统表 ml_device、ml_device_address 和 ml_listening 包含有关用于服务器启动同步的设备的信息。DeviceTracker 网关使用此信息来按 MobiLink 用户名寻址目标设备。

在大多数情况下，您应该不需要改动这些表。不过，如果您的设备不支持设备跟踪或为了进行故障排除而想要忽略设备跟踪，则可以使用预定义存储过程在此系统表中添加或删除行。请参见“添加设备跟踪支持”一节 《MobiLink - 服务器启动的同步》。

如果要停止自动跟踪，请将 ignore_tracking 设置为 **y**。在这种情况下，还建议您使用 **tracking** 以外的源名。

约束

PRIMARY KEY(device_name, medium)

FOREIGN KEY(device_name) REFERENCES ml_device(device_name)

另请参见

- “ml_set_device_address 系统过程”一节 《MobiLink - 服务器启动的同步》
- “ml_delete_device_address 系统过程”一节 《MobiLink - 服务器启动的同步》

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

ml_listening

此表仅用于服务器启动的同步。它将 MobiLink 用户名映射到设备名，以便进行设备跟踪。

列	说明
name	VARCHAR(128)。非空。用于对通知进行寻址的名称。可以通过以下三种方式之一填充此列： ● 如果使用 dblsn -t+ 选项，这是您为 ml_user 定义的别名。 ● 如果使用 dblsn -u 选项，则此项为 ml_user 名。 ● 如果既不使用 -t+，也不使用 -u，则缺省名为 *device-name*-**dblsn**，其中 *device-name* 是您设备的名称。可在监听器消息窗口中找到设备名。也可以使用 dblsn -e 选项设置设备名。 请参见“用于 Windows 的监听器选项”一节 《MobiLink - 服务器启动的同步》。
device_name	VARCHAR(255)。非空。分配给设备的名称.除非您使用 dblsn -e 选项指定名称，否则将从操作系统抽取此名称。
listening	VARCHAR(1)。非空。对于活动的监听器，为 **y**；否则为 **n**。使用 dblsn 选项 -t 时会自动设置此字段，也可使用存储过程手工设置该字段。
ignore_tracking	VARCHAR(1)。非空。如果是 **y**，则不会将跟踪信息写入行。如果是 **n**，则会将跟踪信息写入行。
source	VARCHAR(255)。非空。如果该行是由自动设备跟踪创建的，则为 **tracking**。否则为空，除非您更改它，使用存储过程添加有关此行中数据来源的信息。此列中的值不会影响软件的操作。

注释

MobiLink 系统表 ml_device、ml_device_address 和 ml_listening 包含有关用于服务器启动同步的设备的跟踪信息。DeviceTracker 网关使用此信息来按 MobiLink 用户名寻址目标设备。

在大多数情况下，您应该不需要改动这些表。但是，如果您的设备不支持设备跟踪或者为了排除故障而需要停止设备跟踪，则可以使用预定义的存储过程在此系统表中添加或删除行。请参见“添加设备跟踪支持”一节 《MobiLink - 服务器启动的同步》。

如果要停止自动跟踪，请将 ignore_tracking 设置为 **y**。在这种情况下，还建议您使用 **tracking** 以外的源名。

约束

PRIMARY KEY (name)

FOREIGN KEY(device_name) REFERENCES ml_device(device_name)

另请参见

- “ml_set_listening 系统过程”一节 《MobiLink - 服务器启动的同步》
- “ml_delete_listening 系统过程”一节 《MobiLink - 服务器启动的同步》

Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

ml_passthrough

存储指示每个远程数据库上应运行的脚本的行。

列	说明
remote_id	VARCHAR(128)。应执行脚本的客户端的远程 ID。
run_order	INTEGER。针对此客户端的脚本运行顺序。值必须是非负数。
script_id	INTEGER。在 ml_passthrough_script 表引用中 script_id 的整数。该值指定要执行的脚本。
last_modified	TIMESTAMP。上次添加或修改直通信息的时间。

注释

可以使用系统过程 ml_add_passthrough 和 ml_delete_passthrough 添加、修改及删除表中的条目。

对于 DB2 主机统一数据库类型，该表称为 ml_pt。请参见“IBM DB2 主机系统表名称转换”一节第 656 页。

约束

PRIMARY KEY(remote_id, run_order)

FOREIGN KEY(remote_id) REFERENCES ml_database(remote_id)

FOREIGN KEY(script_id) REFERENCES ml_passthrough_script(script_id)

另请参见

- “ml_add_passthrough 系统过程”一节第 633 页
- “ml_delete_passthrough 系统过程”一节第 641 页

ml_passthrough_repair

存储定义客户端应如何处理脚本错误的规则。

列	说明
failed_script_id	INTEGER。标识应用此规则的脚本。在 ml_passthrough_script 表中引用 script_id 的值。
error_code	INTEGER。此规则处理的错误代码。
new_script_id	INTEGER。ml_passthrough_script 表中定义的 script_id，表示修复脚本。如果操作是 R，该值指定用于替换错误脚本的脚本。该值引用 ml_passthrough_script 表的 script_id 列。如果操作是 S、P 或 H，则该值为空。
action	CHAR(1)。当脚本失败时在客户端执行的操作。该值必须是下列项之一： ● **R** （替换）表示应该用由 **new script name** 指定的脚本替换失败脚本，并尝试运行新脚本。要重新运行失败脚本，请选择 **new script name** 与 **failed script name** 相同。 ● **P** （清除）表示远程数据库应该放弃它收到的所有脚本，在此之后继续正常地执行脚本。 ● **S** （跳过）表示远程数据库应该忽略失败的脚本，并像失败脚本已经成功一样继续执行脚本。 ● **H** （暂停）表示远程数据库不应该再执行任何脚本，直到收到进一步的指示。

注释

您可以使用系统过程 ml_add_passthrough_repair 和 ml_delete_passthrough_repair 添加、修改及删除表中的条目。

对于 DB2 主机统一数据库类型，该表称为 ml_pt_repair。请参见“IBM DB2 主机系统表名称转换”一节第 656 页。

约束

PRIMARY KEY(failed_script_id, error_code)

FOREIGN KEY(failed_script_id) REFERENCES ml_passthrough_script(script_id)

另请参见

- “ml_add_passthrough_repair 系统过程”一节第 634 页
- “ml_delete_passthrough_repair 系统过程”一节第 641 页

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

ml_passthrough_script

为每个直通脚本存储名称和内容。

列	说明
script_id	INTEGER。唯一标识直通脚本名称的整数。该值在内部使用。
script_name	VARCHAR(128)。唯一的直通脚本名。
flags	VARCHAR(256)。该值告知客户端如何运行脚本。该值可以为空，也可以包含以分号分隔的列表中的以下关键字的组合： ● **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。 ● **独占** 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。在此模式下，对数据库模式进行更改以匹配该脚本所描述的模式。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。 例如： `'manual;exclusive;schema_diff'`
affected_pubs	TEXT。脚本运行前必须进行同步的发布列表。空或空值的发布列表表示不需要同步。该值仅对 SQL Anywhere 客户端有意义。
script	TEXT。直通脚本的内容。
description	VARCHAR(2000)。脚本的注释或说明。

注释

使用以下过程添加和删除此表中的条目。

- “ml_add_passthrough_script 系统过程”一节第 635 页
- “ml_delete_passthrough_script 系统过程”一节第 642 页

建议您不要不使用 ml_add_passthrough_script 和 ml_delete_passthrough_script 系统过程而直接更新 ml_passthrough_script 表。如果客户端已经下载了原始直通脚本，他们不会收到新的更新脚本。多客户端上的不同脚本之间的差异使得管理脚本执行变得困难或不可能。

对于 DB2 主机统一数据库类型，该表称为 ml_pt_script。请参见“IBM DB2 主机系统表名称转换”一节第 656 页。

Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

约束

PRIMARY KEY(script_id)

另请参见

- “ml_add_passthrough_script 系统过程”一节第 635 页
- “ml_delete_passthrough_script 系统过程”一节第 642 页

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

ml_passthrough_status

存储每个直通脚本在由客户端执行后的状态。

列	说明
status_id	INTEGER。标识行的唯一整数。
remote_id	VARCHAR(128)。标识执行脚本的远程数据库。在 ml_database 表中引用 remote_id。
run_order	INTEGER。客户端上脚本的运行顺序。
script_id	INTEGER。标识所执行的脚本。在 ml_passthrough_script 表中引用 script_id。
script_status	CHAR(1)。脚本的状态。**S** 代表成功，**E** 代表错误。
error_code	INTEGER。在远程数据库上由脚本生成的 SQL 代码。
error_text	TEXT。在远程数据库上由脚本生成的错误文本。
remote_run_time	TIMESTAMP。在远程数据库执行脚本的时间。

注释

MobiLink 服务器不会从 ml_passthrough_status 表自动删除条目。要从该表删除条目，必须手动执行。

对于 DB2 主机统一数据库类型，该表称为 ml_pt_status。请参见“IBM DB2 主机系统表名称转换”一节第 656 页。

约束

PRIMARY KEY(status_id)

FOREIGN KEY(remote_id) REFERENCES ml_database(remote_id)

FOREIGN KEY(script_id) REFERENCES ml_passthrough_script(script_id)

另请参见

- “upload_insert 表事件”一节第 471 页

ml_property

此表存储某些 MobiLink 属性。

列	说明
component_name	VARCHAR(128)。对于用户定义的属性，此项可以是 **ScriptVersion** 或 **SIS**。
property_set_name	VARCHAR(128)。如果 component_name 是 **ScriptVersion**，则此项是脚本版本的名称。如果 component_name 是 **SIS**，则此项是您设置的属性所属的通告程序、网关或运营公司的名称。
property_name	VARCHAR(128)。属性的名称。如果 component_name 是 **ScriptVersion**，则此项是用户定义的属性。如果 component_name 是 **SIS**，则此为通告程序、网关或运营公司的属性。请参见“用于服务器启动的同步的 MobiLink 服务器设置”《MobiLink - 服务器启动的同步》。
property_value	TEXT。属性值。

注释

此表存储 "名称-值" 对。此表中的一些属性由 MobiLink 在内部使用。此外，您可以使用存储过程 ml_add_property 在此表中添加行或删除其中的行。

还可以使用 component_name **ScriptVersion** 按脚本版本存储可通过 Java 或 .NET 脚本编写逻辑访问的信息。

约束

PRIMARY KEY(component_name, property_set_name, property_name)

另请参见

- “ml_add_property 系统过程”一节第 637 页

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

ml_qa_clients

此表只用于 QAnywhere 应用程序。它是一个仅存在于 SQL Anywhere 和 Oracle 统一数据库上的全局临时表。

小心
不要改动此表。

列	说明
client	VARCHAR(128)。作为所上载消息目标的客户端。

ml_qa_delivery

此表只用于 QAnywhere 应用程序。

小心
不要改动此表。

列	说明
msgid	VARCHAR(128)。全局唯一消息标识符。
seqno	BIGINT。用于对消息进行排序，这是进行真正排队所必需的。
address	VARCHAR(255)。目标接收者的地址。
clientaddress	VARCHAR(128)。地址的客户端部分。
client	VARCHAR(128)。作为当前客户端状态目标的客户端。
originator	VARCHAR(128)。发起客户端的名称。
priority	INTEGER。一个 0 到 9 之间的数字。优先级数字较高的消息将先于优先级数字较低的消息进行传送。缺省值为 4。
expires	TIMESTAMP。到期时间，在此时间之后将不传递消息。
kind	INTEGER。表示该消息是二进制消息 (**1**) 还是文本消息 (**2**)。
contentsize	BIGINT。消息的大小。如果是二进制消息，此为字节数。如果是文本消息，此为字符数。
status	INTEGER。消息的状态。可以是 1（待执行）、10（正在接收）、30（已到期）、40（已取消）、50（无法收到）或 60（已接收）。
statustime	TIMESTAMP。到达此状态的时间。该时间是到达此状态的客户端的本地时间。
syncstatus	INTEGER。就此消息而言客户端与服务器之间同步的状态。可以是 0（不同步）、1（同步）、2（不应同步消息）或 3（正在同步）。
receiverid	VARCHAR(128)。由接收方设置的标识消息接收方的标识符（如果有）。

约束

PRIMARY KEY(msgid, address)

FOREIGN KEY (msgid) REFERENCES ml_qa_repository(msgid)

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

ml_qa_delivery_archive

此表只用于 QAnywhere 应用程序。

小心
不要改动此表。

列	说明
msgid	VARCHAR(128)。全局唯一消息标识符。
seqno	BIGINT。用于对消息进行排序，这是进行真正排队所必需的。
address	VARCHAR(255)。目标接收者的地址。
clientaddress	VARCHAR(128)。地址的客户端部分。
client	VARCHAR(128)。作为当前客户端状态目标的客户端。
originator	VARCHAR(128)。发起客户端的名称。
priority	INTEGER。一个 0 到 9 之间的数字。优先级数字较高的消息将先于优先级数字较低的消息进行传送。缺省值为 4。
expires	TIMESTAMP。到期时间，在此时间之后将不传递消息。
kind	INTEGER。表示该消息是二进制消息 (**1**) 还是文本消息 (**2**)。
contentsize	BIGINT。消息的大小。对于二进制消息，这是字节数。对于文本消息，这是字符数。
status	INTEGER。消息的状态。可以是 1（待执行）、10（正在接收）、30（已到期）、40（已取消）、50（无法收到）或 60（已接收）。
statustime	TIMESTAMP。到达此状态的时间。该时间是到达此状态的客户端的本地时间。
syncstatus	INTEGER。就此消息而言客户端与服务器之间同步的状态。可以是 0（不同步）、1（同步）、2（不应同步消息）或 3（正在同步）。
receiverid	VARCHAR(128)。由接收方设置的标识消息接收方的标识符（如果有）。

约束

PRIMARY KEY(msgid, address)

FOREIGN KEY (msgid) REFERENCES ml_qa_repository(msgid)

ml_qa_global_props

此表只用于 QAnywhere 应用程序。它包含传输规则中使用的全局 *name-value* 对。

小心
不要改动此表。

列	说明
client	VARCHAR(128)。与属性关联的客户端。如果客户端值为 'ianywhere.server.defaultClient'，则表示属性是应用于所有客户端的全局属性。
name	VARCHAR(255)。属性的名称。
modifiers	INTEGER。用于对属性做进一步说明的位字段。目前，只有第一位用于表示不应进行同步的属性。所有其它位字段均留待将来使用。
value	LONG VARCHAR。属性值。
last_modified	TIMESTAMP。上次更改值的时间。对于表明何时需要将属性与客户端同步，这是必需的。

约束

PRIMARY KEY (client, name)

Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

ml_qa_notifications

此表只用于 QAnywhere 应用程序。通告程序使用它确定通知哪些 QAnywhere 客户端来启动同步。

小心
不要改动此表。

列	说明
user_id	INTEGER。
name	VARCHAR(128)。对客户端消息存储库进行唯一标识的 QAnywhere 客户端名。

约束

PRIMARY KEY (name)

ml_qa_repository

此表只用于 QAnywhere 应用程序。它存储消息及其属性。

小心
不要改动此表。

列	说明
msgid	VARCHAR(128)。全局唯一的消息标识符。
props	LONG BINARY。消息属性的编码。
content	LONG BINARY。消息的内容。文本消息以 UTF-8 格式编码。

约束

PRIMARY KEY(msgid)

Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

ml_qa_repository_archive

此表只用于 QAnywhere 应用程序。它存档消息及其属性。

小心
不要改动此表。

列	说明
msgid	VARCHAR(128)。全局唯一的消息标识符。
props	LONG BINARY。消息属性的编码。
content	LONG BINARY。消息的内容。文本消息以 UTF-8 格式编码。

约束

PRIMARY KEY(msgid)

ml_qa_repository_props

此表只用于 QAnywhere 应用程序。这是 ml_qa_repository 表中 props 列的扩展。属性仅在需要时由传输规则引擎扩展。如果未关联任何规则，则不会对属性进行扩展。

小心
不要改动此表。

列	说明
msgid	VARCHAR(128)。全局唯一的消息标识符。
name	VARCHAR(128)。属性的名称。如果属性名以 Unicode 格式提供，则系统会将其转换为数据库的本地字符集。
value	LONG VARCHAR。属性值。

约束

PRIMARY KEY(msgid, name)

FOREIGN KEY (msgid) REFERENCES ml_qa_repository(msgid)

Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

ml_qa_repository_props_archive

此表只用于 QAnywhere 应用程序。这是 ml_qa_repository 表中 props 列的扩展。属性仅在需要时由传输规则引擎扩展。如果未关联任何规则，则不会对属性进行扩展。

小心
不要改动此表。

列	说明
msgid	VARCHAR(128)。全局唯一的消息标识符。
name	VARCHAR(128)。属性的名称。如果属性名以 Unicode 格式提供，则系统会将其转换为数据库的本地字符集。
value	LONG VARCHAR。属性值。

约束

PRIMARY KEY(msgid, name)

FOREIGN KEY (msgid) REFERENCES ml_qa_repository(msgid)

ml_qa_repository_staging

此表只用于 QAnywhere 应用程序。它包含要使用 SQL Anywhere 9.0.1 版发送到 QAnywhere 客户端的消息。

小心
不要改动此表。

列	说明
seqno	BIGINT。用于对消息进行全面排序，这是真正排队所必需的。
msgid	VARCHAR(255)。全局唯一消息标识符。
destination	VARCHAR(128)。消息的地址。
originator	VARCHAR(128)。发起 MobiLink 用户的名称。
status	VARCHAR(128)。消息的状态。可以设为 [**pending**]、[**receiving**]、[**received**]、[**unreceivable**]、[**expired**] 或 [**cancelled**]。缺省值为 **pending**。
statustime	TIMESTAMP。上次更改状态的时间。
expires	TIMESTAMP。到期时间，在此时间之后将不传递消息。
priority	INTEGER。一个 0 到 9 之间的数字。该数字较大的消息总是先于该数字较小的消息进行传送。缺省值为 4。
props	LONG BINARY。消息属性的编码。
kind	INTEGER。表示该消息是二进制消息 (**1**) 还是文本消息 (**2**)。
content	LONG BINARY。消息的内容。文本消息以 UTF-8 格式编码。
contentsize	BIGINT。消息的大小。如果是二进制消息，此为字节数。如果是文本消息，此为字符数。
mluser	VARCHAR(128)。唯一标识远程数据库的 MobiLink 用户名。

约束

PRIMARY KEY(msgid)

Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

ml_qa_status_history

此表只用于 QAnywhere 应用程序。它包含消息状态更改的历史记录。

小心
不要改动此表。

列	说明
msgid	VARCHAR(128)。全局唯一的消息标识符。
address	VARCHAR(255)。目标接收者的地址。
status	INTEGER。消息的状态。可以是 1（待执行）、10（正在接收）、30（已到期）、40（已取消）、50（无法收到）或 60（已接收）。
statustime	TIMESTAMP。到达此状态的时间。该时间是到达此状态的客户端的本地时间。
servertime	TIMESTAMP。服务器接收到状态更改的时间。
details	VARCHAR(1000)。状态更改的详细信息（如果有）。
syncstatus	INTEGER。就此消息而言客户端与服务器之间同步的状态。可以是 **0**（不同步）、1（同步）、**2**（不应同步消息）或 **3**（正在同步）。

约束

PRIMARY KEY(msgid)

FOREIGN KEY (msgid) REFERENCES ml_qa_repository(msgid)

ml_qa_status_history_archive

此表只用于 QAnywhere 应用程序。它包含消息状态更改的历史记录。

小心
不要改动此表。

列	说明
msgid	VARCHAR(128)。全局唯一的消息标识符。
address	VARCHAR(255)。目标接收者的地址。
status	INTEGER。消息的状态。可以是 1（待执行）、10（正在接收）、30（已到期）、40（已取消）、50（无法收到）或 60（已接收）。
statustime	TIMESTAMP。到达此状态的时间。该时间是到达此状态的客户端的本地时间。
servertime	TIMESTAMP。服务器接收到状态更改的时间。
details	VARCHAR(1000)。状态更改的详细信息（如果有）。
syncstatus	INTEGER。就此消息而言客户端与服务器之间同步的状态。可以是 **0**（不同步）、1（同步）、**2**（不应同步消息）或 **3**（正在同步）。

约束

PRIMARY KEY(msgid)

FOREIGN KEY (msgid) REFERENCES ml_qa_repository(msgid)

Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

ml_qa_status_staging

此表只用于 QAnywhere 应用程序。它是在将状态变更与发起客户端（使用 SQL Anywhere 9.0.1 版）进行同步时使用的分段表。

小心
不要改动此表。

列	说明
msgid	VARCHAR(128)。全局唯一的消息标识符。
status	VARCHAR(255)。消息的状态。可以设为 [**pending**]、[**receiving**]、[**received**]、[**unreceivable**]、[**expired**] 或 [**cancelled**]。缺省值为 **pending**。
statustime	TIMESTAMP。上次更改状态的时间。
mluser	VARCHAR(128)。对远程数据库进行唯一标识的 MobiLink 用户名。

约束

PRIMARY KEY(msgid)

ml_script

此表存储所有脚本的内容。

列	说明
script_id	INTEGER。标识脚本的唯一整数。
script	TEXT。脚本文本。
script_language	VARCHAR(128)。脚本使用的脚本编写语言。脚本编写语言可以是 **sql**、**java** 或 **dnet**。
checksum	VARCHAR(64)。此列在内部使用。

约束

PRIMARY KEY(script_id)

Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

ml_script_version

此表存储与每个脚本版本关联的脚本的名称和说明。

列	说明
version_id	INTEGER。标识版本的唯一整数。
name	VARCHAR(128)。脚本版本的名称。
description	TEXT。对给定版本的说明。该说明并非为 MobiLink 所用，但它对应用程序特定的注释非常有用。例如，可以说明给定脚本版本的用途。

约束

PRIMARY KEY(version_id)

ml_scripts_modified

此表存储上次更改脚本表的时间。MobiLink 服务器会检查此表，以确定它是否必须装载新的脚本。

列	说明
last_modified	DATETIME。上次改动 ml_script、ml_table_script 或 ml_connection_script 系统表的时间。

注释

对于 DB2 主机统一数据库类型，该表称为 ml_script_modified。请参见“IBM DB2 主机系统表名称转换”一节第 656 页。

约束

PRIMARY KEY(last_modified)

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

ml_server

为每个在服务器群中运行的 MobiLink 服务器存储消息。

列	说明
server_id	INTEGER。标识 MobiLink 服务器名的服务器 ID 的唯一整数。该值在内部使用。
name	VARCHAR(128)。运行在服务器群上的活动的 MobiLink 服务器的名称。在整个服务器群上，该值必须唯一。
version	VARCHAR(10)。MobiLink 服务器的当前版本。
connection_info	VARCHAR(2048)。MobiLink 服务器的连接信息。
instance_key	VARCHAR(32)。指派的实例键。该值在内部使用。
start_time	TIMESTAMP。MobiLink 服务器的启动时间。
liveness	TIMESTAMP。上次客户端响应服务器的时间。

注释

除非有服务器群在运行，否则该表不包含数据。

约束

PRIMARY KEY(server_id)

ml_sis_sync_state

Sybase Central 使用此表为服务器启动的同步生成请求游标。

小心
不要改动此表。

列	说明
remote_id	VARCHAR(128)。唯一标识数据库的远程 ID。
subscription_id	VARCHAR(128)。subscription_id 是远程数据库生成的号码。对于 SQL Anywhere 客户端，此值与 SYS.ISYSSYNC 系统表中的 sync_id 相同。
publication_name	VARCHAR(128)。预订的用户定义的发布名。每次同步时，客户端都会发送每个 subscription_id 的发布名。 对于 10.0.0 之前版本的 UltraLite 客户端，此列始终为 <unknown>；对于 UltraLite 10 及其更高版本，此列为发布或字符串 ul_no_pub（如果没有发布）。
user_name	VARCHAR(128)。MobiLink 用户名。
last_upload	TIMESTAMP。上次对给定远程 ID 和 subscription_id 的统一数据库应用上载的时间。缺省值为 January 1, 1900, 00:00:00。
last_download	TIMESTAMP。上次对给定用户和 subscription_id 的统一数据库应用下载的时间。缺省值为 January 1, 1900, 00:00:00。

约束

PRIMARY KEY(remote_id, subscription_id)

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

ml_subscription

此表存储每个远程数据库的状态信息。

列	说明
rid	INTEGER。标识远程 ID 的唯一整数。此值在内部使用。
subscription_id	VARCHAR(128)。subscription_id 是远程数据库生成的号码。对于 SQL Anywhere 客户端，此值与 SYS.ISYSSYNC 系统表中的 sync_id 相同。 由于 UltraLite 客户端不使用预订，因此对于 UltraLite 客户端，此值为 UltraLite 发布 ID（如果是 10.0.0 版本及更高版本），或为 <unknown>（如果是版本 8 和 9）。
user_id	INTEGER。对给定 rid 和 subscription_id 执行上次同步的用户。可以使用 user_id 列查找上次成功运行同步的 MobiLink 用户。
progress	NUMERIC(20,0)。同步进度，也称为偏移、状态、序列号或进度计数器。
publication_name	VARCHAR(128)。预订的用户定义的发布名。每次同步时，客户端都会发送每个 subscription_id 的发布名。 对于 10.0.0 之前版本的 UltraLite 客户端，此列始终为 <unknown>；对于 UltraLite 10 及其更高版本，此列为发布或字符串 ul_no_pub（如果没有发布）。
last_upload_time	TIMESTAMP。上次对给定远程 ID 和 subscription_id 的统一数据库应用上载的时间。缺省值为 January 1, 1900, 00:00:00。
last_download_time	TIMESTAMP。上次对给定用户和 subscription_id 的统一数据库应用下载的时间。缺省值为 January 1, 1900, 00:00:00。请参见“如何生成和使用下载时间戳”一节第 122 页。

注释

在 SQL Anywhere 客户端中，进度是指远程数据库事务日志中的位置。它表示一个时间点，截至该时间点，预订的所有已提交操作均已从数据库上载完毕。dbmlsync 实用程序使用该偏移来决定要上载的数据。在 SQL Anywhere 远程数据库上，该偏移存储在 SYS.ISYSSYNC 系统表的 progress 列中。

请参见：

- “SYSSYNC 系统视图”一节 《SQL Anywhere 服务器 - SQL 参考》
- “进度偏移”一节 《MobiLink - 客户端管理》

Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

在 UltraLite 客户端中，进度是给定发布的同步序列号或进度计数器。此计数器表示哪些行进行了同步。发布每次同步时该值便会增加 1。此数字在 UltraLite 数据库内部使用，无法进行访问。

请参见“进度计数器”一节 《UltraLite - 数据库管理和参考》。

约束

PRIMARY KEY(rid, subscription_id)

FOREIGN KEY(rid) REFERENCES ml_database(rid)

FOREIGN KEY(user_id) REFERENCES ml_user(user_id)

另请参见

- “远程 ID”一节 《MobiLink - 客户端管理》

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

ml_table

此表存储远程表的名称。该列表包含标记为已同步表的所有表。

列	说明
table_id	INTEGER。标识表的唯一整数。
name	VARCHAR(128)。分配给表的名称。

约束

PRIMARY KEY(table_id)

ml_table_script

对于给定脚本版本，此表会将表脚本与给定表和事件相关联。

列	说明
version_id	INTEGER。标识脚本版本的数字。
table_id	INTEGER。标识表的数字。
event	VARCHAR(128)。事件的名称。
script_id	INTEGER。标识脚本的数字。脚本存储在 ml_script 表中。

注释

存在一个系统视图 ml_table_scripts，通过该视图可更轻松地查看 ml_table_script MobiLink 系统表的内容。

约束

FOREIGN KEY(version_id) REFERENCES ml_script_version(version_id)

FOREIGN KEY(table_id) REFERENCES ml_table(table_id)

FOREIGN KEY(script_id) REFERENCES ml_script(script_id)

Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

ml_user

存储注册用户及其散列口令。

列	说明
user_id	INTEGER。标识用户的唯一整数。该值在内部使用。
name	VARCHAR(128)。已注册的用户名。
hashed_password	BINARY(32)。模糊处理格式的口令。尽管该值可以为空，但建议您指定一个口令。

注释

该表存储 MobiLink 服务器已知的所有已注册用户。用户名在每次同步时由客户端发送，客户端可以选择发出用户口令以进行验证。

MobiLink 服务器使用其自己的算法对用户口令进行散列处理。

不要直接将任何口令不为空值的用户名插入此表。用户名可使用实用程序 mluser 来添加。

约束

PRIMARY KEY(user_id)

另请参见

- “MobiLink 用户验证实用程序 (mluser)”一节第 650 页

远程数据库和统一数据库之间的 MobiLink 数据映射

目录

Adaptive Server Enterprise 数据映射 696
IBM DB2 LUW 数据映射 704
IBM DB2 主机数据映射 711
Microsoft SQL Server 数据映射 720
MySQL 数据映射 726
Oracle 数据映射 731

Adaptive Server Enterprise 数据映射

映射到 Adaptive Server Enterprise 统一数据类型

下表列出了 SQL Anywhere 和 UltraLite 远程数据类型与 Adaptive Server Enterprise 统一数据类型的映射关系。例如，远程数据库中类型为 FLOAT 的列对应统一数据库中的类型 REAL。

最大列长度 (MCL) 取决于 Adaptive Server Enterprise 的页面大小。如果页面大小为 2K，则 MCL 为 1954；如果页面大小为 4K，则 MCL 为 4002。有关 MCL 的信息，请参见 Adaptive Server Enterprise 文档。

SQL Anywhere 或 UltraLite 数据类型	Adaptive Server Enterprise 数据类型	注释
BIGINT	NUMERIC(20)[1] 或 BIGINT[2]	
BIT	BIT	
BINARY(*n*=<MCL)	BINARY(*n*)	
BINARY(*n*>MCL)	IMAGE	
CHAR(*n*=<MCL)	VARCHAR(*n*)	
CHAR(*n*>MCL)	TEXT	下载时，确保值不要过长。
DATE	DATE[3] 或 DATETIME[4]	对于 Adaptive Server Enterprise DATETIME，年份必须在 1753-9999 范围内。 对于 SQL Anywhere 和 UltraLite，时间值格式必须为 00:00:00。
DATETIME	DATETIME	Adaptive Server Enterprise DATETIME 值精确到 1/300 秒。小数秒的最后一位数始终为 0、3 或 6。其它数字将舍入为这三个数之一，即 0 和 1 舍入为 0，2、3 和 4 舍入为 3，5、6、7 和 8 舍入为 6，9 舍入为 10。 下载时，SQL Anywhere 会保持来自 Adaptive Server Enterprise 的原始值，但上载时，这些值可能与原始值不完全一致。 如果 DATETIME 用于主键，冲突解决可能会失败。为了成功同步 DATETIME，您应将小数秒舍入到 10 毫秒。此外，年份必须在 1753-9999 范围内。

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

SQL Anywhere 或 UltraLite 数据类型	Adaptive Server Enterprise 数据类型	注释
DECIMAL(*p*<39, *s*)	DECIMAL(*p*,*s*)	Adaptive Server Enterprise NUMERIC 可以精确到 1 至 38 位数 (*p*<39)。
DECIMAL(*p*>=39,*s*)		Adaptive Server Enterprise 中没有相应的数据类型。
DOUBLE	DOUBLE PRECISION	
FLOAT(*p*)	FLOAT(*p*)	
IMAGE	IMAGE	
INTEGER	INTEGER	
LONG BINARY	IMAGE	
LONG NVARCHAR	UNITEXT	
LONG VARBIT	TEXT	
LONG VARCHAR	TEXT	
MONEY	MONEY	
NCHAR(*c*=<MCL)	UNIVARCHAR(*c*/2)	
NCHAR(*c*>MCL)	UNITEXT	下载时，确保值不要过长。
NTEXT	UNITEXT	
NUMERIC(*p*<39, *s*)	NUMERIC(*p*,*s*)	Adaptive Server Enterprise 的 DECIMAL 可以精确到 1 至 38 位数 (*p*<39)。
NUMERIC(*p*>=39,*s*)		
NVARCHAR(*c*=<MCL)	UNIVARCHAR(*c*/2)	
NVARCHAR(*c*>MCL)	UNITEXT	下载时，确保值不要过长。
REAL	REAL	

SQL Anywhere 或 UltraLite 数据类型	Adaptive Server Enterprise 数据类型	注释
SMALLDATETIME	DATETIME[4]	SQL Anywhere 和 UltraLite SMALLDATETIME 作为 TIMESTAMP 来实现。 Adaptive Server Enterprise DATETIME 精确到分钟。29.998 或更低的秒值将向下舍入为最接近的分钟；29.999 或更高的秒值将向上舍入为最接近的分钟。SQL Anywhere 或 UltraLite SMALLDATETIME 精确到微秒。为了成功地进行同步，SQL Anywhere 或 UltraLite SMALLDATETIME 必须舍入到分钟。此外，年份必须在 1753-9999 范围内。
SMALLINT	SMALLINT	
SMALLMONEY	SMALLMONEY	
TEXT	TEXT	
TIME	TIME[3] 或 DATETIME[4]	Adaptive Server Enterprise TIME 值精确到 1/300 秒。小数秒的最后一位数始终为 0、3 或 6。其它数字将舍入为这三个数之一，即 0 和 1 舍入为 0，2、3 和 4 舍入为 3，5、6、7 和 8 舍入为 6，9 舍入为 10。下载时，SQL Anywhere 会保留 Adaptive Server Enterprise 的原始值，但上载时，这些值可能与原始值不完全一致。如果 TIME 用于主键，冲突解决可能会失败。为了成功同步 TIME，您应将小数秒舍入到 10 毫秒。

Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

SQL Anywhere 或 UltraLite 数据类型	Adaptive Server Enterprise 数据类型	注释
TIMESTAMP	DATETIME	Adaptive Server Enterprise DATETIME 值精确到 1/300 秒。小数秒的最后一位数始终为 0、3 或 6。其它数字将舍入为这三个数之一，即 0 和 1 舍入为 0，2、3 和 4 舍入为 3，5、6、7 和 8 舍入为 6，9 舍入为 10。 下载时，SQL Anywhere 会保持来自 Adaptive Server Enterprise 的原始值，但上载时，这些值可能与原始值不完全一致。 如果 DATETIME 用于主键，冲突解决可能会失败。为了成功同步 DATETIME，您应将小数秒舍入到 10 毫秒。此外，年份必须在 1753-9999 范围内。
TINYINT	TINYINT	
UNIQUEIDENTIFIER	CHAR(36)	
UNIQUEIDENTIFIERSTR	CHAR(36)	不使用 UNIQUEIDENTIFIERSTR。请改用 UNIQUEIDENTIFIER。
UNSIGNED BIGINT	NUMERIC(20)[1] 或 UNSIGNED BIGINT[2]	
UNSIGNED INTEGER	UNSIGNED INT	
UNSIGNED SMALLINT	UNSIGNED SMALLINT	
UNSIGNED TINYINT	TINYINT	
VARBINARY(*n*=<MCL)	VARBINARY	
VARBINARY(*n*>MCL)	IMAGE	
VARBIT(*n*=<MCL)	VARCHAR(*n*)	
VARBIT(*n*>MCL)	TEXT	
VARCHAR(*n*=<MCL)	VARCHAR(*n*)	
VARCHAR(*n*>MCL)	TEXT	

SQL Anywhere 或 UltraLite 数据类型	Adaptive Server Enterprise 数据类型	注释
XML	TEXT	

[1] 只适用于 15.0 版本之前的 Adaptive Server Enterprise。

[2] 只适用于 Adaptive Server Enterprise 版本 15.0 或更高版本。

[3] 只适用于 Adaptive Server Enterprise 版本 12.5.1 或更高版本。

[4] 只适用于 12.5.1 版本之前的 Adaptive Server Enterprise。

映射到 SQL Anywhere 或 UltraLite 远程数据类型

下表列出了 Adaptive Server Enterprise 统一数据类型与 SQL Anywhere 和 UltraLite 远程数据类型的映射关系。例如，统一数据库中类型为 DOUBLE PRECISION 的列对应远程数据库中的类型 DOUBLE。

Adaptive Server Enterprise 数据类型	SQL Anywhere 或 UltraLite 数据类型	注释
BIGINT[1]	BIGINT	
BINARY(*n*)	BINARY(*n*)	
BIT	BIT	
CHAR(*n*)	VARCHAR(*n*)	SQL Anywhere CHAR/NCHAR 和 Adaptive Server Enterprise CHAR/NCHAR 之间不存在对等关系。SQL Anywhere CHAR/NCHAR 与 VARCHAR/ NVARCHAR 对等。不应在要同步的统一数据库列中使用 CHAR/NCHAR。如果必须使用非 SQL Anywhere CHAR/NCHAR，请用 -b 选项运行 MobiLink 服务器。
DATE	DATE	对于 SQL Anywhere 和 UltraLite，时间值格式必须为 00:00:00。

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

Adaptive Server Enterprise 数据类型	SQL Anywhere 或 UltraLite 数据类型	注释
DATETIME	DATETIME	Adaptive Server Enterprise DATETIME 值精确到 1/300 秒。小数秒的最后一位数始终为 0、3 或 6 之一。其它数字将舍入为这三个数之一，即 0 和 1 舍入为 0，2、3 和 4 舍入为 3，5、6、7 和 8 舍入为 6，9 舍入为 10。 下载时，SQL Anywhere 会保持来自 Adaptive Server Enterprise 的原始值，但上载时，这些值可能与原始值不完全一致。冲突解决可能会失败。为了成功同步 DATETIME，您应将小数秒舍入到 10 毫秒。此外，年份必须在 1753-9999 范围内。
DECIMAL(*p*,*s*)	DECIMAL(*p*,*s*)	
DOUBLE PRECISION	DOUBLE	
FLOAT(*p*)	FLOAT(*p*)	
IMAGE	LONG BINARY	
INT	INT	
MONEY	MONEY	
NCHAR(*n*)	VARCHAR(*n*)	与 SQL Anywhere NCHAR 和 NVARCHAR 不同，Adaptive Server Enterprise NCHAR 和 NVARCHAR 存储多字节本国字符串。在多字节环境中，使用 SQL Anywhere 或 UltraLite VARCHAR。
NUMERIC(*p*,*s*)	NUMERIC(*p*,*s*)	
NVARCHAR(*n*)	VARCHAR(*n*)	与 SQL Anywhere NCHAR 和 NVARCHAR 不同，Adaptive Server Enterprise NCHAR 和 NVARCHAR 存储多字节本国字符串。在多字节环境中，使用 SQL Anywhere 或 UltraLite VARCHAR。
REAL	REAL	

Adaptive Server Enterprise 数据类型	SQL Anywhere 或 UltraLite 数据类型	注释
SMALLDATETIME	SMALLDATETIME	SQL Anywhere 和 UltraLite SMALLDATETIME 作为 TIMESTAMP 来实现。 Adaptive Server Enterprise SMALLDATETIME 精确到分钟。29.998 或更低的秒值将向下舍入为最接近的分钟；29.999 或更高的秒值将向上舍入为最接近的分钟。SQL Anywhere 或 UltraLite SMALLDATETIME 精确到微秒。为了成功地进行同步，SQL Anywhere 或 UltraLite SMALLDATETIME 必须舍入到分钟。此外，年份必须在 1900-2078 范围内。
SMALLINT	SMALLINT	
SMALLMONEY	SMALLMONEY	
TEXT	LONG VARCHAR	
TIME	TIME	Adaptive Server Enterprise TIME 值精确到 1/300 秒。小数秒的最后一位数始终为 0、3 或 6 之一。其它数字将舍入为这三个数之一，即 0 和 1 舍入为 0，2、3 和 4 舍入为 3，5、6、7 和 8 舍入为 6，9 舍入为 10。 下载时，SQL Anywhere 会保持来自 Adaptive Server Enterprise 的原始值，但上载时，这些值可能与原始值不完全一致。冲突解决可能会失败。为了成功同步 TIME，建议您将小数秒舍入到 10 毫秒。
TIMESTAMP	VARBINARY(8)	在 Adaptive Server Enterprise 内，TIMESTAMP 是一个二进制计数器，它会随行的每一次更改而相应递增。因此，每个表只能包含一个 TIMESTAMP 列，并且对其进行同步没有任何意义。如果必须对其进行同步，请将其映射到 SQL Anywhere 或 UltraLite 中的 VARBINARY(8) 数据类型。 不能显式地插入或更新 TIMESTAMP 列，因为它由服务器进行维护。当您为含有这种列的表实现上载脚本时，请牢记这一点。

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

Adaptive Server Enterprise 数据类型	SQL Anywhere 或 UltraLite 数据类型	注释
TINYINT	TINYINT	
UNSIGNED BIGINT[1]	UNSIGNED BIGINT	
UNSIGNED INT[1]	UNSIGNED INT	
UNSIGNED SMALLINT[1]	UNSIGNED SMALLINT	
VARBINARY(*n*)	VARBINARY(*n*)	
VARCHAR(*n*)	VARCHAR(*n*)	
UNICHAR(*n*)	NVARCHAR(*n*)	在 UltraLite 中不可用。
UNITEXT[1]	LONG NVARCHAR	在 UltraLite 中不可用。
UNIVARCHAR(*n*)	NVARCHAR(*n*)	在 UltraLite 中不可用。

[1] 只适用于 15.0 版本之前的 Adaptive Server Enterprise。

IBM DB2 LUW 数据映射

映射到 IBM DB2 LUW 统一数据类型

下表列出了 SQL Anywhere 和 UltraLite 远程数据类型与 IBM DB2 LUW 统一数据类型的映射关系。例如，远程数据库中类型为 BIT 的列对应统一数据库中的类型 SMALLINT。

在创建 DB2 表时，您需要注意 DB2 页面大小。DB2 的最大行长度 (MRL) 取决于此页面大小：页面大小为 4K 时 MRL 是 4005，8K 时是 8101，16K 时是 16293，32K 时是 32677。表中所有列的长度都不能超出上述限制。如果表中包含 BLOB 或 CLOB 列，则使用 LOB 定位器而不是直接使用 BLOB 或 CLOB 数据计算行长度。有关详细信息，请参见 DB2 文档。

SQL Anywhere 或 UltraLite 数据类型	IBM DB2 LUW 数据类型	注释
BIGINT	BIGINT	
BINARY(*n*<MRL)	VARCHAR(*n*) FOR BIT DATA	
BINARY(*n*>=MRL)	BLOB(*n*)	
BIT	SMALLINT	
CHAR(*n*<MRL)	VARCHAR(*n*)	
CHAR(*n*>=MRL)	CLOB(*n*)	DB2 值可以长于 SQL Anywhere 或 UltraLite 值，因此下载时请确保值不要过大。
DATE	DATE	对于 SQL Anywhere 和 UltraLite，时间值格式必须为 00:00:00。
DATETIME	TIMESTAMP	
DECIMAL(*p*<32, *s*)	DECIMAL(*p*,*s*)	SQL Anywhere DECIMAL 的精度介于 1 和 127 之间。DB2 DECIMAL 的最大精度为 31。
DECIMAL(*p*>=32,*s*)		精度大于 31 的任何 SQL Anywhere DECIMAL 数据都不能同步到 DB2。

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

SQL Anywhere 或 UltraLite 数据类型	IBM DB2 LUW 数据类型	注释
DOUBLE	DOUBLE	DOUBLE 是一种需要舍入的不精确的数字数据类型。使用不同类型的计算机时，DOUBLE 的基础存储经常不同，从而导致不同的舍入。因为主键查找等同性，所以在主键中使用 DOUBLE 是不好的选择。同步环境中尤其如此，因为统一数据库经常运行在与远程数据库不同的硬件上。
FLOAT(1-24)	REAL	如果统一数据库和远程数据库不允许存在完全相同（不精确）的值，FLOAT 就会导致问题。我们并没有对所有可能的值进行测试，因此，务必要谨慎。为避免出现问题，请不要将这些类型用作主键的一部分。
FLOAT(25-53)	DOUBLE	如果统一数据库和远程数据库不允许存在完全相同（不精确）的值，FLOAT 就会导致问题。我们并没有对所有可能的值进行测试，因此，务必要谨慎。为避免出现问题，请不要将这些类型用作主键的一部分。
IMAGE	BLOB(*n*)	
INTEGER	INTEGER	
LONG BINARY	BLOB(*n*)	
LONG NVARCHAR	CLOB(*n*)	DB2 中没有相应的数据类型。如果 DB2 字符集是 Unicode，SQL Anywhere LONG NVARCHAR 可同步到 DB2 CLOB。UltraLite 没有 LONG NVARCHAR。
LONG VARBIT	CLOB(*n*)	
LONG VARCHAR	CLOB(*n*)	
MONEY	DECIMAL(19,4)	

SQL Anywhere 或 UltraLite 数据类型	IBM DB2 LUW 数据类型	注释
NCHAR(*c*)	VARCHAR(*n*) 或 CLOB(*n*)	DB2 中没有相应的数据类型。如果 DB2 字符集是 Unicode，NCHAR 可同步到 DB2 VARCHAR 或 CLOB。SQL Anywhere NCHAR 的大小是字符数，DB2 VARCHAR 的大小是字节数。如果映射到 VARCHAR，NCHAR 的总字节数不能大于 MRL。否则，NCHAR 应映射到 CLOB。NCHAR(*c*) 中的字节数很难计算，但可以近似为 *c*=*n*/4。通常，如果 *c* 小于 MRL/4，则映射到 VARCHAR(*n*)；如果 *c* 大于或等于 MRL/4，则映射到 CLOB(*n*)。
NUMERIC(*p*<32, *s*)	NUMERIC(*p*,*s*)	
NUMERIC(*p*>=32,*s*)		DB2 中没有相应的数据类型。
NTEXT	CLOB(*n*)	DB2 中没有相应的数据类型。如果 DB2 字符集是 Unicode，则 NTEXT 可同步到 DB2 CLOB。
NVARCHAR(*c*)	VARCHAR(*n*) 或 CLOB(*n*)	DB2 中没有相应的数据类型。如果 DB2 字符集是 Unicode，NVARCHAR 可同步到 DB2 VARCHAR 或 CLOB。SQL Anywhere NVARCHAR 的大小是字符数，DB2 VARCHAR 的大小是字节数。如果映射到 VARCHAR，则 NVARCHAR 的总字节数不能大于 MRL。否则，NVARCHAR 应映射到 CLOB。NVARCHAR(*c*) 中的字节数很难计算，但可以近似为 *c*=*n*/4。通常，如果 *c* 小于 MRL/4，则映射到 VARCHAR(*n*)；如果 *c* 大于或等于 MRL/4，则映射到 CLOB(*n*)。
REAL	REAL	如果统一数据库和远程数据库不允许存在完全相同（不精确）的值，REAL 就会导致问题。我们并没有对所有可能的值进行测试，因此，务必要谨慎。为避免出现问题，请不要将这些类型用作主键的一部分。
SMALLDATETIME	TIMESTAMP	
SMALLINT	SMALLINT	
SMALLMONEY	DECIMAL(10,4)	

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

SQL Anywhere 或 UltraLite 数据类型	IBM DB2 LUW 数据类型	注释
TEXT	CLOB(*n*)	
TIME	TIMESTAMP 或 TIME	带小数秒的 SQL Anywhere 和 UltraLite TIME 值需要 DB2 TIMESTAMP。小数秒始终为零的 SQL Anywhere 和 UltraLite 时间值可以使用 DB2 TIME。
TIMESTAMP	TIMESTAMP	
TINYINT	SMALLINT	对于下载，DB2 值必须是非负数。
UNIQUEIDENTIFIER	CHAR(36)	
UNIQUEIDENTIFIERSTR	CHAR(36)	不建议将 UNIQUEIDENTIFIERSTR 用于 DB2。请改用 UNIQUEIDENTIFIER。
UNSIGNED BIGINT	DECIMAL(20)	对于下载，DB2 值必须是非负数。
UNSIGNED INTEGER	DECIMAL(11)	对于下载，DB2 值必须是非负数。
UNSIGNED SMALLINT	DECIMAL(5)	对于下载，DB2 值必须是非负数。
UNSIGNED TINYINT	SMALLINT	对于下载，DB2 值必须是非负数。
VARBINARY(*n*<MRL)	VARCHAR(*n*) FOR BIT DATA	
VARBINARY(*n*>=MRL)	BLOB(*n*)	
VARBIT(*n*<MRL)	VARCHAR(*n*)	
VARBIT(*n*>=MRL)	CLOB(*n*)	
VARCHAR(*n*<MRL)	VARCHAR(*n*)	
VARCHAR(*n*>=MRL)	CLOB(*n*)	DB2 值可以长于 SQL Anywhere 或 UltraLite 值，因此下载时请确保值不要过大。
XML	CLOB(*n*)	

映射到 SQL Anywhere 或 UltraLite 远程数据类型

下表列出了 IBM DB2 LUW 统一数据类型与 SQL Anywhere 和 UltraLite 远程数据类型的映射关系。例如，统一数据库中类型为 INT 的列对应远程数据库中的类型 INTEGER。

在创建 DB2 表时，您需要注意 DB2 页面大小。DB2 的最大行长度取决于此页面大小：页面大小为 4K 时 MRL 是 4005，8K 时是 8101，16K 时是 16293，32K 时是 32677。表中所有列的长度都不能超出上述限制。如果表中包含 BLOB 或 CLOB 列，则使用 LOB 定位器而不是直接使用 BLOB 或 CLOB 数据计算行长度。有关详细信息，请参见 DB2 文档。

IBM DB2 LUW 数据类型	SQL Anywhere 或 UltraLite 数据类型	注释
BLOB	LONG BINARY	
BIGINT	BIGINT	
CHAR(*n*)	VARCHAR(*n*)	SQL Anywhere 中没有与 DB2 CHAR 对等的类型。不应在要同步的统一数据库列中使用 CHAR。如果必须同步 DB2 CHAR 列，请使用 -b 选项运行 MobiLink 服务器。
CHAR(*n*) FOR BIT DATA	BINARY(*n*)	
CLOB(*n*)	LONG VARCHAR	
DATE	DATE	对于 SQL Anywhere 和 UltraLite，时间值格式必须为 00:00:00。
DBCLOB(*n*)	LONG VARCHAR	数据类型 DBCLOB(*n*) 仅用于双字节字符。SQL Anywhere 没有相对应的数据类型。DB2 字符集为 Unicode 时，DBCLOB(*n*) 等同于 CLOB。
DECIMAL(*p*,*s*)	DECIMAL(*p*,*s*)	
DOUBLE	DOUBLE	DOUBLE 是一种需要舍入的不精确的数字数据类型。使用不同类型的计算机时，DOUBLE 的基础存储经常不同，从而导致不同的舍入。因为主键查找等同性，所以在主键中使用 DOUBLE 是不好的选择。同步环境中尤其如此，因为统一数据库经常运行在与远程数据库不同的硬件上。
FLOAT	DOUBLE	如果统一数据库和远程数据库不允许存在完全相同（不精确）的值，FLOAT 就会导致问题。我们并没有对所有可能的值进行测试，因此，务必要谨慎。为避免出现问题，请不要将这些类型用作主键的一部分。

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

IBM DB2 LUW 数据类型	SQL Anywhere 或 UltraLite 数据类型	注释
GRAPHIC(*n*)	VARCHAR(2*n*)	DB2 GRAPHIC 会使用空白进行填充，而 SQL Anywhere CHAR 则不会。我们建议您不要使用此数据类型。 数据类型 GRAPHIC 仅用于双字节字符。SQL Anywhere 没有相对应的数据类型。DB2 字符集为 Unicode 时，GRAPHIC 等同于 CHAR。
INT	INTEGER	
LONG VARCHAR	VARCHAR(32700)	
LONG VARCHAR FOR BIT DATA	VARBINARY(32700)	
LONG VARGRAPHIC(*n*)	VARCHAR(32700)	数据类型 LONG VARGRAPHIC 仅用于双字节字符。SQL Anywhere 没有相对应的数据类型。DB2 字符集为 Unicode 时，LONG VARGRAPHIC 等同于 LONG VARCHAR。
NUMERIC(*p*,*s*)	NUMERIC(*p*,*s*)	
REAL	REAL	如果统一数据库和远程数据库不允许存在完全相同（不精确）的值，REAL 就会导致问题。我们并没有对所有可能的值进行测试，因此，务必要谨慎。为避免出现问题，请不要将这些类型用作主键的一部分。
SMALLINT	SMALLINT	
TIME	TIME	SQL Anywhere TIME 值中的小数秒值在下载时会被截断。要避免出现问题，请不要使用小数秒。
TIMESTAMP	TIMESTAMP	
VARCHAR(*n*)	VARCHAR(*n*)	
VARCHAR(*n*) FOR BIT DATA	VARBINARY(*n*)	

IBM DB2 LUW 数据类型	SQL Anywhere 或 UltraLite 数据类型	注释
VARGRAPHIC(*n*)	VARCHAR(2*n*)	数据类型 VARGRAPHIC 仅用于双字节字符。SQL Anywhere 没有相对应的数据类型。DB2 字符集为 Unicode 时，VARGRAPHIC 等同于 VARCHAR。

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

IBM DB2 主机数据映射

映射到 DB2 主机统一数据类型

下表列出了 SQL Anywhere 和 UltraLite 远程数据类型与 DB2 主机统一数据类型的映射关系。例如，远程数据库中类型为 BIT 的列对应统一数据库中的类型 SMALLINT。

SQL Anywhere 或 UltraLite 数据类型	IBM DB2 主机数据类型	注释
BIGINT	DECIMAL(20)	
BINARY(*n*<MRL)	VARCHAR(*n*) FOR BIT DATA	在创建 DB2 表时，您需要注意 DB2 页面大小。DB2 的最大行长度取决于此页面大小：页面大小为 4K 时 MRL 是 4005，8K 时是 8101，16K 时是 16293，32K 时是 32677。表中所有列的长度都不能超出上述限制。如果表中包含 BLOB 或 CLOB 列，则使用 LOB 定位器而不是直接使用 BLOB 或 CLOB 数据计算行长度。有关详细信息，请参见 DB2 文档。
BINARY(*n*>=MRL)	BLOB(*n*)	在创建 DB2 表时，您需要注意 DB2 页面大小。DB2 的最大行长度取决于此页面大小：页面大小为 4K 时 MRL 是 4005，8K 时是 8101，16K 时是 16293，32K 时是 32677。表中所有列的长度都不能超出上述限制。如果表中包含 BLOB 或 CLOB 列，则使用 LOB 定位器而不是直接使用 BLOB 或 CLOB 数据计算行长度。有关详细信息，请参见 DB2 文档。
BIT	SMALLINT	
CHAR(*n*<MRL)	VARCHAR(*n*)	MRL 是 DB2 的最大行长度。 DB2 VARCHAR 最多可以保存 32672 个字节，具体情况取决于页面大小。 DB2 的最大行长度 (MRL) 取决于此页面大小：页面大小为 4K 时 MRL 是 4005，8K 时是 8101，16K 时是 16293，32K 时是 32677。有关详细信息，请参见 DB2 文档。 SQL Anywhere CHAR 与 SQL Anywhere VARCHAR 完全相同。

SQL Anywhere 或 UltraLite 数据类型	IBM DB2 主机数据类型	注释
CHAR(*n*>=MRL)	CLOB(*n*)	DB2 的最大行长度 (MRL) 取决于此页面大小：页面大小为 4K 时 MRL 是 4005，8K 时是 8101，16K 时是 16293，32K 时是 32677。有关详细信息，请参见 DB2 文档。 SQL Anywhere CHAR 与 SQL Anywhere VARCHAR 完全相同。 DB2 CLOB 值可以比 SQL Anywhere 或 UltraLite 值长。您必须确保下载的值不能过大。
DATE	DATE	对于 SQL Anywhere 和 UltraLite，时间值格式必须为 00:00:00。
DATETIME	TIMESTAMP	
DECIMAL(*p*<32, *s*)	DECIMAL(*p*,*s*)	SQL Anywhere DECIMAL 的精度介于 1 和 127 之间。DB2 DECIMAL 的最大精度为 31。任何 DECIMAL 精度大于 31 的 SQL Anywhere 数据都不能同步到 DB2。
DECIMAL(*p*>=32,*s*)		精度大于 31 的任何 SQL Anywhere DECIMAL 数据都不能同步到 DB2。
DOUBLE	DOUBLE	如果统一数据库和远程数据库不允许存在完全相同（不精确）的值，DOUBLE 就会导致问题。我们并没有对所有可能的值进行测试，因此，务必要谨慎。为避免出现问题，请不要将这些类型用作主键的一部分。 为了成功地进行同步，SQL Anywhere/ UltraLite DOUBLE 值必须在 DB2 主机的 DOUBLE 值域之内。
FLOAT(1-24)	REAL	如果统一数据库和远程数据库不允许存在完全相同（不精确）的值，FLOAT 就会导致问题。我们并没有对所有可能的值进行测试，因此，务必要谨慎。为避免出现问题，请不要将这些类型用作主键的一部分。 为了成功地进行同步，DB2 主机的 REAL 值必须在 SQL Anywhere/UltraLite REAL 的值域之内。

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

SQL Anywhere 或 UltraLite 数据类型	IBM DB2 主机数据类型	注释
FLOAT(25-53)	DOUBLE	如果统一数据库和远程数据库不允许存在完全相同（不精确）的值，FLOAT 就会导致问题。我们并没有对所有可能的值进行测试，因此，务必要谨慎。为避免出现问题，请不要将这些类型用作主键的一部分。 为了成功地进行同步，SQL Anywhere/ UltraLite DOUBLE 值必须在 DB2 主机的 DOUBLE 值域之内。
IMAGE	BLOB(*n*)	
INTEGER	INTEGER	
LONG BINARY	BLOB(*n*)	
LONG NVARCHAR	CLOB(*n*)	DB2 中没有相应的数据类型。如果 DB2 字符集是 Unicode，SQL Anywhere long NVARCHAR 可同步到 DB2 CLOB。UltraLite 没有 LONG NVARCHAR。
LONG VARBIT	BLOB(*n*)	
LONG VARCHAR	CLOB(*n*)	
MONEY	DECIMAL(19.4)	
NCHAR(*c*)	VARCHAR(*n*) 或 CLOB(*n*)	SQL Anywhere NCHAR 与 SQL Anywhere NVARCHAR 完全相同。 DB2 中没有相应的数据类型。如果 DB2 字符集是 Unicode，NCHAR 可同步到 DB2 VARCHAR 或 CLOB。SQL Anywhere NCHAR 的大小是字符数，DB2 VARCHAR 的大小是字节数。如果映射到 VARCHAR，则 NCHAR 的总字节数不能大于 MRL。否则，NCHAR 应映射到 CLOB。NCHAR(*c*) 中的字节数很难计算，但可以近似为 *c*=*n*/4。通常，如果 *c* 小于 MRL/4，则映射到 VARCHAR(*n*)；如果 *c* 大于或等于 MRL/4，则映射到 CLOB(*n*)。

SQL Anywhere 或 UltraLite 数据类型	IBM DB2 主机数据类型	注释
NTEXT	CLOB(*n*)	SQL Anywhere NTEXT 与 SQL Anywhere LONG NVARCHAR 完全相同。 DB2 中没有相应的数据类型。如果 DB2 字符集是 Unicode，NCHAR 可同步到 DB2 VARCHAR 或 CLOB。SQL Anywhere NCHAR 的大小是字符数，DB2 VARCHAR 的大小是字节数。如果映射到 VARCHAR，则 NCHAR 的总字节数不能大于 MRL。否则，NCHAR 应映射到 CLOB。NCHAR(*c*) 中的字节数很难计算，但可以近似为 *c*=*n*/4。通常，如果 *c* 小于 MRL/4，则映射到 VARCHAR(*n*)；如果 *c* 大于或等于 MRL/4，则映射到 CLOB(*n*)。
NUMERIC(*p*<32, *s*)	NUMERIC(*p*,*s*)	
NUMERIC(*p*>=32,*s*)		DB2 中没有相应的数据类型。
NVARCHAR(*c*)	VARCHAR(*n*) 或 CLOB(*n*)	DB2 中没有相应的数据类型。如果 DB2 字符集是 Unicode，NCHAR 可同步到 DB2 VARCHAR 或 CLOB。SQL Anywhere NCHAR 的大小是字符数，DB2 VARCHAR 的大小是字节数。如果映射到 VARCHAR，则 NCHAR 的总字节数不能大于 MRL。否则，NCHAR 应映射到 CLOB。NCHAR(*c*) 中的字节数很难计算，但可以近似为 *c*=*n*/4。通常，如果 *c* 小于 MRL/4，则映射到 VARCHAR(*n*)；如果 *c* 大于或等于 MRL/4，则映射到 CLOB(*n*)。
REAL	REAL	如果统一数据库和远程数据库不允许存在完全相同（不精确）的值，REAL 就会导致问题。我们并没有对所有可能的值进行测试，因此，务必要谨慎。为避免出现问题，请不要将这些类型用作主键的一部分。 为了成功地进行同步，DB2 主机的 REAL 值必须在 SQL Anywhere/UltraLite REAL 的值域之内。
SMALLDATETIME	TIMESTAMP	
SMALLINT	SMALLINT	

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

SQL Anywhere 或 UltraLite 数据类型	IBM DB2 主机数据类型	注释
SMALLMONEY	DECIMAL(10,4)	
TEXT	CLOB(*n*)	SQL Anywhere TEXT 与 SQL Anywhere LONG VARCHAR 完全相同。
TIME	TIMESTAMP 或 TIME	带小数秒的 SQL Anywhere 和 UltraLite TIME 值需要 DB2 TIMESTAMP。小数秒始终为零的 SQL Anywhere 和 UltraLite 时间值可以使用 DB2 TIME。
TIMESTAMP	TIMESTAMP	
TINYINT	SMALLINT	
UNIQUEIDENTIFIER	CHAR(36)	
UNIQUEIDENTIFIERSTR	CHAR(36)	不建议将 UNIQUEIDENTIFIERSTR 用于 DB2。请改用 UNIQUEIDENTIFIER。
UNSIGNED BIGINT	DECIMAL(20)	DB2 值必须是非负数。
UNSIGNED INTEGER	DECIMAL(11)	DB2 值必须是非负数。
UNSIGNED SMALLINT	DECIMAL(5)	DB2 值必须是非负数。
UNSIGNED TINYINT	SMALLINT	DB2 值必须是非负数。
VARBINARY(*n*<MRL)	VARCHAR(*n*) FOR BIT DATA	在创建 DB2 表时，您需要注意 DB2 页面大小。DB2 的最大行长度取决于此页面大小：页面大小为 4K 时 MRL 是 4005，8K 时是 8101，16K 时是 16293，32K 时是 32677。表中所有列的长度都不能超出上述限制。如果表中包含 BLOB 或 CLOB 列，则使用 LOB 定位器而不是直接使用 BLOB 或 CLOB 数据计算行长度。有关详细信息，请参见 DB2 文档。

SQL Anywhere 或 UltraLite 数据类型	IBM DB2 主机数据类型	注释
VARBINARY(*n*>=MRL)	BLOB(*n*)	在创建 DB2 表时，您需要注意 DB2 页面大小。DB2 的最大行长度取决于此页面大小：页面大小为 4K 时 MRL 是 4005，8K 时是 8101，16K 时是 16293，32K 时是 32677。表中所有列的长度都不能超出上述限制。如果表中包含 BLOB 或 CLOB 列，则使用 LOB 定位器而不是直接使用 BLOB 或 CLOB 数据计算行长度。有关详细信息，请参见 DB2 文档。
VARBIT(*n*<MRL)	VARCHAR(*n*)	在创建 DB2 表时，您需要注意 DB2 页面大小。DB2 的最大行长度取决于此页面大小：页面大小为 4K 时 MRL 是 4005，8K 时是 8101，16K 时是 16293，32K 时是 32677。表中所有列的长度都不能超出上述限制。如果表中包含 BLOB 或 CLOB 列，则使用 LOB 定位器而不是直接使用 BLOB 或 CLOB 数据计算行长度。有关详细信息，请参见 DB2 文档。
VARBIT(*n*>=MRL)	CLOB(*n*)	在创建 DB2 表时，您需要注意 DB2 页面大小。DB2 的最大行长度取决于此页面大小：页面大小为 4K 时 MRL 是 4005，8K 时是 8101，16K 时是 16293，32K 时是 32677。表中所有列的长度都不能超出上述限制。如果表中包含 BLOB 或 CLOB 列，则使用 LOB 定位器而不是直接使用 BLOB 或 CLOB 数据计算行长度。有关详细信息，请参见 DB2 文档。
VARCHAR(*n*<MRL)	VARCHAR(*n*)	DB2 VARCHAR 最多可以保存 32672 个字节，具体情况取决于页面大小。 DB2 的最大行长度 (MRL) 取决于此页面大小：页面大小为 4K 时 MRL 是 4005，8K 时是 8101，16K 时是 16293，32K 时是 32677。有关详细信息，请参见 DB2 文档。 SQL Anywhere CHAR 与 SQL Anywhere VARCHAR 完全相同。

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

SQL Anywhere 或 UltraLite 数据类型	IBM DB2 主机数据类型	注释
VARCHAR(*n*>=MRL)	CLOB(*n*)	DB2 的最大行长度 (MRL) 取决于此页面大小：页面大小为 4K 时 MRL 是 4005，8K 时是 8101，16K 时是 16293，32K 时是 32677。有关详细信息，请参见 DB2 文档。 SQL Anywhere CHAR 与 SQL Anywhere VARCHAR 完全相同。 DB2 CLOB 值可以比 SQL Anywhere 或 UltraLite 值长。您必须确保下载的值不能过大。
XML	CLOB(*n*)	SQL Anywhere XML 与 SQL Anywhere LONG VARCHAR 完全相同。

映射到 SQL Anywhere 或 UltraLite 远程数据类型

下表列出了 DB2 主机统一数据类型与 SQL Anywhere 和 UltraLite 远程数据类型的映射关系。例如，统一数据库中类型为 INT 的列对应远程数据库中的类型 INTEGER。

IBM DB2 主机数据类型	SQL Anywhere 或 UltraLite 数据类型	注释
BLOB	LONG BINARY	
CHAR(*n*)	VARCHAR(*n*)	SQL Anywhere 中没有与 DB2 CHAR 对等的类型。不应在要同步的统一数据库列中使用 CHAR。如果必须同步 DB2 CHAR 列，请使用 -b 选项运行 MobiLink 服务器。
CHAR(*n*) FOR BIT DATA	BINARY(*n*)	SQL Anywhere 中没有与 DB2 CHAR 对等的类型。不应在要同步的统一数据库列中使用 CHAR。如果必须同步 DB2 CHAR 列，请使用 -b 选项运行 MobiLink 服务器。
CLOB(*n*)	LONG VARCHAR	
DATE	DATE	对于 SQL Anywhere 和 UltraLite，时间值格式必须为 00:00:00。
DBCLOB(*n*)	LONG VARCHAR	数据类型 DBCLOB(*n*) 仅用于双字节字符。SQL Anywhere 没有相对应的数据类型。DB2 字符集为 Unicode 时，DBCLOB(*n*) 等同于 CLOB。

IBM DB2 主机数据类型	SQL Anywhere 或 UltraLite 数据类型	注释
DECIMAL(*p*,*s*)	DECIMAL(*p*,*s*)	
DOUBLE	DOUBLE	如果统一数据库和远程数据库不允许存在完全相同（不精确）的值，DOUBLE 就会导致问题。我们并没有对所有可能的值进行测试，因此，务必要谨慎。为避免出现问题，请不要将这些类型用作主键的一部分。 为了成功地进行同步，SQL Anywhere/ UltraLite DOUBLE 值必须在 DB2 DOUBLE 值域之内。
FLOAT	DOUBLE	如果统一数据库和远程数据库不允许存在完全相同（不精确）的值，FLOAT 就会导致问题。我们并没有对所有可能的值进行测试，因此，务必要谨慎。为避免出现问题，请不要将这些类型用作主键的一部分。 为了成功地进行同步，SQL Anywhere/ UltraLite DOUBLE 值必须在 DB2 DOUBLE 值域之内。
GRAPHIC(*n*)	VARCHAR(2*n*)	DB2 GRAPHIC 不会使用空白进行填充。SQL Anywhere 中没有与 DB2 GRAPHIC 对等的类型。不应在要同步的统一数据库列中使用 GRAPHIC。 数据类型 GRAPHIC 仅用于双字节字符。SQL Anywhere 没有相对应的数据类型。DB2 字符集为 Unicode 时，GRAPHIC 等同于 CHAR。
INT	INTEGER	
NUMERIC(*p*,*s*)	NUMERIC(*p*,*s*)	

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

IBM DB2 主机数据类型	SQL Anywhere 或 UltraLite 数据类型	注释
REAL	DOUBLE	如果统一数据库和远程数据库不允许存在完全相同（不精确）的值，REAL 就会导致问题。我们并没有对所有可能的值进行测试，因此，务必要谨慎。为避免出现问题，请不要将这些类型用作主键的一部分。 在 DB2 主机上，REAL、DOUBLE 和 FLOAT 具有相同的值域，都在 -7.2E +75 到 7.2E+75 之间。最大的负值约为 -5.4E-79，最小的正值约为 5.4E-79。SA/UL REAL 的值域为 -3.402823e+38 到 3.402823e+38，DOUBLE 的值域为 2.22507385850721e-308 到 1.79769313486231e+308。为了成功地进行同步，DB2 主机 REAL 的值必须在 SA/UL REAL 的值域之内，而 SA/UL DOUBLE 的值必须在 DB2 主机 DOUBLE 的值域之内。
ROWID		在 SQL Anywhere 或 UltraLite 中没有相应的数据类型。ROWID 由 DB2 服务器进行维护。该数据类型无法同步。
SMALLINT	SMALLINT	
TIME	TIME	SQL Anywhere TIME 值中的小数秒值在下载时会被截断。要避免出现问题，请不要使用小数秒。
TIMESTAMP	TIMESTAMP	
VARCHAR(*n*)	VARCHAR(*n*)	
VARCHAR(*n*) FOR BIT DATA	VARBINARY(*n*)	
VARGRAPHIC(*n*)	VARCHAR(2*n*)	数据类型 VARGRAPHIC 仅用于双字节字符。SQL Anywhere 没有相对应的数据类型。DB2 字符集为 Unicode 时，VARGRAPHIC 等同于 VARCHAR。

Microsoft SQL Server 数据映射

映射到 Microsoft SQL Server 统一数据类型

下表列出了 SQL Anywhere 和 UltraLite 远程数据类型与 Microsoft SQL Server 统一数据类型的映射关系。例如，远程数据库中类型为 DATE 的列对应统一数据库中的类型 DATETIME。

SQL Anywhere 或 UltraLite 数据类型	Microsoft SQL Server 数据类型	注释
BIGINT	BIGINT	
BINARY(*n*<=8000)	VARBINARY(*n*)	
BINARY(*n*>8000)	VARBINARY(MAX)	
BIT	BIT	
CHAR(*n*<=8000)	VARCHAR(*n*)	
CHAR(*n*>8000)	VARCHAR(MAX)	
DATE	DATE	
DATETIME	DATETIME2	SQL Server DATETIME2 和 TIME 的值精确到 100 纳秒。但是，TIMESTAMP 和 TIME 的值只精确到 1 微秒。为了成功地同步 DATETIME2 和 TIME，建议您将小数秒舍入到 1 微秒。
DECIMAL(*p*=<38,*s*)	DECIMAL(*p*,*s*)	SQL Server DECIMAL/NUMERIC 精度范围是从 1 到 38，因此 *p* 必须小于 39。
DECIMAL(*p*>38,*s*)		SQL Server 中没有相应的数据类型。
DOUBLE	FLOAT(53)	
FLOAT(*p*)	FLOAT(*p*)	
IMAGE	VARBINARY(MAX)	
INTEGER	INT	
LONG BINARY	VARBINARY(MAX)	

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

SQL Anywhere 或 UltraLite 数据类型	Microsoft SQL Server 数据类型	注释
LONG NVARCHAR	NVARCHAR(MAX)	
LONG VARBIT	VARCHAR(MAX)	
LONG VARCHAR	VARCHAR(MAX)	
MONEY	MONEY	
NCHAR(*n*<=4000)	NVARCHAR(*c*)	
NCHAR(*n*>4000)	NVARCHAR(MAX)	
NTEXT	NVARCHAR(MAX)	
NUMERIC(*p*=<38,*s*)	NUMERIC(*p*,*s*)	SQL Server DECIMAL/NUMERIC 精度范围是从 1 到 38，因此 p 必须小于 39。
NUMERIC(*p*>38,*s*)		SQL Server 中没有相应的数据类型。
NVARCHAR(*n*<=4000)	NVARCHAR(*c*)	
NVARCHAR(*n*>4000)	NVARCHAR(MAX)	
REAL	REAL	
SMALLDATETIME	SMALLDATETIME	SQL Anywhere 和 UltraLite SMALLDATETIME 作为 TIMESTAMP 来实现。SQL Server SMALLDATETIME 精确到分钟。29.998 或更低的秒值将向下舍入为最接近的分钟；29.999 或更高的秒值将向上舍入为最接近的分钟。SQL Anywhere 或 UltraLite SMALLDATETIME 精确到微秒。为了成功地进行同步，SQL Anywhere 或 UltraLite SMALLDATETIME 必须舍入到分钟。年份必须在 1900-2078 范围内。
SMALLINT	SMALLINT	
SMALLMONEY	SMALLMONEY	
TEXT	VARCHAR(MAX)	

SQL Anywhere 或 UltraLite 数据类型	Microsoft SQL Server 数据类型	注释
TIME	TIME	SQL Server DATETIME2 和 TIME 的值精确到 100 纳秒。但是，TIMESTAMP 和 TIME 的值只精确到 1 微秒。为了成功地同步 DATETIME2 和 TIME，建议您将小数秒舍入到 1 微秒。
TIMESTAMP	DATETIME2	SQL Server DATETIME2 和 TIME 的值精确到 100 纳秒。但是，TIMESTAMP 和 TIME 的值只精确到 1 微秒。为了成功地同步 DATETIME2 和 TIME，建议您将小数秒舍入到 1 微秒。
TINYINT	TINYINT	对于下载，值必须是非负数。
UNIQUEIDENTIFIER	UNIQUEIDENTIFIER	
UNIQUEIDENTIFIERSTR	UNIQUEIDENTIFIER	
UNSIGNED BIGINT	NUMERIC(20)	对于下载，值必须是非负数。
UNSIGNED INTEGER	NUMERIC(11)	对于下载，值必须是非负数。
UNSIGNED TINYINT	TINYINT	对于下载，值必须是非负数。
UNSIGNED SMALLINT	INT	对于下载，值必须是非负数。
VARBINARY(*n*<=8000)	VARBINARY(*n*)	
VARBINARY(*n*>8000)	VARBINARY(MAX)	
VARBIT(*n*<=8000)	VARCHAR(*n*)	
VARBIT(*n*>8000)	VARCHAR(MAX)	
VARCHAR(*n*<=8000)	VARCHAR(*c*)	
VARCHAR(*n*>8000)	VARCHAR(MAX)	
XML	XML 或 VARCHAR(MAX)	对于 SQL Server 2005，使用 XML。对于其它版本，请使用 VARCHAR(MAX)。

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

映射到 SQL Anywhere 或 UltraLite 远程数据类型

下表列出了 Microsoft SQL Server 统一数据类型与 SQL Anywhere 和 UltraLite 远程数据类型的映射关系。例如，远程数据库中类型为 TEXT 的列对应统一数据库中的类型 LONG VARCHAR。

Microsoft SQL Server 数据类型	SQL Anywhere 或 UltraLite 数据类型	注释
BIGINT	BIGINT	
BINARY(*n*)	BINARY(*n*)	
BIT	BIT	
CHAR(*n*)	VARCHAR(*n*)	Microsoft SQL Server 的 CHAR 列以空白填补。缺省情况下，SQL Anywhere 的 CHAR 列不是用空白添补的，它等同于 VARCHAR 列。所以，应避免在 Microsoft SQL Server 的同步表中使用 CHAR 数据类型。如果必须在 Microsoft SQL Server 统一数据库中使用 CHAR 数据类型，请使用 -b 命令行选项运行 MobiLink 服务器，以帮助解决 SQL Anywhere CHAR 与非 SQL Anywhere CHAR 之间的不同。
DATE	DATE	
DATETIME	TIMESTAMP 或 DATETIME	SQL Server DATETIME 值精确到 1/300 秒。小数秒的最后一位数始终为 0、3 或 6。其它数字将舍入为这三个数之一，即 0 和 1 舍入为 0，2、3 和 4 舍入为 3，5、6、7 和 8 舍入为 6，9 舍入为 10。下载时，SQL Anywhere 会保留 SQL Server 的原始值，但上载时，这些值可能与原始值不完全一致。如果 DATETIME 用于主键，冲突解决可能会失败。为了成功同步 DATETIME，您应将小数秒舍入到 10 毫秒。年份必须在 1753-9999 范围内。
DATETIME2	TIMESTAMP	SQL Server DATETIME2 和 TIME 的值精确到 100 纳秒。但是，TIMESTAMP 和 TIME 的值只精确到 1 微秒。为了成功地同步 DATETIME2 和 TIME，建议您将小数秒舍入到 1 微秒。
DECIMAL(*p*,*s*)	DECIMAL(*p*,*s*)	

Microsoft SQL Server 数据类型	SQL Anywhere 或 UltraLite 数据类型	注释
FLOAT(*p*)	FLOAT(*p*)	
IMAGE	LONG BINARY	
INT	INT	
MONEY	MONEY	
NCHAR(*n*)	NVARCHAR(*c*)	在 UltraLite 中不可用。 SQL Anywhere NCHAR 和非 SQL Anywhere NCHAR 之间不存在对等关系。SQL Anywhere NCHAR 与 NVARCHAR 对等。不应在要同步的统一数据库列中使用 NCHAR。如果必须使用非 SQL Anywhere NCHAR，请用 -b 选项运行 MobiLink 服务器。
NTEXT	LONG NVARCHAR	在 UltraLite 中不可用。
NVARCHAR(*c*)	NVARCHAR(*c*)	在 UltraLite 中不可用。
NVARCHAR(MAX)	LONG NVARCHAR	
NUMERIC(*p*,*s*)	NUMERIC(*p*,*s*)	
REAL	REAL	如果统一数据库和远程数据库不允许存在完全相同（不精确）的值，REAL 就会导致问题。我们并没有对所有可能的值进行测试，因此，务必要谨慎。为避免出现问题，请不要将这些类型用作主键的一部分。
SMALLDATETIME	SMALLDATETIME	SQL Anywhere 和 UltraLite SMALLDATETIME 作为 TIMESTAMP 来实现。SQL Server SMALLDATETIME 精确到分钟。29.998 或更低的秒值将向下舍入为最接近的分钟；29.999 或更高的秒值将向上舍入为最接近的分钟。SQL Anywhere 或 UltraLite SMALLDATETIME 精确到微秒。为了成功地进行同步，SQL Anywhere 或 UltraLite SMALLDATETIME 必须舍入到分钟。年份必须在 1900-2078 范围内。
SMALLINT	SMALLINT	

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

Microsoft SQL Server 数据类型	SQL Anywhere 或 UltraLite 数据类型	注释
SMALLMONEY	SMALLMONEY	
TEXT	LONG VARCHAR	
TIME	TIME	SQL Server DATETIME2 和 TIME 的值精确到 100 纳秒。但是，TIMESTAMP 和 TIME 的值只精确到 1 微秒。为了成功地同步 DATETIME2 和 TIME，建议您将小数秒舍入到 1 微秒。
TIMESTAMP	VARBINARY(8)	在 Microsoft SQL Server 内，TIMESTAMP 是一个二进制计数器，它会随行的每一次更改而相应递增。因此，每个表只能包含一个 TIMESTAMP 列，并且对其进行同步没有任何意义。如果必须对其进行同步，请将其映射到 SQL Anywhere 或 UltraLite 中的 VARBINARY(8) 数据类型。 不能显式地插入或更新时间戳列，因为它由服务器进行维护。当您为含有这种列的表实现上载脚本时，请牢记这一点。
TINYINT	TINYINT	
UNIQUEIDENTIFIER	UNIQUEIDENTIFIER	
VARBINARY(*n*)	VARBINARY(*n*)	
VARBINARY(MAX)	LONG BINARY	
VARCHAR(*n*)	VARCHAR(*n*)	
VARCHAR(MAX)	LONG VARCHAR	
XML	XML	

MySQL 数据映射

映射到 MySQL 统一数据类型

下表列出了 SQL Anywhere 和 UltraLite 远程数据类型与 MySQL 统一数据类型的映射关系。例如，远程数据库中类型为 TEXT 的列对应统一数据库中的类型 LONGTEXT。

SQL Anywhere 或 UltraLite 数据类型	MySQL 数据类型	注意
BIGINT	BIGINT	
BINARY(*n*<=255)	BINARY(*n*)	
BINARY(*n*>255)	BLOB	
BIT	BIT	
CHAR(*n*<=255)	CHAR(*n*)	
CHAR(*n*>255)	TEXT(*n*)	
DATE	DATE	年份的范围必须在 1000 到 9999 之间。
DATETIME	DATETIME	MySQL DATETIME 数据类型不支持小数秒。年份的范围必须在 1000 到 9999 之间。
DECIMAL(*p*<=65,*s*<=30)	DECIMAL(*p*,*s*)	
DECIMAL(*p*>65,*s*>30)		如果精度大于 65 或小数位数大于 30，则 MySQL 中将没有对应的数据类型。
DOUBLE	DOUBLE	
FLOAT	FLOAT	
IMAGE	LONGBLOB	
INTEGER	INTEGER	
LONG BINARY	LONGBLOB	
LONG NVARCHAR	LONGTEXT CHARACTER SET UTF8	
LONG VARBIT	LONGTEXT	
LONG VARCHAR	LONGTEXT	

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

SQL Anywhere 或 UltraLite 数据类型	MySQL 数据类型	注意
MONEY	NUMERIC(19,4)	
NCHAR(*n*<=255)	CHAR(*n*) CHARACTER SET UTF8	
NCHAR(*n*>255)	TEXT CHARACTER SET UTF8	
NTEXT	LONGTEXT CHARACTER SET UTF8	
NUMERIC(*p*<=65,*s*<=30)	DECIMAL(*p*,*s*)	
NUMERIC(*p*>65,*s*>30)		MySQL 中没有对应的数据类型。
NVARCHAR(*n*)	VARCHAR(*n*) CHARACTER SET UTF8	
REAL	REAL	
SMALLDATETIME	DATETIME	MySQL DATETIME 数据类型不支持小数秒。年份的范围必须在 1000 到 9999 之间。
SMALLINT	SMALLINT	
SMALLMONEY	NUMERIC(10,4)	
TEXT	LONGTEXT	
TIME	TIME	MySQL TIME 数据类型不支持小数秒。
TIMESTAMP	DATETIME	MySQL DATETIME 数据类型不支持小数秒。年份的范围必须在 1000 到 9999 之间。
TINYINT	TINYINT UNSIGNED	在 SQL Anywhere 和 UltraLite 中，TINYINT 始终是无符号的。
UNIQUEIDENTIFIER	CHAR(36)	
UNIQUEIDENTIFIERSTR	CHAR(36)	
VARBINARY(*n*)	VARCHAR(*n*)	

Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

SQL Anywhere 或 UltraLite 数据类型	MySQL 数据类型	注意
VARBIT(*n*<=8000)	VARCHAR(*n*)	
VARBIT(*n*>8000)	TEXT	
VARCHAR(*n*)	VARCHAR(*n*)	
XML	LONGTEXT	

映射到 SQL Anywhere 或 UltraLite 远程数据类型

下表列出了 MySQL 统一数据类型与 SQL Anywhere 和 UltraLite 远程数据类型的映射关系。例如，统一数据库中类型为 BOOL 的列对应远程数据库中的类型 BIT。

MySQL 数据类型	SQL Anywhere 或 UltraLite 数据类型	注意
BIGINT	BIGINT	
BINARY(*n*)	BINARY(*n*)	
BIT(1)	BIT	
BIT(*n*>1)	UNSIGNED BIGINT	
BLOB(*n*<=32767)	VARBINARY(*n*)	
BLOB(*n*>32767)	IMAGE	
BOOL	BIT	
CHAR(*n*)	CHAR(*n*)	
DATE	DATE	年份的范围必须在 1000 到 9999 之间。
DATETIME	DATETIME	MySQL DATETIME 数据类型不支持小数秒。年份的范围必须在 1000 到 9999 之间。
DOUBLE	DOUBLE	
DECIMAL	DECIMAL	
ENUM		在 SQL Anywhere 或 UltraLite 中没有相应的数据类型。
GEOMETRY		在 SQL Anywhere 或 UltraLite 中没有相应的数据类型。

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

MySQL 数据类型	SQL Anywhere 或 UltraLite 数据类型	注意
INTEGER	INTEGER	
LINESTRING		在 SQL Anywhere 或 UltraLite 中没有相应的数据类型。
LONGBLOB	IMAGE	
LONGTEXT	TEXT	
MEDIUMBLOB	IMAGE	
MEDIUMINT	INTEGER	
MEDIUMTEXT	TEXT	
MULTILINESTRING		在 SQL Anywhere 或 UltraLite 中没有相应的数据类型。
MULTIPOINT		在 SQL Anywhere 或 UltraLite 中没有相应的数据类型。
MULTIPOLYGON		在 SQL Anywhere 或 UltraLite 中没有相应的数据类型。
NCHAR	NCHAR	在 UltraLite 中不可用。
NUMERIC	NUMERIC	
NVARCHAR	NVARCHAR	在 UltraLite 中不可用。
POINT		在 SQL Anywhere 或 UltraLite 中没有相应的数据类型。
POLYGON		在 SQL Anywhere 或 UltraLite 中没有相应的数据类型。
REAL	REAL	
SET		在 SQL Anywhere 或 UltraLite 中没有相应的数据类型。
SMALLINT	SMALLINT	
TEXT(*n*<=32767)	VARCHAR(*n*)	
TEXT(*n*>32767)	TEXT	

MySQL 数据类型	SQL Anywhere 或 UltraLite 数据类型	注意
TIME	TIME	MySQL TIME 数据类型不支持小数秒。在 MySQL 中，TIME 的范围是 '-838:59:59' 到 '838:59:59'。在 SQL Anywhere 或 UltraLite 中，TIME 的范围是 '00:00:00.000000' 到 '23:59:59:999999'。
TIMESTAMP	TIMESTAMP	MySQL DATETIME 数据类型不支持小数秒。年份的范围必须在 1000 到 9999 之间。虽然 MySQL 提供对 TIMESTAMP 列的自动初始化和更新，但 SQL Anywhere 和 UltraLite 只提供自动初始化。
TINYBLOB	VARBINARY	
TINYINT	SMALLINT	在 SQL Anywhere 和 UltraLite 中，TINYINT 始终是无符号的。必须是正值。
TINYINT UNSIGNED	TINYINT	在 SQL Anywhere 和 UltraLite 中，TINYINT 始终是无符号的。
TINYTEXT	VARCHAR	
VARBINARY(*n*<=32767)	VARBINARY(*n*)	
VARBINARY(*n*>32767)	IMAGE	
VARCHAR(*n*<=32767)	VARCHAR(*n*)	
VARCHAR(*n*>32767)	TEXT	
YEAR[(2|4)]	INTEGER	SQL Anywhere 和 UltraLite 不支持 YEAR 数据类型。需要将 YEAR 映射到远程数据库中的 INTEGER。INTEGER 值必须在 1000 到 9999 之间。

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

Oracle 数据映射

映射到 Oracle 统一数据类型

下表列出了 SQL Anywhere 和 UltraLite 远程数据类型与 Oracle 统一数据类型的映射关系。例如，远程数据库中类型为 BIT 的列对应统一数据库中的类型 NUMBER。

SQL Anywhere 或 UltraLite 数据类型	Oracle 数据类型	注释
BIGINT	NUMBER(20)	
BINARY(*n*<=2000)	RAW(*n*)	
BINARY(*n*>2000)	BLOB	Oracle 值可以长于 SQL Anywhere 或 UltraLite 值，因此下载时请确保值不要太大。
BIT	NUMBER(1)	
CHAR(*n*<=4000)	VARCHAR2(*n* byte)	Oracle VARCHAR2 允许您指定最大字节或字符数。VARCHAR2 数据的最大长度为 4000 字节。如果指定字符数，请确保数据的最大长度不超过 4000 字节。
CHAR(*n*>4000)	CLOB	Oracle 值可以长于 SQL Anywhere 或 UltraLite 值，因此下载时请确保值不要太大。
DATE	DATE[2] 或 TIMESTAMP	使用没有小数秒的 Oracle DATE 数据类型时，将不会保留 SQL Anywhere 或 UltraLite 小数秒。要避免出现问题，请不要使用小数秒。年份必须在 1-9999 范围内。 使用 Interactive SQL 实用程序时，请先关闭 Return_date_time_as_string 选项，然后再执行 SQL 语句。
DATETIME	DATE[2] 或 TIMESTAMP	使用没有小数秒的 Oracle DATE 数据类型时，将不会保留 SQL Anywhere 或 UltraLite 小数秒。要避免出现问题，请不要使用小数秒。年份必须在 1-9999 范围内。 使用 Interactive SQL 实用程序时，请先关闭 Return_date_time_as_string 选项，然后再执行 SQL 语句。

SQL Anywhere 或 UltraLite 数据类型	Oracle 数据类型	注释
DECIMAL(*p*<=38,*s*)	NUMBER(*p*, 0<=*s*<=38)	在 SQL Anywhere DECIMAL 中，p 介于 1 到 127 之间，s 始终小于或等于 p。在 Oracle NUMBER 中，p 的范围从 1 到 38，s 的范围从 -84 到 127。为了进行同步，Oracle NUMBER 的数值范围必须限制在 0 和 38 之间。
DECIMAL(*p*>38,*s*)		Oracle 中没有相应的数据类型。
DOUBLE	DOUBLE PRECISION 或 BINARY_DOUBLE[1]	Oracle 10g BINARY_FLOAT 和 BINARY_DOUBLE 的特殊值 INF、-INF 和 NAN 不能与 SQL Anywhere 或 UltraLite 同步。
FLOAT(*p*)	FLOAT(*p*)	
IMAGE	BLOB	Oracle 值可以长于 SQL Anywhere 或 UltraLite 值，因此下载时请确保值不要太大。
INTEGER	INT	
LONG BINARY	BLOB	Oracle 值可以长于 SQL Anywhere 或 UltraLite 值，因此下载时请确保值不要太大。
LONG NVARCHAR	NCLOB	Oracle CLOB 和 NCLOB 最多可以保存 4G 数据。SQL Anywhere LONG VARCHAR 和 LONG NVARCHAR 只能保存最多 2G 数据。 Oracle 值可以长于 SQL Anywhere 或 UltraLite 值，因此下载时请确保值不要太大。
LONG VARBIT	CLOB	Oracle 值可以长于 SQL Anywhere 或 UltraLite 值，因此下载时请确保值不要太大。

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

SQL Anywhere 或 UltraLite 数据类型	Oracle 数据类型	注释
LONG VARCHAR	CLOB	Oracle CLOB 和 NCLOB 最多可以保存 4G 数据。SQL Anywhere LONG VARCHAR 和 LONG NVARCHAR 只能保存最多 2G 数据。 Oracle 值可以长于 SQL Anywhere 或 UltraLite 值，因此下载时请确保值不要太大。
MONEY	NUMBER(19,4)	
NCHAR(*c*)	NVARCHAR2(*c* char) 或 NCLOB	SQL Anywhere NCHAR 和 Oracle NVARCHAR2 的大小表示 Unicode 字符的最大数目。Oracle NVARCHAR2 的数据长度不能超过 4000 字节。很难通过字符大小计算出最大字节长度。通常，如果字符大小超过 1000，则映射到 NCLOB，否则映射到 NVARCHAR2。
NTEXT	NCLOB	Oracle NCLOB 最多可以保存 4G 数据。SQL Anywhere NTEXT（或 LONG NVARCHAR）最多只能保存 2G 数据。 Oracle 值可以长于 SQL Anywhere 或 UltraLite 值，因此下载时请确保值不要太大。
NUMERIC(*p*<=38,*s*)	NUMBER(*p*, 0<=*s*<=38)	在 SQL Anywhere NUMERIC 中，p 介于 1 到 127 之间，s 始终小于或等于 p。在 Oracle NUMBER 中，p 的范围从 1 到 38，s 的范围从 -84 到 127。为了进行同步，Oracle NUMBER 的数值范围必须限制在 0 和 38 之间。
NUMERIC(*p*>38,*s*)		Oracle 中没有相应的数据类型。
NVARCHAR	NVARCHAR2(*c* char) 或 NCLOB	SQL Anywhere NCHAR 和 Oracle NVARCHAR2 的大小表示 Unicode 字符的最大数目。Oracle NVARCHAR2 的数据长度不能超过 4000 字节。很难通过字符大小计算出最大字节长度。通常，如果字符大小超过 1000，则映射到 NCLOB，否则映射到 NVARCHAR2。

SQL Anywhere 或 UltraLite 数据类型	Oracle 数据类型	注释
REAL	REAL 或 BINARY_FLOAT[1]	Oracle 10g BINARY_FLOAT 和 BINARY_DOUBLE 的特殊值 INF、-INF 和 NAN 不能与 SQL Anywhere 或 UltraLite 同步。
SMALLDATETIME	DATE[2] 或 TIMESTAMP	使用没有小数秒的 Oracle DATE 数据类型时，将不会保留 SQL Anywhere 或 UltraLite 小数秒。要避免出现问题，请不要使用小数秒。年份必须在 1-9999 范围内。
SMALLINT	NUMBER(5)	
SMALLMONEY	NUMBER(10,4)	
TEXT	CLOB	Oracle CLOB 最多可以保存 4G 数据。SQL Anywhere TEXT（或 LONG VARCHAR）最多只能保存 2G 数据。 Oracle 值可以长于 SQL Anywhere 或 UltraLite 值，因此下载时请确保值不要太大。
TIME	DATE[2] 或 TIMESTAMP	使用没有小数秒的 Oracle DATE 数据类型时，将不会保留 SQL Anywhere 或 UltraLite 小数秒。要避免出现问题，请不要使用小数秒。 使用 Interactive SQL 实用程序时，请先关闭 Return_date_time_as_string 选项，然后再执行 SQL 语句。
TIMESTAMP	DATE[2] 或 TIMESTAMP	使用没有小数秒的 Oracle DATE 数据类型时，将不会保留 SQL Anywhere 或 UltraLite 小数秒。要避免出现问题，请不要使用小数秒。年份必须在 1-9999 范围内。 使用 Interactive SQL 实用程序时，请先关闭 Return_date_time_as_string 选项，然后再执行 SQL 语句。
TINYINT	NUMBER(3)	对于下载，Oracle 值必须是非负数。
UNSIGNED BIGINT	NUMBER(20)	对于下载，Oracle 值必须是非负数。

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

SQL Anywhere 或 UltraLite 数据类型	Oracle 数据类型	注释
UNSIGNED INTEGER	NUMBER(11)	对于下载，Oracle 值必须是非负数。
UNSIGNED SMALLINT	NUMBER(5)	对于下载，Oracle 值必须是非负数。
UNSIGNED TINYINT	NUMBER(3)	对于下载，Oracle 值必须是非负数。
UNIQUEIDENTIFIER	CHAR(36)	
UNIQUEIDENTIFIERSTR	CHAR(36)	建议不要将 UNIQUEIDENTIFIERSTR 用于 Oracle。请改用 UNIQUEIDENTIFIER。
VARBINARY(*n*<=2000)	RAW(*n*)	
VARBINARY(*n*>2000)	BLOB	Oracle 值可以长于 SQL Anywhere 或 UltraLite 值，因此下载时请确保值不要太大。
VARBIT(*n*<=4000)	VARCHAR2(*n* byte)	
VARBIT(*n*>4000)	CLOB	Oracle 值可以长于 SQL Anywhere 或 UltraLite 值，因此下载时请确保值不要太大。
VARCHAR(*n*<=4000)	VARCHAR2(*n* byte)	Oracle VARCHAR2 允许您指定最大字节或字符数。VARCHAR2 数据的最大长度为 4000 字节。如果指定字符数，请确保数据的最大长度不超过 4000 字节。
VARCHAR(*n*>4000)	CLOB	Oracle 值可以长于 SQL Anywhere 或 UltraLite 值，因此下载时请确保值不要太大。
XML	CLOB	Oracle CLOB 和 NCLOB 最多可以保存 4G 数据。SQL Anywhere XML 最多只能保存 2G 数据。 Oracle 值可以长于 SQL Anywhere 或 UltraLite 值，因此下载时请确保值不要太大。

[1] 仅适用于 Oracle 版本 10g 或更高版本。

[2] 仅适用于 Oracle 版本 8i。

注意
Oracle 8、8i 和 9i 中不再支持 LONG 数据类型。

要正确同步 Oracle LONG 数据类型，必须在 iAnywhere Solutions Oracle ODBC 驱动程序的 [**ODBC 数据源配置**] 窗口中选中 [**Oracle Force Retrieval Of Long Columns**] 选项。

映射到 SQL Anywhere 或 UltraLite 远程数据类型

下表列出了 Oracle 的统一数据类型与 SQL Anywhere 或 UltraLite 远程数据类型的映射关系。例如，统一数据库中的类型为 LONG 的列对应远程数据库中的类型 LONG VARCHAR。

Oracle 数据类型	SQL Anywhere 或 UltraLite 数据类型	注释
BFILE	LONG BINARY	仅下载。 Oracle 值可以长于 SQL Anywhere 或 UltraLite 值，因此下载时请确保值不要太大。
BINARY_DOUBLE	DOUBLE	BINARY_FLOAT 的特殊值 INF、-INF 和 NAN 不能与 SQL Anywhere 或 UltraLite 同步。FLOAT 和 DOUBLE 的精度在 Oracle 中与在 SQL Anywhere 和 UltraLite 中不同。数据的值可能依据精度而更改。
BINARY_FLOAT	REAL	BINARY_FLOAT 的特殊值 INF、-INF 和 NAN 不能与 SQL Anywhere 或 UltraLite 同步。FLOAT 和 DOUBLE 的精度在 Oracle 中与在 SQL Anywhere 和 UltraLite 中不同。数据的值可能依据精度而更改。
BLOB	LONG BINARY	Oracle 值可以长于 SQL Anywhere 或 UltraLite 值，因此下载时请确保值不要太大。
CHAR(*n* byte)	VARCHAR(*n*)	SQL Anywhere CHAR 和 Oracle CHAR 之间不存在对等关系。SQL Anywhere CHAR 与 VARCHAR 对等。不应在要同步的统一数据库列中使用 CHAR/NCHAR。如果必须使用非 SQL Anywhere CHAR，请用 -b 选项运行 MobiLink 服务器。 SQL Anywhere 或 UltraLite 值可以长于 Oracle 值，因此上载时请确保值不要太大。
CLOB	LONG VARCHAR	Oracle 值可以长于 SQL Anywhere 或 UltraLite 值，因此下载时请确保值不要太大。

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

Oracle 数据类型	SQL Anywhere 或 UltraLite 数据类型	注释
DATE	TIMESTAMP	年份必须在 1-9999 范围内。
INTERVAL YEAR(*year_precision*) TO MONTH		在 SQL Anywhere 或 UltraLite 中没有相应的数据类型。
INTERVAL DAY(*day_precision*) TO SECOND(*p*)		在 SQL Anywhere 或 UltraLite 中没有相应的数据类型。
LONG	LONG VARCHAR	
LONG RAW	LONG BINARY	
NCHAR(*c* char)	NVARCHAR(*c*)	SQL Anywhere NCHAR 和 Oracle NCHAR 之间不存在对等关系。SQL Anywhere NCHAR 与 NVARCHAR 对等。不应在要同步的统一数据库列中使用 NCHAR。如果必须使用非 SQL Anywhere NCHAR，请用 -b 选项运行 MobiLink 服务器。 SQL Anywhere 或 UltraLite 值可以长于 Oracle 值，因此上载时请确保值不要太大。
NCLOB	LONG NVARCHAR	在 UltraLite 中不可用。 Oracle 值可以长于 SQL Anywhere 或 UltraLite 值，因此下载时请确保值不要太大。
NUMBER(*p*,*s*)	NUMBER(*p*,*s*)	在 SQL Anywhere NUMBER 中，*p* 在 1 到 127 之间，而 *s* 始终小于或等于 *p*。在 Oracle NUMBER 中，*p* 的范围是从 1 到 38，而 *s* 的范围是从 -84 到 127。要进行同步，Oracle NUMBER 的数值范围必须在 0 到 38 之间。
NVARCHAR2(*c* char)	NVARCHAR(*c*)	在 UltraLite 中不可用。 SQL Anywhere 或 UltraLite 值可以长于 Oracle 值，因此上载时请确保值不要太大。
RAW	BINARY	SQL Anywhere 或 UltraLite 值可以长于 Oracle 值，因此上载时请确保值不要太大。

Oracle 数据类型	SQL Anywhere 或 UltraLite 数据类型	注释
ROWID	VARCHAR(64)	UROWID 和 ROWID 是只读的，因此不可能进行同步。
TIMESTAMP(*p*<=6)	TIMESTAMP	p<6 时，您可能需要确保 SQL Anywhere 或 UltraLite 值具有相同的精度。否则，冲突检测可能会失败，也可能出现重复行。年份必须在 1-9999 范围内。
TIMESTAMP(*p*>6)		在 SQL Anywhere 或 UltraLite 中没有相应的数据类型。
TIMESTAMP(*p*) WITH LOCAL TIME ZONE		在 SQL Anywhere 或 UltraLite 中没有相应的数据类型。
TIMESTAMP(*p*) WITH TIME ZONE		在 SQL Anywhere 或 UltraLite 中没有相应的数据类型。
UROWID	VARCHAR(64)	UROWID 和 ROWID 是只读的，因此不可能进行同步。
VARCHAR2(*n* byte)	VARCHAR(*n*)	SQL Anywhere 或 UltraLite 值可以长于 Oracle 值，因此上载时请确保值不要太大。

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

字符集注意事项

目录

字符集注意事项 740

字符集注意事项

文本的每个字符用一个或多个字节表示。从字符到二进制码的映射关系称为**字符集编码**。一些字母表比较小的语言的字符集使用单字节表示，如欧洲语言。而其它字符集，如 Unicode，则使用双字节表示。由于每个字符使用双倍的存储空间，双字节的字符集可以表示更多数量的字符。

当使用一种字符集的文本必须转换为另一种字符集时，可能出现转换错误或数据丢失情况。并不是所有的字符都可以用任何字符集表示。尤其是，由于可用编码的数量限制，单字节的字符集比多字节系统表示的字符要少很多。

如果 MobiLink 远程数据库的字符集与统一数据库的相同，可避免字符转换问题。

文本通常需要通过排序建立索引，并准备有序的结果集，如目录列表。**排序顺序**标识字符的顺序。例如，排序顺序通常规定字母 "a" 在字母 "b" 之前，字母 "b" 在字母 "c" 之前。

每个数据库都有一个**归类序列**。您可以在创建数据库时设置归类序列，不过设置的方式可能在不同数据库系统之间有所不同。归类序列定义该数据库的字符集与排序顺序。

提示
只要有可能，您就要将远程数据库的归类序列定义为与统一数据库的归类序列相同。这种安排可以降低错误转换的可能性。

另请参见

- SQL Anywhere 客户端： “国际语言和字符集” 《SQL Anywhere 服务器 - 数据库管理》
- UltraLite 客户端： “UltraLite 字符集”一节 《UltraLite - 数据库管理和参考》
- 特定于 RDBMS 的信息： “MobiLink 统一数据库”第 3 页

同步过程中的字符集转换

在同步期间，可能需要将字符从一种字符集转换到另一种字符集。在远程应用程序和统一数据库之间传递字符时，将进行以下转换。

上载中的字符集转换

MobiLink 客户端采用远程数据库的字符集将数据发送给 MobiLink 服务器。

1. MobiLink 服务器通过 Unicode ODBC API 与统一数据库进行通信。为此，MobiLink 服务器将收到的、来自远程数据库的所有字符转换为 Unicode，并将 Unicode 发送到 ODBC 驱动程序。
2. 如果有必要，统一数据库服务器的 ODBC 驱动程序将字符从 Unicode 转换为统一数据库的字符集。该转换完全由统一数据库系统中的 ODBC 驱动程序控制。因此，两种不同数据库系统的行为可能不同，尤其是不同生产商生产的系统。MobiLink 同步适用于多种数据库系统。有关详细信息，请查看特定的统一服务器和 ODBC 驱动程序的文档。

下载中的字符集转换

1. 统一数据库系统的 ODBC 驱动程序将接收按统一数据库字符集编码的字符。它将这些字符转换为 Unicode，并通过 Unicode API 发送给 MobiLink 服务器。该转换完全由统一数据库系统中的 ODBC 驱动程序控制。有关详细信息，请查看特定的统一服务器和 ODBC 驱动程序的文档。

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

2. MobiLink 服务器通过 Unicode ODBC API 接收字符。如果远程数据库使用不同的字符集，MobiLink 服务器将在下载字符之前将其转换。

示例

- Windows Mobile 设备上的 UltraLite 应用程序使用 Unicode 字符集。

  当您对 Windows Mobile 应用程序进行同步时，MobiLink 服务器中不会发生字符转换。服务器发现从应用程序发送过来的数据已经是 Unicode 编码，并将其直接传送给 ODBC 驱动程序。类似地，在下载数据时，不需要进行字符集转换。

- 在除 Windows Mobile 之外的其它平台上，所有的 SQL Anywhere 数据库和所有的 UltraLite 应用程序都将使用由远程数据库的归类序列确定的字符集。

  当您对远程数据库进行同步时，MobiLink 服务器在远程数据库的字符集和 Unicode 之间进行字符集转换。

控制 ODBC 驱动程序字符集转换

因为多数统一数据库不太可能使用 Unicode，所以应该了解统一数据库系统的 ODBC 驱动程序如何与 Unicode 字符集进行数据转换。有些 ODBC 驱动程序采用运行 MobiLink 的计算机的语言设置来确定应使用的字符集。在这些情况下，最好能使运行 MobiLink 服务器的计算机的语言和代码页设置与统一数据库的语言和代码页设置相匹配。

其它 ODBC 驱动程序（如 Sybase Adaptive Server Enterprise 的驱动程序）允许各个连接使用特定的字符集。为避免发生转换错误，MobiLink 使用的字符集的设置应该与统一数据库中的相匹配。

有关字符集如何在统一数据库服务器的 ODBC 驱动程序中进行转换的详细说明，请查阅该产品的 ODBC 驱动程序文档。

用于 MobiLink 的 iAnywhere Solutions ODBC 驱动程序

目录

MobiLink 支持的 ODBC 驱动程序 744
iAnywhere Solutions Oracle 驱动程序 745

MobiLink 支持的 ODBC 驱动程序

MobiLink 服务器可使用多种统一数据库及 ODBC 驱动程序，如下表所示。有些驱动程序，尽管可以与 MobiLink 兼容使用，但是可能在使用时有些功能上的限制。

有关所支持版本的详细信息，请参见 http://www.sybase.com/detail?id=1062617。

数据库	ODBC 驱动程序
SQL Anywhere 11	SQL Anywhere 11[1]
Oracle 10g 或 11g	iAnywhere Solutions 11 - Oracle[1]
Microsoft SQL Server	Microsoft SQL Server ODBC 驱动程序 [2]
Sybase Adaptive Server Enterprise 12.5.1 或更高版本	Sybase Adaptive Server Enterprise 驱动程序 [2]
用于 Windows、Linux 和 Unix 的 IBM DB2 LUW 8.1 或 8.2	IBM DB2 8.2 CLI 驱动程序 [2]
用于 Windows、Linux 和 Unix 的 IBM DB2 LUW 9.x	IBM DB2 9.1 CLI 驱动程序 [2]
用于 Z/OS 的 IBM DB2 主机 8.1	IBM DB2 8.2 CLI 驱动程序 [2]
用于 Z/OS 的 IBM DB2 主机 9.1	IBM DB2 9.1 CLI 驱动程序 [2]
MySQL 5.1	MySQL ODBC 驱动程序 5.1[2]

[1] 随 SQL Anywhere 版本 11 一起提供。请参见 Recommended ODBC Drivers for MobiLink。

[2] 不随 SQL Anywhere 版本 11 一起提供。有关安装及配置说明，请参见 Recommended ODBC Drivers for MobiLink。

Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

iAnywhere Solutions Oracle 驱动程序

iAnywhere Solutions 11 - Oracle ODBC 驱动程序是为配合 iAnywhere 软件使用而自定义定制的。该驱动程序不使用第三方软件。

如果将 Oracle 与 MobiLink 或 OMNI 配合使用，则必须在 Oracle 驱动程序所在的同一台计算机上安装 Oracle 客户端。

该 Oracle 驱动程序可以使用 ODBC 管理器、.odbc.ini 文件（在 Unix 中）或 dbdsn 实用程序来配置。

下表提供了 Oracle 驱动程序的配置选项。

Windows ODBC 管理器	dbdsn 命令行或 .odbc.ini 文件的配置	说明
数据源名	对于 dbdsn，使用 -w 选项。	标识数据源的名称。
用户 ID	**UserID** 在 dbdsn 中，在连接字符串中设置此选项。	连接到 Oracle 数据库时应用程序使用的缺省登录 ID。如果您将此字段留空，当您连接时会提示您输入该信息。
口令	**Password** 在 dbdsn 中，在连接字符串中设置此选项。	应用程序用于连接到 Oracle 数据库的口令。如果您将此字段留空，当您连接时会提示您输入该信息。
SID	**SID**	TNS 服务名，存储在 Oracle 安装目录下的 *network/admin/tnsnames.ora* 中。
启用 Microsoft 分布式事务	对于 dbdsn，在连接字符串中使用 **enableMSDIC** 选项。 不支持 *.odbc.ini*。	如果想征用 "Microsoft 分布式事务协调器" 中的事务，请选中此复选框。当选中此复选框后，Oracle ODBC 驱动程序会需要一个 Oracle 二进制文件（如果是 Oracle 9i 客户端，则为 *oramts.dll*；如果是 Oracle 10g，则为 *oramts10.dll*）。
加密口令	对于 dbdsn，使用 -pe 选项。 不支持 *.odbc.ini*。	如果希望口令以加密形式存储在数据源中，则选中此复选框。
过程返回结果	**ProcResults** 在 dbdsn 中，在连接字符串中设置此选项。	如果您的存储过程可以返回结果，则选择此字段。缺省为过程不返回结果（不选择）。如果 download_cursor 或 download_delete_cursor 脚本是存储过程调用，则将其设置为 YES。

Windows ODBC 管理器	dbdsn 命令行或 .odbc.ini 文件的配置	说明
数组大小	**ArraySize** 在 dbdsn 中，在连接字符串中设置此选项。	基于每个语句，用于预读取行的字节数组大小（以字节为单位）。缺省值为 60000。增加此值可显著提高读取性能（例如，在 MobiLink 服务器下载期间），但代价是需要额外的内存分配。

Windows 配置

♦ **在 Windows 中为 Oracle 驱动程序创建一个 DSN**

1. 打开 ODBC 管理器：
   - 选择 [**开始**] » [**程序**] » [**SQL Anywhere 11**] » [**ODBC 管理器**]。

   即会出现 [**ODBC 数据源管理器**]。
2. 单击 [**添加**]。
3. 选择 [**iAnywhere Solutions 11 - Oracle**]。
4. 指定您需要的配置选项。上面已介绍了该字段。
5. 单击 [**测试连接**]，然后单击 [**确定**]。

Unix 配置

在 Unix 上，如果在 ODBC 系统信息文件（通常称为 *.odbc.ini*）中设置驱动程序，则此驱动程序部分应如下所示（已为各字段输入了适当值）：

```
[sample_dsn_using_the_ias_odbc_driver_for_oracle]
Driver=full-path/libdboraodbc10_r.so
UserID=user-id
Password=password
SID=TNS-service-name
ProcResults=[yes|no]
ArraySize=bytes
```

有关各字段的说明，请参见以上信息。

DBDSN 配置

要用 dbdsn 实用程序创建一个 Oracle DSN，请使用下面的语法：

dbdsn -w *data-source-name* **-or -c** *configuration-options*

configuration-options 如上所述。

例如：

```
dbdsn -w MyOracleDSN -or -pe -c
Userid=dba;Password=sql;SID=abcd;ArraySize=100000;ProcResults=y;enableMSDIC=n
```

请参见“数据源实用程序 (dbdsn)”一节 《SQL Anywhere 服务器 - 数据库管理》。

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

另请参见

- Recommended ODBC Drivers for MobiLink
- “数据源实用程序 (dbdsn)”一节 《SQL Anywhere 服务器 - 数据库管理》

部署 MobiLink 应用程序

目录

MobiLink 部署简介 750
部署 MobiLink 服务器 751
部署 SQL Anywhere MobiLink 客户端 763
部署 UltraLite MobiLink 客户端 765
部署 QAnywhere 应用程序 766

MobiLink 部署简介

部署 MobiLink 应用程序包括以下活动：

- 将 MobiLink 服务器部署到生产设置。
- 根据需要，部署重定向器。
- 部署任何 SQL Anywhere MobiLink 客户端。
- 部署任何 UltraLite MobiLink 客户端。

本章介绍为进行以下每项部署而需要在应用程序的安装程序中包含的文件。

帮助您在 Windows 上进行部署的 [**部署同步模型向导**]。请参见“使用 [部署向导]”一节 《SQL Anywhere 服务器 - 编程》。

检查许可协议
重新分发文件受到许可协议的制约。本文中的任何声明都不能替代许可协议中的任何内容。在考虑部署之前，请检查许可协议。

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

部署 MobiLink 服务器

将 MobiLink 服务器部署到生产环境的最简单方式是将 SQL Anywhere 的许可副本安装到生产计算机。

不过，如果在单独的安装程序中重新分发 MobiLink 服务器，则可以仅包含这些文件的一个子集。在这种情况下，您需要在安装中包括以下文件。

注意

- 重新分发前请在一台干净的计算机上测试。
- 除示例外，必须将文件安装到 SQL Anywhere 安装目录。
- 除非另行说明，否则这些文件应位于同一目录中。
- 给定位置时，文件必须复制到具有相同名称的目录中。
- 在 Unix 上，为了使系统能够定位 SQL Anywhere 应用程序和库，必须设置环境变量。建议使用适合您的 shell 的文件作为设置所需环境变量的模板，即 *sa_config.sh* 或 *sa_config.csh* 文件（在 32 位环境中它们位于目录 *install-dir/bin32* 中，而在 64 位环境中则位于目录 *install-dir/bin64* 中）。部分由 sa_config 文件设置的环境变量包括 PATH、LD_LIBRARY_PATH、SQLANY11 和 SQLANYSAMP11。
- 在 Windows 上，为了使系统能够定位 SQL Anywhere 应用程序和库，必须设置 PATH 环境变量。请检查 PATH 变量，以确保在 32 位环境下它包括 *install-dir/bin32*，在 64 位环境下它包括 *install-dir/bin64*。如果两个条目都存在，请删除不适用于您的环境的路径。
- 要使用 Java 同步逻辑和图形管理工具（Sybase Central 和 MobiLink 监控器），必须安装 JRE 1.6.0。
- 要部署 Sybase Central，请参见“部署管理工具”一节 《SQL Anywhere 服务器 - 编程》。
- 有一个用于 Windows 的部署向导。请参见“使用 [部署向导]”一节 《SQL Anywhere 服务器 - 编程》。

Windows 32 位应用程序

所有目录都相对于 *install-dir*。有关 64 位 Windows 环境文件结构的详细信息，请参见“Windows 64 位应用程序”一节第 754 页。

说明	Windows 文件
MobiLink 服务器	● *bin32\mlodbc11.dll* ● *bin32\mlsrv11.exe* ● *bin32\mlsrv11.lic* ● *bin32\mlsql11.dll* ● *bin32\dbicu11.dll* ● *bin32\dbicudt11.dll*
语言库	● *bin32\dblgen11.dll*[1]

说明	Windows 文件
同步流库（支持版本 8 和版本 9 客户端）	● *bin32\mlhttp11.dll* ● *bin32\mlsock11.dll*
Java 同步逻辑	● *java\activation.jar*[2] ● *java\imap.jar*[2] ● *java\jodbc.jar* ● *java\log4j.jar*[2] ● *java\mailapi.jar*[2] ● *java\mlscript.jar* ● *java\mlsupport.jar* ● *java\pop3.jar*[2] ● *java\smtp.jar*[2] ● *bin32\mljava11.dll* ● *bin32\dbjodbc11.dll* ● *bin32\mljodbc11.dll*
.NET 同步逻辑	● *MobiLink\setup\dnet\mlDomConfig.xml* ● *bin32\mldnet11.dll* ● *bin32\dnetodbc11.dll* ● *Assembly\v2\iAnywhere.MobiLink.dll* ● *Assembly\v2\iAnywhere.MobiLink.Script.dll* ● *Assembly\v2\iAnywhere.MobiLink.Script.xml* ● *bin32\mlDomConfig.xsd*
版本 10 和版本 11 客户端 (mlsrv11 -x) 的安全性组件 [3]	● *bin32\mlecc_tls11.dll* ● *bin32\mlrsa_tls11.dll* ● *bin32\mlrsa_tls_fips11.dll* ● *bin32\sbgse2.dll*
版本 8 和版本 9 客户端 (mlsrv11 -xo)[5] 的安全性组件 [3]	● *bin32\mlhttps11.dll* ● *bin32\mlhttpsfips11.dll* ● *bin32\mlrsafips11.dll* ● *bin32\mljrsa11.dll* ● *bin32\mljtls11.dll* ● *bin32\mlrsa11.dll* ● *bin32\mltls11.dll* ● *bin32\defaultmem.dll* ● *bin32\libsb.dll*
安装脚本（部署统一数据库的脚本）	● *MobiLink\setup* ● *MobiLink\upgrade*
mluser 实用程序	● *bin32\mluser.exe* ● *bin32\mlodbc11.dll* ● *bin32\dbicu11.dll* ● *bin32\dbicudt11.dll*

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

说明	Windows 文件
mlstop 实用程序	● *bin32\mlstop.exe* ● *bin32\dbicu11.dll*
MobiLink 监控器	● *java\mlmon.jar* ● *java\JComponents1101.jar* ● *java\mlstream.jar* ● *java\jsyblib600.jar* ● *Sun\JavaHelp-2_0\jh.jar* ● *Sun\jaxb1.0* ● *bin32\jsyblib600.dll* ● *bin32\mlmon.exe* 对于使用 MobiLink 监控器的安全性：[3] ● *bin32\mlcecc11.dll* ● *bin32\mlcrsa11.dll* ● *bin32\mlcrsafips11.dll* ● *bin32\mlczlib11.dll*
MobiLink 插件和监控器的联机帮助	● *\documentation\en\htmlhelp\dbadmin_en11.chm*[1] ● *\documentation\en\htmlhelp\dbadmin_en11.map*[1]
MobiLink 重定向器	● *MobiLink\redirector*
通告程序	● *java\activation.jar*[2] ● *java\jodbc.jar* ● *java\log4j.jar*[4] ● *java\mailapi.jar*[2] ● *java\mlnotif.jar* ● *java\mlscript.jar* ● *java\smtp.jar*[2] ● *bin32\mljodbc11.dll* ● *bin32\mljstrm11.dll*
QAnywhere 必需的 MobiLink 服务器文件	● 通告程序文件 ● *java\commons-logging.jar* ● *java\commons-codec-1.3.jar* ● *java\commons-httpclient-3.0.jar* ● *java\jsyblib600.jar* ● *java\log4j.jar*[4] ● *java\mlscript.jar* ● *java\mlstream.jar* ● *java\qaconnector.jar* ● *bin32\jsyblib600.dll*

说明	Windows 文件
中继服务器出站启动器	● *bin32\rsoe.exe* 对于使用出站启动器的安全性： ● *bin32\mlcrsa11.dll*

[1] 对于德语、日语和中文版本，请将 en 分别替换为 de、ja 和 zh。

[2] 如果要重新分发应用程序，您必须直接从 Sun 获得这些文件。

[3] ECC 和 FIPS 要求您获取单独授权的 SQL Anywhere 安全性组件，并受出口法规的约束。版本 10 及更高版本的 SQL Anywhere 包含了 RSA 安全性。要订购此组件，请参见“单独授权的组件”一节 《SQL Anywhere 11 - 简介》。

[4] 如果要重新分发应用程序，您必须直接从 Apache 获得此文件。

[5] 还必须创建一个名为 *HKEY_LOCAL_MACHINE\SOFTWARE\Certicom\libsb* 的注册表项，并添加一个名为 expectedtag 且带有数据 5B0F4FA6E24AEF3B4407052EB04902711FD991B6 的 REG_BINARY 值。

Windows 64 位应用程序

所有目录都相对于 *install-dir*。有关 32 位 Windows 环境文件结构的详细信息，请参见“Windows 32 位应用程序”一节第 751 页。

说明	Windows 文件
MobiLink 服务器	● *bin64\mlodbc11.dll* ● *bin64\mlsrv11.exe* ● *bin64\mlsrv11.lic* ● *bin64\mlsql11.dll* ● *bin64\dbicu11.dll* ● *bin64\dbicudt11.dll*
语言库	● *bin64\dblgen11.dll*[1]
同步流库（支持版本 8 和版本 9 客户端）	● *bin64\mlhttp11.dll* ● *bin64\mlsock11.dll*

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

说明	Windows 文件
Java 同步逻辑	● *java\activation.jar*[2] ● *java\imap.jar*[2] ● *java\jodbc.jar* ● *java\log4j.jar*[2] ● *java\mailapi.jar*[2] ● *java\mlscript.jar* ● *java\mlsupport.jar* ● *java\pop3.jar*[2] ● *java\smtp.jar*[2] ● *bin64\mljava11.dll* ● *bin64\dbjodbc11.dll* ● *bin64\mljodbc11.dll*
.NET 同步逻辑	● *MobiLink\setup\dnet\mlDomConfig.xml* ● *bin64\mldnet11.dll* ● *bin64\dnetodbc11.dll* ● *Assembly\v2\iAnywhere.MobiLink.dll* ● *Assembly\v2\iAnywhere.MobiLink.Script.dll* ● *Assembly\v2\iAnywhere.MobiLink.Script.xml* ● *bin64\mlDomConfig.xsd*
版本 10 和版本 11 客户端 (mlsrv11 -x) 的安全性组件 [3]	● *bin64\mlecc_tls11.dll* ● *bin64\mlrsa_tls11.dll* ● *bin64\mlrsa_tls_fips11.dll* ● *bin64\sbgse2.dll*
版本 8 和版本 9 客户端 (mlsrv11 -xo)[5] 的安全性组件 [3]	● *bin64\mlhttps11.dll* ● *bin64\mlhttpsfips11.dll* ● *bin64\mlrsafips11.dll* ● *bin64\mljrsa11.dll* ● *bin64\mljtls11.dll* ● *bin64\mlrsa11.dll* ● *bin64\mltls11.dll* ● *bin64\defaultmem.dll* ● *bin64\libsb.dll*
安装脚本（部署统一数据库的脚本）	● *MobiLink\setup* ● *MobiLink\upgrade*
mluser 实用程序	● *bin64\mluser.exe* ● *bin64\mlodbc11.dll* ● *bin64\dbicu11.dll* ● *bin64\dbicudt11.dll*
mlstop 实用程序	● *bin64\mlstop.exe* ● *bin64\dbicu11.dll*

说明	Windows 文件
MobiLink 监控器	● *java\mlmon.jar* ● *java\JComponents1100.jar* ● *java\mlstream.jar* ● *java\jsyblib600.jar* ● *Sun\JavaHelp-2_0\jh.jar* ● *Sun\jaxb1.0* ● *bin64\jsyblib600.dll* ● *bin64\mlmon.exe* 对于使用 MobiLink 监控器的安全性：[3] ● *bin64\mlcecc11.dll* ● *bin64\mlcrsa11.dll* ● *bin64\mlcrsafips11.dll* ● *bin64\mlczlib11.dll*
MobiLink 插件和监控器的联机帮助	● *\documentation\en\htmlhelp\dbadmin_en11.chm*[1] ● *\documentation\en\htmlhelp\dbadmin_en11.map*[1]
MobiLink 重定向器	● *MobiLink\redirector*
通告程序	● *java\activation.jar*[2] ● *java\jodbc.jar* ● *java\log4j.jar*[4] ● *java\mailapi.jar*[2] ● *java\mlnotif.jar* ● *java\mlscript.jar* ● *java\smtp.jar*[2] ● *bin64\mljodbc11.dll* ● *bin64\mljstrm11.dll*
QAnywhere 必需的 MobiLink 服务器文件	● 通告程序文件 ● *java\commons-logging.jar* ● *java\commons-codec-1.3.jar* ● *java\commons-httpclient-3.0.jar* ● *java\jsyblib600.jar* ● *java\log4j.jar*[4] ● *java\mlscript.jar* ● *java\mlstream.jar* ● *java\qaconnector.jar* ● *bin64\jsyblib600.dll*

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

说明	Windows 文件
中继服务器出站启动器	● *bin64\rsoe.exe* 对于使用出站启动器的安全性： ● *bin64\mlcrsa11.dll*

[1] 对于德语、日语和中文版本，请将 en 分别替换为 de、ja 和 zh。

[2] 如果要重新分发应用程序，您必须直接从 Sun 获得这些文件。

[3] ECC 和 FIPS 要求您获取单独授权的 SQL Anywhere 安全性组件，并受出口法规的约束。版本 10 及更高版本的 SQL Anywhere 包含了 RSA 安全性。要订购此组件，请参见“单独授权的组件”一节 《SQL Anywhere 11 - 简介》。

[4] 如果要重新分发应用程序，您必须直接从 Apache 获得此文件。

[5] 还必须创建一个名为 *HKEY_LOCAL_MACHINE\SOFTWARE\Certicom\libsb* 的注册表项，并添加一个名为 expectedtag 且带有数据 5B0F4FA6E24AEF3B4407052EB04902711FD991B6 的 REG_BINARY 值。

Unix、Linux 和 Macintosh 上的 Unix 32 位应用程序

所有目录都相对于 *install-dir*。有关 64 位 Unix 环境文件结构的详细信息，请参见“Unix 和 Linux 上的 Unix 64 位应用程序”一节第 760 页。

说明	Unix 文件
MobiLink 服务器	● *bin32/mlsrv11* ● *bin32/mlsrv11.lic* ● *lib32/libdbodm11_r.so*[3] ● *lib32/libmlodbc11_r.so*[3] ● *lib32/libmlsql11_r.so*[3] ● *lib32/libdbtasks11_r.so*[3] ● *lib32/libdbicu11_r.so*[3] ● *lib32/libdbicudt11_r.so*[3] ● *lib32/libdbodbcinst11_r.so*[3]
语言库	● *res/dblgen11.res*[1]
版本 8 和版本 9 客户端的同步流库（部署您使用的同步流库）	● *lib32/libmlhttp11_r.so*[3] ● *lib32/libmlsock11_r.so*[3]

说明	Unix 文件
Java 同步逻辑	● *java/activation.jar*[2] ● *java/imap.jar*[2] ● *java/jodbc.jar* ● *java/log4j.jar*[5] ● *java/mailapi.jar*[2] ● *java/mlscript.jar* ● *java/mlsupport.jar* ● *java/pop3.jar*[2] ● *java/smtp.jar*[2] ● *lib32/libmljava11_r.so*[3] ● *lib32/libmljodbc11.so*[3]
.NET 同步逻辑	● 不适用
版本 10 和版本 11 客户端 (mlsrv11 -x) 的安全性组件 [4]	● *lib32/libmlecc_tls11_r.so*[3] ● *lib32/libmlrsa_tls11_r.so*[3]
版本 8 和版本 9 客户端 (mlsrv11 -xo) 的安全性组件 [4]	● *lib32/libmlhttps11_r.so*[3] ● *lib32/libmljrsa11_r.so*[3] ● *lib32/libmljtls11_r.so*[3] ● *lib32/libmlrsa11_r.so*[3] ● *lib32/libmltls11_r.so*[3]
安装脚本（部署统一数据库的脚本）	● *MobiLink/setup* ● *MobiLink/upgrade*
mluser 实用程序	● *bin32/mluser* ● *lib32/libmlodbc11_r.so*[3] ● *lib32/libdbicu11.so*[3] ● *lib32/libdbicudt11.so*[3]
mlstop 实用程序	● *bin32/mlstop* ● *lib32/libdbicu11.so*[3]
MobiLink 监控器	● *bin32/mlmon* ● *java/mlmon.jar* ● *java/mlstream.jar* ● *lib32/libjsyblib600_r.so*[3] ● *sun/JavaHelp-2_0/jh.jar* ● *sun/jaxb1.0/* ● *java/JComponents1100.jar* ● *java/jsyblib600.jar*

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

说明	Unix 文件
MobiLink 重定向器	● *Mobilink/redirector/redirector.config* ● *MobiLink/redirector/apache/* ● *MobiLink/redirector/java/* ● *MobiLink/redirector/MBusinessAnywhere/* ● *MobiLink/redirector/nsapi/*
MobiLink 插件和监控器的联机帮助	● *java/sqlanywhere_en11.jar*[1]
通告程序	● *java/activation.jar*[2] ● *java/jodbc.jar* ● *java/log4j.jar*[2] ● *java/mailapi.jar*[2] ● *java/mlnotif.jar* ● *java/mlscript.jar* ● *java/smtp.jar*[2] ● *lib32/libmljstrm11_r.so*[3]
QAnywhere 必需的 MobiLink 服务器文件	● 通告程序文件 ● *java/commons-codec-1.3.jar* ● *java/commons-httpclient-3.0.jar* ● *java/commons-logging.jar* ● *java/jsyblib600.jar* ● *java/log4j.jar*[5] ● *java/mlscript.jar* ● *java/mlstream.jar* ● *java/qaconnector.jar* ● *lib32/libjsyblib600_r.so*[3]
中继服务器出站启动器	● *bin32/rsoe* 对于使用出站启动器的安全性： ● *lib32/libmlcrsa11_r.so*[3]

[1] 对于德语、日语和中文版本，请将 en 分别替换为 de、ja 和 zh。

[2] 如果要重新分发应用程序，您必须直接从 Sun 获得这些文件。

[3] 如果是 Linux，文件扩展名为 *.so*。如果是 Macintosh，文件扩展名为 *.dylib*。

[4] 传送层安全性要求您获取单独授权的 SQL Anywhere 安全性组件，并受出口法规的约束。要订购此组件，请参见“单独授权的组件”一节 《SQL Anywhere 11 - 简介》 。

[5] 如果要重新分发应用程序，您必须直接从 Apache 获得这些文件。

Unix 和 Linux 上的 Unix 64 位应用程序

所有目录都相对于 *install-dir*。有关 32 位 Unix 环境文件结构的详细信息，请参见“Unix、Linux 和 Macintosh 上的 Unix 32 位应用程序”一节第 757 页。

说明	Unix 文件
MobiLink 服务器	● *bin64/mlsrv11* ● *bin64/mlsrv11.lic* ● *lib64/libdbodm11_r.so*[3] ● *lib64/libmlodbc11_r.so*[3] ● *lib64/libmlsql11_r.so*[3] ● *lib64/libdbtasks11_r.so*[3] ● *lib64/libdbicu11_r.so*[3] ● *lib64/libdbicudt11_r.so*[3] ● *lib64/libdbodbcinst11_r.so*[3]
语言库	● *res/dblgen11.res*[1]
版本 8 和版本 9 客户端的同步流库（部署您使用的同步流库）	● *lib64/libmlhttp11_r.so*[3] ● *lib64/libmlsock11_r.so*[3]
Java 同步逻辑	● *java/activation.jar*[2] ● *java/imap.jar*[2] ● *java/jodbc.jar* ● *java/log4j.jar*[5] ● *java/mailapi.jar*[2] ● *java/mlscript.jar* ● *java/mlsupport.jar* ● *java/pop3.jar*[2] ● *java/smtp.jar*[2] ● *lib64/libmljava11_r.so*[3] ● *lib64/libmljodbc11.so*[3]
.NET 同步逻辑	● 不适用
版本 10 和版本 11 客户端 (mlsrv11 -x) 的安全性组件 [4]	● *lib64/libmlecc_tls11_r.so*[3] ● *lib64/libmlrsa_tls11_r.so*[3]
版本 8 和版本 9 客户端 (mlsrv11 -xo) 的安全性组件 [4]	● *lib64/libmlhttps11_r.so*[3] ● *lib64/libmljrsa11_r.so*[3] ● *lib64/libmljtls11_r.so*[3] ● *lib64/libmlrsa11_r.so*[3] ● *lib64/libmltls11_r.so*[3]
安装脚本（部署统一数据库的脚本）	● *MobiLink/setup* ● *MobiLink/upgrade*

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

说明	Unix 文件
mluser 实用程序	● *bin64/mluser* ● *lib64/libmlodbc11_r.so*[3] ● *lib64/libdbicu11.so*[3] ● *lib64/libdbicudt11.so*[3]
mlstop 实用程序	● *bin64/mlstop* ● *lib64/libdbicu11.so*[3]
MobiLink 监控器	● *bin64/mlmon* ● *java/mlmon.jar* ● *java/mlstream.jar* ● *lib64/libjsyblib600_r.so*[3] ● *sun/JavaHelp-2_0/jh.jar* ● *sun/jaxb1.0* ● *java/JComponents1100.jar* ● *java/jsyblib600.jar*
MobiLink 重定向器	● *Mobilink/redirector/redirector.config* ● *MobiLink/redirector/apache/* ● *MobiLink/redirector/java/* ● *MobiLink/redirector/MBusinessAnywhere/* ● *MobiLink/redirector/nsapi/*
MobiLink 插件和监控器的联机帮助	● *java/sqlanywhere_en11.jar*[1]
通告程序	● *java/activation.jar*[2] ● *java/jodbc.jar* ● *java/log4j.jar*[2] ● *java/mailapi.jar*[2] ● *java/mlnotif.jar* ● *java/mlscript.jar* ● *java/smtp.jar*[2]
QAnywhere 必需的 MobiLink 服务器文件	● 通告程序文件 ● *java/commons-codec-1.3.jar* ● *java/commons-httpclient-3.0.jar* ● *java/commons-logging.jar* ● *java/jsyblib600.jar* ● *java/log4j.jar*[5] ● *java/mlscript.jar* ● *java/mlstream.jar* ● *java/qaconnector.jar* ● *lib64/libjsyblib600_r.so*[3]

说明	Unix 文件
中继服务器出站启动器	● *bin64/rsoe* 对于使用出站启动器的安全性： ● *lib64/libmlcrsa11_r.so*[3]

[1] 对于德语、日语和中文版本，请将 en 分别替换为 de、ja 和 zh。

[2] 如果要重新分发应用程序，您必须直接从 Sun 获得这些文件。

[3] 如果是 Solaris SPARC 和 Linux，文件扩展名为 *.so*。如果是 AIX，文件扩展名为 *.a*。

[4] 传送层安全性要求您获取单独授权的 SQL Anywhere 安全性组件，并受出口法规的约束。要订购此组件，请参见“单独授权的组件”一节 《SQL Anywhere 11 - 简介》。

[5] 如果要重新分发应用程序，您必须直接从 Apache 获得这些文件。

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

部署 SQL Anywhere MobiLink 客户端

注意

- 对于 SQL Anywhere 客户端，您需要部署 SQL Anywhere 数据库服务器和 MobiLink 客户端。

  请参见“部署数据库和应用程序”《SQL Anywhere 服务器 - 编程》。
- 如果您准备重新分发 MobiLink 同步客户端，则除了那些 SQL Anywhere 数据库所需的文件之外，还需要在安装中包括以下文件。
- 部署以下文件时，除非另行说明，否则请将文件放在同一个目录结构中。
- 要部署 Sybase Central，请参见“部署管理工具”一节《SQL Anywhere 服务器 - 编程》。
- 有一个用于 Windows 的部署向导。请参见“使用 [部署向导]”一节《SQL Anywhere 服务器 - 编程》。
- 对于 Windows Mobile 部署，以下列出的 *bin32* 目录中的文件位于 *ce\arm.50* 目录中。.NET 程序集放在 *ce\Assembly\v2* 目录中。

Windows 应用程序

所有目录都相对于 *install-dir*。

说明	Windows 文件
MobiLink 同步客户端 (dbmlsync)	● *bin32\dbcon11.dll*[2] ● *bin32\dbicu11.dll*[3] ● *bin32\dblgen11.dll*[1] ● *bin32\dblib11.dll* ● *bin32\dbmlsync.exe* ● *bin32\dbtool11.dll*[2]
Dbmlsync 集成组件（不建议使用）	● MobiLink 同步客户端文件 ● 可视化组件：*bin32\dbmlsynccomg.dll* ● 非可视化组件：*bin32\dbmlsynccom.dll*
安全性组件 [2]	● *bin32\mlcecc11.dll* ● *bin32\mlcrsa11.dll* ● *bin32\mlcrsafips11.dll* ● *bin32\sbgse2.dll*
ActiveSync 和 HotSync 实用程序	● *bin32\mlasinst.exe* ● *bin32\mlasdesk.dll* ● *bin32\dbcon11.exe* ● *ce\chip\mlasdev.dll*（其中 *chip* 可以是任何支持的芯片，例如 arm.50）

Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

说明	Windows 文件
监听器	● *bin32\dblgen11.dll*[1] ● *bin32\dblsn.exe* ● *bin32\lsn_udp.dll* ● *bin32\lsn_swi510.dll* ● *bin32\maac555.dll* ● *bin32\maac750.dll* ● *bin32\maac750r3.dll* ● *bin32\mabridge.dll*

[1] 对于德语、日语和中文版本，请将 en 分别替换为 de、ja 和 zh。

[2] 在 Windows Mobile 上不需要，除非您使用 dbtools 界面。

[3] 如果使用 dbinit -zn UTF8BIN 对数据库进行了初始化，则不需要。请参见“初始化实用程序 (dbinit)”一节 《SQL Anywhere 服务器 - 数据库管理》。

Unix、Linux 和 Macintosh 上的 Unix 应用程序

所有目录都相对于 *install-dir*。

说明	Unix 文件
MobiLink 同步客户端	● *bin32/dbmlsync* ● *res/dblgen11.res* ● *lib32/libdbcon11_r.so*[1] ● *lib32/libdbicu11_r.so*[1] ● *lib32/libdblib11_r.so*[1] ● *lib32/libdbtool11_r.so*[1]
安全性组件 [2]	● *lib32/libmlcecc11_r.so*[1] ● *lib32/libmlcrsa11_r.so*[1]

[1] 如果是 Linux，文件扩展名为 *.so*。如果是 Macintosh，文件扩展名为 *.dylib*。

[2] 传送层安全性要求您获取单独授权的 SQL Anywhere 安全性组件，并受出口法规的约束。要订购此组件，请参见“单独授权的组件”一节 《SQL Anywhere 11 - 简介》。

Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

部署 UltraLite MobiLink 客户端

对于 UltraLite 客户端，UltraLite 运行时库或 UltraLite 组件包括必需的同步流函数。UltraLite 运行时库将编译到您的应用程序中。部署应该按照您的许可协议来进行。

另请参见

- “将 UltraLite 部署到设备” 《UltraLite - 数据库管理和参考》
- C/C++： “部署 Palm 应用程序”一节 《UltraLite - C 及 C++ 编程》 和 “部署 Windows Mobile 应用程序”一节 《UltraLite - C 及 C++ 编程》
- M-Business Anywhere： “部署 UltraLite for M-Business Anywhere 应用程序”一节 《UltraLite - M-Business Anywhere 编程》
- .NET： “第 5 课：构建和部署应用程序”一节 《UltraLite - .NET 编程》

Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

部署 QAnywhere 应用程序

QAnywhere 提供 C++、Java 和 .NET API 来支持 SQL Anywhere 消息存储库。Java 和 .NET API 也支持 UltraLite 消息存储库。部署 QAnywhere 应用程序所需的文件要根据您的 Windows 环境、消息存储库类型以及所选择的 API 来确定。如果准备部署移动 Web 服务应用程序，则还需要附加的文件。

除了下面列出的文件外，QAnywhere 应用程序需要：

- 在“部署 SQL Anywhere MobiLink 客户端”一节第 763 页的 "MobiLink 同步客户端"、"监听器" 和可选的 "安全" 部分中列出的所有文件。只有使用推式通知时（缺省情况）才需要监听器文件。
- “部署数据库服务器”一节 《SQL Anywhere 服务器 - 编程》中的 dbeng11 或 dbsrv11 文件。

要部署 Sybase Central，请参见“部署管理工具”一节 《SQL Anywhere 服务器 - 编程》。

Windows 应用程序

所有目录都相对于 *install-dir*。

有关 Windows Mobile 环境文件结构的详细信息，请参见“Windows Mobile 应用程序”一节第 768 页。

以下是安装 SQL Anywhere 消息存储库所需文件的列表。

客户端 API	Windows 文件
C++	● *bin32\qany11.dll* ● *bin32\qaagent.exe* ● *bin32\qastop.exe*
Java	● *bin32\qaagent.exe* ● *bin32\qastop.exe* ● *java\qaclient.jar* ● *java\jodbc.jar* 对于移动 Web 服务应用程序，还需要以下文件： ● *java\iawsrt.jar* ● *java\jaxrpc.jar*

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

客户端 API	Windows 文件
.NET	● *bin32\qazlib.dll* ● *bin32\qaagent.exe* ● *bin32\qastop.exe* ● *assembly\v2\iAnywhere.QAnywhere.Client.dll* ● *assembly\v2\iAnywhere.QAnywhere.Resources.dll* ● *assembly\v2\iAnywhere.Data.SQLAnywhere.dll* 对于移动 Web 服务应用程序，还需要以下文件： ● *Assembly\v2\iAnywhere.QAnywhere.WS.dll*

以下是安装 UltraLite 消息存储库并使用 QAnywhere 代理进行部署所需文件的列表。

客户端 API	Windows 文件
Java	● *bin32\qauagent.exe* ● *bin32\qastop.exe* ● *bin32\qadbiuljni.dll* ● *java\qaclient.jar* 对于移动 Web 服务应用程序，还需要以下文件： ● *java\iawsrt.jar* ● *java\jaxrpc.jar*
.NET	● *bin32\qazlib.dll* ● *bin32\qauagent.exe* ● *bin32\qastop.exe* ● *assembly\v2\iAnywhere.QAnywhere.Client.dll* ● *assembly\v2\iAnywhere.QAnywhere.Resources.dll* ● *ultralite\ultralite.NET\assembly\v2\iAnywhere.Data.UltraLite.dll* 对于移动 Web 服务应用程序，还需要以下文件： ● *Assembly\v2\iAnywhere.QAnywhere.WS.dll*

创建 UltraLite 消息存储库时，必须使用 UltraLite 创建数据库实用程序创建一个 udb 数据库文件，然后再使用 QAnywhere UltraLite 代理的 -si 选项初始化此数据库。请参见“UltraLite 创建数据库实用程序 (ulcreate)”一节 《UltraLite - 数据库管理和参考》和“qauagent 实用程序”一节《QAnywhere》。

以下是使用 QAnywhere 独立客户端安装部署所需文件的列表。

客户端 API	Windows 文件
Java	● *java\qastandaloneclient.jar* ● *bin32\qadbiulsjni.dll*

客户端 API	Windows 文件
.NET	● *assembly\v2\iAnywhere.QAnywhere.StandAloneClient.dll* ● *assembly\v2\iAnywhere.QAnywhere.Resources.dll* ● *ultralite\ultralite.NET\assembly\v2\iAnywhere.Data.UltraLite.dll*

Windows Mobile 应用程序

所有目录都相对于 *install-dir*。

有关 Windows 环境文件结构的详细信息，请参见“Windows 应用程序”一节第 766 页。

以下是安装 SQL Anywhere 消息存储库所需文件的列表。

客户端 API	Windows Mobile 文件
C++	● *ce\arm.50\qany11.dll* ● *ce\arm.50\qaagent.exe* ● *ce\arm.50\qastop.exe*
Java	● *ce\arm.50\qaagent.exe* ● *ce\arm.50\qastop.exe* ● *java\qaclient.jar* ● *java\jodbc.jar* 对于移动 Web 服务应用程序，还需要以下文件： ● *java\iawsrt.jar* ● *java\jaxrpc.jar*
.NET	● *ce\arm.50\qazlib.dll* ● *ce\arm.50\qaagent.exe* ● *ce\arm.50\qastop.exe* ● *ce\assembly\v2\iAnywhere.QAnywhere.Client.dll* ● *ce\assembly\v2\iAnywhere.QAnywhere.Resources.dll* ● *ce\assembly\v2\iAnywhere.Data.SQLAnywhere.dll* 对于移动 Web 服务应用程序，还需要以下文件： ● *ce\Assembly\v2\iAnywhere.QAnywhere.WS.dll*

以下是安装 UltraLite 消息存储库并使用 QAnywhere 代理进行部署所需文件的列表。

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

客户端 API	Windows Mobile 文件
Java	● *ce\arm.50\qauagent.exe* ● *ce\arm.50\qastop.exe* ● *ce\arm.50\qadbiuljni.dll* ● *java\qaclient.jar* 对于移动 Web 服务应用程序，还需要以下文件： ● *java\iawsrt.jar* ● *java\jaxrpc.jar*
.NET	● *ce\arm.50\qazlib.dll* ● *ce\arm.50\qauagent.exe* ● *ce\arm.50\qastop.exe* ● *ce\assembly\v2\iAnywhere.QAnywhere.Client.dll* ● *ce\assembly\v2\iAnywhere.QAnywhere.Resources.dll* ● *ultralite\ultralite.NET\ce\assembly\v2\iAnywhere.Data.UltraLite.dll* 对于移动 Web 服务应用程序，还需要以下文件： ● *ce\Assembly\v2\iAnywhere.QAnywhere.WS.dll*

创建 UltraLite 消息存储库时，必须使用 UltraLite 创建数据库实用程序创建一个 udb 数据库文件，然后再使用 QAnywhere UltraLite 代理的 -si 选项初始化此数据库。请参见“UltraLite 创建数据库实用程序 (ulcreate)”一节 《UltraLite - 数据库管理和参考》和“qauagent 实用程序”一节 《QAnywhere》 。

以下是使用 QAnywhere 独立客户端安装部署所需文件的列表。

客户端 API	Windows Mobile 文件
Java	● *java\qastandaloneclient.jar* ● *ce\arm.50\qadbiulsjni.dll*
.NET	● *ce\assembly\v2\iAnywhere.QAnywhere.StandAloneClient.dll* ● *ce\assembly\v2\iAnywhere.QAnywhere.Resources.dll* ● *ultralite\ultralite.NET\ce\assembly\v2\iAnywhere.Data.UltraLite.dll*

注册 QAnywhere .NET API DLL

QAnywhere .NET API DLL（*Assembly\v2\iAnywhere.QAnywhere.Client.dll*）需要在 Windows（Windows Mobile 除外）的全局程序集高速缓存中注册。全局程序集高速缓存列出了计算机上所有已注册的程序。在安装 SQL Anywhere 时，安装程序会对它进行注册。在 Windows Mobile 上无需注册 DLL。

如果要部署 QAnywhere，必须使用包含在 .NET Framework 中的 gacutil 实用程序注册 QAnywhere .NET API DLL（*Assembly\v2\iAnywhere.QAnywhere.Client.dll*）。

术语表

术语表 773

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 790 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 800 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 800 页
- “代码页”一节第 775 页
- “归类”一节第 779 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 787 页
- “隔离级别”一节第 778 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 800 页
- “外键”一节第 792 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 791 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 790 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 777 页

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

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 779 页
- “语句级触发器”一节第 797 页
- “完整性”一节第 793 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 798 页

代理 ID

另请参见： “客户端消息存储库 ID”一节第 783 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 800 页
- “编码”一节第 773 页
- “归类”一节第 779 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 789 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 790 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 791 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 777 页
- “项目”一节第 795 页
- “发布更新”一节第 776 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。

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

另请参见：

- “复制”一节第 777 页
- “发布”一节第 776 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 777 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 777 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 784 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 784 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 777 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 777 页
- “发布更新”一节第 776 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 783 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 789 页

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

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 800 页
- “代码页”一节第 775 页
- “编码”一节第 773 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 775 页
- “语句级触发器”一节第 797 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 787 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 780 页
- “jConnect”一节第 780 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 780 页
- “iAnywhere JDBC 驱动程序”一节第 779 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 780 页
- “iAnywhere JDBC 驱动程序”一节第 779 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 783 页
- “视图”一节第 787 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。

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

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 777 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 799 页
- “外键约束”一节第 793 页
- “主键约束”一节第 800 页
- “唯一约束”一节第 794 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 787 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 788 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 792 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 783 页
- “全局临时表”一节第 786 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。

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

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 778 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 783 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 777 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 783 页
- “生成的连接条件”一节第 788 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 782 页
- “全局临时表”一节第 786 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 778 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 777 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 791 页
- “同步”一节第 791 页
- “UltraLite”一节第 792 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。

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

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 783 页
- “外连接”一节第 793 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 777 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 783 页
- “局部临时表”一节第 782 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 787 页
- “事务日志镜像”一节第 788 页
- “完全备份”一节第 793 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。

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

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 777 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 780 页
- “视图”一节第 787 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 781 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 787 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 783 页
- “连接条件”一节第 783 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。

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

另请参见： “域”一节第 797 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 792 页
- “主键”一节第 800 页
- “数据库管理系统 (DBMS)”一节第 789 页
- “关系数据库管理系统 (RDBMS)”一节第 778 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 778 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 790 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 789 页
- “SYS”一节第 791 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 776 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 785 页
- “SQL”一节第 790 页
- “数据库管理系统 (DBMS)”一节第 789 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。

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

另请参见：

- “隔离级别”一节第 778 页
- “并发”一节第 773 页
- “完整性”一节第 793 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 791 页
- “复制”一节第 777 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 777 页
- “监听器”一节第 781 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 784 页
- “SQL Remote”一节第 790 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 777 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 786 页
- “服务器启动的同步”一节第 777 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 792 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。

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

另请参见：

- “主键”一节第 800 页
- “外表”一节第 792 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 799 页
- “检查约束”一节第 781 页
- “主键约束”一节第 800 页
- “唯一约束”一节第 794 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 783 页
- “内连接”一节第 785 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 799 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 774 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 777 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 778 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 793 页
- “主键约束”一节第 800 页
- “约束”一节第 799 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 794 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 781 页

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

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 777 页
- “发布”一节第 776 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 782 页
- “服务器消息存储库”一节第 777 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 777 页
- “统一数据库”一节第 791 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 777 页
- “FILE”一节第 777 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 799 页
- “用户定义数据类型”一节第 797 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 792 页

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

用户定义数据类型

请参见“域”一节第 797 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 797 页
- “游标位置”一节第 797 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 797 页
- “游标位置”一节第 797 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 797 页
- “游标结果集”一节第 797 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 775 页
- “行级触发器”一节第 779 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 788 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 776 页
- “MobiLink 用户”一节第 785 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 785 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 776 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 791 页
- “统一数据库”一节第 791 页

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

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 781 页
- “外键约束”一节第 793 页
- “主键约束”一节第 800 页
- “唯一约束”一节第 794 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 777 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 787 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 791 页
- “基于 SQL 的同步”一节第 781 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 792 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 799 页
- “检查约束”一节第 781 页
- “外键约束”一节第 793 页
- “唯一约束”一节第 794 页
- “完整性”一节第 793 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 775 页
- “编码”一节第 773 页
- “归类”一节第 779 页

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

索引

其它

.NET
  MobiLink 同步脚本, 551
  MobiLink 基于对象的数据流, 609
  MobiLink 数据类型, 556
  MobiLink 服务器 API 参考, 567
.NET CLR
  MobiLink 选项, 86
.NET MobiLink 服务器 API (*见* 用于 .NET 的 MobiLink 服务器 API)
.NET 类
  为 .NET 同步逻辑实例化, 555
.NET 同步技术
  关于, 562
.NET 同步逻辑
  .NET 类实例化, 555
  DBCommand, 567
  DBConnection 接口, 569
  DBConnectionContext, 570
  DBParameter 类, 573
  DBParameterCollection 类, 576
  DBRowReader 接口, 581
  LogCallback 委派, 587
  LogMessage 类, 587
  MessageType 枚举, 588
  MobiLink 性能, 162
  MobiLink 服务器 API, 567
  ServerContext 接口, 589
  ShutdownCallback 委派, 593
  SQLType 枚举, 593
  SynchronizationException 类, 601
  调试, 560
  在 32 位 Unix 上部署, 757
  在 32 位 Windows 上部署, 751
  在 64 位 Unix 上部署, 760
  在 64 位 Windows 上部署, 754
  支持的语言, 552
  方法, 557
  示例, 565
  设置, 553
.NET 同步示例
  MobiLink .NET 同步逻辑, 565
[图表] 窗格
  MobiLink 监控器, 174
[详细信息表] 窗格
  MobiLink 监控器, 171
[选取框] 工具
  MobiLink 监控器一览表窗格, 176
[选项] 窗口
  MobiLink 监控器, 176
[运用图形] 窗格
  MobiLink 监控器, 172
@data 选项
  MobiLink 停止实用程序 (mlstop), 649
  MobiLink 服务器 (mlsrv11), 46
  MobiLink 用户验证 (mluser), 650
@EmployeeID 变量
  用于 MobiLink 主键池, 135
-a 选项
  MobiLink 服务器 (mlsrv11), 47
-bn 选项
  MobiLink 服务器 (mlsrv11), 49
-b 选项
  MobiLink 服务器 (mlsrv11), 48
-classic 选项
  MobiLink 服务器 (mlsrv11) -sl java, 87
-classpath 选项
  MobiLink 服务器 (mlsrv11) -sl java, 87
-clrConGC 选项
  MobiLink 服务器 (mlsrv11) -sl dnet, 86
-clrFlavor 选项
  MobiLink 服务器 (mlsrv11) -sl dnet, 86
-clrVersion 选项
  MobiLink 服务器 (mlsrv11) -sl dnet, 86
-cm 选项
  MobiLink 服务器 (mlsrv11), 51
-cn 选项
  MobiLink 服务器 (mlsrv11), 52
-cp 选项
  MobiLink 服务器 (mlsrv11) -sl java, 87
-cr 选项
  MobiLink 服务器 (mlsrv11), 53
-cs 选项
  MobiLink 服务器 (mlsrv11), 54
-ct 选项
  MobiLink 服务器 (mlsrv11), 55
-c 选项
  MobiLink 服务器 (mlsrv11), 50
  MobiLink 用户验证 (mluser), 650
-dl 选项

MobiLink 服务器 (mlsrv11), 56
MobiLink 用户验证 (mluser), 650
-DMLStartClasses
Java 用户定义的启动类, 501
MobiLink 服务器 (mlsrv11) -sl java, 87
-dr 选项
MobiLink 服务器 (mlsrv11), 57
-dsd 选项
MobiLink 服务器 (mlsrv11), 59
-ds 选项
MobiLink 服务器 (mlsrv11), 58
-dt 选项
MobiLink 服务器 (mlsrv11), 60
-d 选项
MobiLink 服务器 (mlsrv11) -sl java, 87
MobiLink 用户验证 (mluser), 650
-esu 选项
MobiLink 服务器 (mlsrv11), 62
-et 选项
MobiLink 服务器 (mlsrv11), 63
-e 选项
MobiLink 服务器 (mlsrv11), 61
-fr 选项
MobiLink 服务器 (mlsrv11), 66
-ftr 选项
MobiLink 服务器 (mlsrv11), 67
-f 选项
MobiLink 停止实用程序 (mlstop), 649
MobiLink 服务器 (mlsrv11), 64
MobiLink 用户验证 (mluser), 650
-hotspot 选项
MobiLink 服务器 (mlsrv11) -sl java, 87
-h 选项
MobiLink 停止实用程序 (mlstop), 649
-jrepath 选项
MobiLink 服务器 (mlsrv11) -sl java, 87
-lsc 选项
MobiLink 服务器 (mlsrv11), 68
-MLAutoLoadPath 选项
MobiLink 服务器 (mlsrv11) -sl dnet, 86
关于, 563
-MLDomConfigFile 选项
MobiLink 服务器 (mlsrv11) -sl dnet, 86
关于, 563
-MLStartClasses
.NET 用户定义的启动类, 557
MobiLink 服务器 (mlsrv11) -sl dnet, 86
-m 选项
MobiLink 服务器 (mlsrv11), 69
QAnywhere 启动 MobiLink 服务器 (mlsrv11), 69
-nba 选项
MobiLink 服务器 (mlsrv11), 70
-nc 选项
MobiLink 服务器 (mlsrv11), 71
-notifier 选项
MobiLink 服务器 (mlsrv11), 72
-on 选项
MobiLink 服务器 (mlsrv11), 74
-oq 选项
MobiLink 服务器 (mlsrv11), 75
-os 选项
MobiLink 服务器 (mlsrv11), 76
MobiLink 用户验证 (mluser), 650
-ot 选项
MobiLink 服务器 (mlsrv11), 77
MobiLink 用户验证 (mluser), 650
-o 选项
MobiLink 服务器 (mlsrv11), 73
MobiLink 用户验证 (mluser), 650
-pc 选项
MobiLink 用户验证 (mluser), 650
-ppv 选项
MobiLink 服务器 (mlsrv11), 78
-p 选项
MobiLink 用户验证 (mluser), 650
-q 选项
MobiLink 停止实用程序 (mlstop), 649
MobiLink 服务器 (mlsrv11), 82
-rd 选项
MobiLink 服务器 (mlsrv11), 84
-r 选项
MobiLink 服务器 (mlsrv11), 83
-server 选项
MobiLink 服务器 (mlsrv11) -sl java, 87
-sl dnet 选项
MobiLink 服务器 (mlsrv11), 86
使用 -MLAutoLoadPath, 554
使用 -MLDomConfigFile, 563
用户定义的启动类, 557
-sl java 选项
MobiLink 服务器 (mlsrv11), 87
用户定义的启动类, 501
-sm 选项
MobiLink 服务器 (mlsrv11), 89

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

-s 选项
MobiLink 服务器 (mlsrv11), 85
-tc 选项
MobiLink 服务器 (mlsrv11), 91
-tf 选项
MobiLink 服务器 (mlsrv11), 92
-tx 选项
MobiLink 服务器 (mlsrv11), 93
-t 选项
MobiLink 停止实用程序 (mlstop), 649
-ud 选项
MobiLink 服务器 (mlsrv11), 94
-ui 选项
MobiLink 服务器 (mlsrv11), 95
-urc 选项
MobiLink 性能优点, 162
-ux 选项
MobiLink 服务器 (mlsrv11), 96
-u 选项
MobiLink 用户验证 (mluser), 650
-v+ 选项
MobiLink 服务器 (mlsrv11), 97
-vc 选项
MobiLink 服务器 (mlsrv11), 97
-verbose 选项
MobiLink 服务器 (mlsrv11) -sl java, 87
-ve 选项
MobiLink 服务器 (mlsrv11), 97
-vf 选项
MobiLink 服务器 (mlsrv11), 97
-vh 选项
MobiLink 服务器 (mlsrv11), 97
-vm 选项
MobiLink 服务器 (mlsrv11), 97
-vn 选项
MobiLink 服务器 (mlsrv11), 97
-vp 选项
MobiLink 服务器 (mlsrv11), 97
-vr 选项
MobiLink 服务器 (mlsrv11), 97
-vs 选项
MobiLink 服务器 (mlsrv11), 97
-vt 选项
MobiLink 服务器 (mlsrv11), 97
-vu 选项
MobiLink 服务器 (mlsrv11), 97
-v 选项
MobiLink [dbmlsync] 性能, 161
MobiLink 服务器 (mlsrv11), 97
-wu 选项
MobiLink 服务器 (mlsrv11), 101
-w 选项
MobiLink 停止实用程序 (mlstop), 649
MobiLink 服务器 (mlsrv11), 100
-xo 选项
MobiLink 服务器 (mlsrv11), 107
-x 选项
MobiLink 服务器 (mlsrv11), 102
MobiLink 服务器 (mlsrv11) -sl java, 87
-zp 选项
MobiLink 服务器 (mlsrv11), 111
-zs 选项
MobiLink 服务器 (mlsrv11), 112
共享服务器状态, 112
-zt 选项
MobiLink 服务器 (mlsrv11), 113
-zus 选项
MobiLink 服务器 (mlsrv11), 115
-zu 选项
MobiLink 服务器 (mlsrv11), 114
-zwd 选项
MobiLink 服务器 (mlsrv11), 117
-zwe 选项
MobiLink 服务器 (mlsrv11), 118
-zw 选项
MobiLink 服务器 (mlsrv11), 116

A

a.
MobiLink 命名参数前缀, 299
MobiLink 用户定义的参数前缀, 301
ActiveSync
在 Windows 上部署 MobiLink 客户端, 763
active 属性
MobiLink 监控器同步统计信息, 181
Adaptive Server Enterprise
begin_connection_autocommit 事件, 344
MobiLink 同步, 10
MobiLink 数据映射, 696
MobiLink 统一数据库, 10
MobiLink 隔离级别, 156
StaticCursorLongColBuffLen, 10
在 MobiLink 中使用 DDL, 344
Add(object value) 方法 [ML .NET]

DBParameterCollection 类语法, 577
addErrorListener 方法 [ML Java]
ServerContext 语法, 532
addInfoListener 方法 [ML Java]
ServerContext 语法, 530
addShutdownListener 方法 [ML Java]
ServerContext 语法, 532
addWarningListener 方法 [ML Java]
ServerContext 语法, 532
AdventureWorks
同步问题, 20
Apache
为 Apache 重定向器配置, 269
为 MobiLink 配置 Servlet 重定向器, 267
Apache Tomcat
servlet 重定向器, 267
Apache Web 服务器
配置 Apache 重定向器, 269
Apache 重定向器
配置, 269
API
用于 .NET 的 MobiLink 服务器 API, 567
面向 Java 的 MobiLink 服务器 API, 508
ASE
(*参见* Adaptive Server Enterprise)
authenticate_file_transfer
连接事件, 331
authenticate_parameters
连接事件, 332
authenticate_user
连接事件, 335
authenticate_user_hashed
连接事件, 339
authenticate_user 属性
MobiLink 监控器同步统计信息, 181
authentication_status 同步参数
关于, 335
AvantGo (*见* M-Business Anywhere)
安全
监控器用户, 214
安全性
MobiLink 用户验证 (mluser) 实用程序, 650
安装
单独计算机上的监控器, 220
安装脚本
MobiLink 系统数据库, 7
MobiLink 统一数据库, 6

B

begin_connection
连接事件, 343
begin_connection_autocommit
连接事件, 344
begin_download
表事件, 347
连接事件, 345
begin_download_deletes
表事件, 349
begin_download_rows
表事件, 352
begin_publication
连接事件, 355
begin_synchronization
表事件, 360
连接事件, 358
begin_sync 属性
MobiLink 监控器同步统计信息, 181
begin_upload
表事件, 364
连接事件, 362
begin_upload_deletes
表事件, 366
begin_upload_rows
表事件, 368
BLOB
从 ASE 下载, 10
buffer_size 协议选项
适用于 HTTP 的 MobiLink 服务器 (mlsrv11) -x 选项, 104
适用于 HTTPS 的 MobiLink 服务器 (mlsrv11) -x 选项, 105
版本
关于 MobiLink 同步脚本, 303
添加脚本版本, 304
帮助
技术支持, xviii
包
术语定义, 773
保存监控器数据
MobiLink 监控器, 179
报告错误
MobiLink 同步, 317
被引用对象
术语定义, 773
必需的脚本

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

MobiLink, 305
编码
术语定义, 773
编写
.NET 同步逻辑, 551
Java 同步逻辑, 493
编写 .NET 同步逻辑
关于, 555
编写 download_cursor 脚本
MobiLink, 313
编写 download_delete_cursor 脚本
MobiLink, 314
编写 Java 同步逻辑
关于, 497
编写 upload_delete 脚本
MobiLink, 310
编写 upload_fetch 脚本
MobiLink, 311
编写 upload_insert 脚本
MobiLink, 309
编写 upload_update 脚本
MobiLink, 310
编写同步脚本
SQL, 293
支持的 DBMS 脚本策略, 8
编写用于处理错误的脚本
MobiLink, 317
编写用于上载行的脚本
MobiLink, 309
编写用于下载行的脚本
MobiLink, 312
标识符
在 IBM DB2 LUW 中的最大长度, 655
在 IBM DB2 主机中的最大长度, 624, 655, 656
术语定义, 773
表
MobiLink ml_table 系统表, 691
分区, 127
将统一表与 MobiLink 远程表相关联, 5
表级脚本
定义, 298
表脚本
使用 Sybase Central 添加, 306
关于, 298
删除 .NET 脚本, 629
删除 Java 脚本, 631
删除 SQL 脚本, 639
定义, 295, 298
按字母顺序排列的 MobiLink 脚本列表, 320
添加 .NET 脚本, 629
添加 Java 脚本, 631
添加 SQL 脚, 639
表空间容量
DB2 MobiLink 统一数据库, 13
并发
MobiLink 性能, 160
术语定义, 773
不一致
MobiLink 冲突处理, 137
部署 (*见* 部署)
MobiLink 应用程序, 749
MobiLink 应用程序和数据库, 749
MobiLink 性能, 159
MobiLink 服务器, 751
MobiLink 概述, 750
QAnywhere 应用程序, 766
SQL Anywhere MobiLink 客户端, 763
UltraLite MobiLink 客户端, 765
部署 MobiLink 服务器
关于, 751
部署 MobiLink 应用程序
关于, 749
部署 QAnywhere 客户端
关于, 766
部署 SQL Anywhere MobiLink 客户端
关于, 763
部署 UltraLite MobiLink 客户端
关于, 765
部署概述
MobiLink, 750
部署远程数据库
关于, 749

C

C# 编程语言
MobiLink .NET 支持, 552
MobiLink 同步脚本, 551
MobiLink 选项, 86
C++ 编程语言
MobiLink .NET 支持, 552
CHAR 列
ASE MobiLink 统一数据库, 10
DB2 MobiLink 统一数据库, 13
MobiLink 服务器 (mlsrv11) -b 选项, 48

MobiLink 问题, 8
Oracle MobiLink 统一数据库, 23
SQL Server MobiLink 统一数据库, 19
CHAR 数据类型
MobiLink 和其它 DBMS, 8
CHECK 约束
术语定义, 781
重定向器
Apache 本机, 269
负载平衡示例（适用于不支持服务器组的重定向器）, 260
负载平衡示例（适用于支持服务器组的重定向器）, 258
ISAPI, 265
M-Business Anywhere, 271
Microsoft Web 服务器, 265
MobiLink 服务器组, 255
配置（适用于不支持服务器组的重定向器）, 258
Unix 上的 iPlanet, 263
Unix 上的 NSAPI 版本, 263
Unix 上的 Sun One, 263
Windows 上的 iPlanet, 261
Windows 上的 NSAPI 版本, 261
Windows 上的 Sun One, 261
为 Tomcat 配置 servlet 重定向器, 267
何时使用, 251
使用, 250
关于, 249
在 32 位 Unix 上部署 MobiLink 服务器, 757
在 32 位 Windows 上部署 MobiLink 服务器, 751
在 64 位 Unix 上部署 MobiLink 服务器, 760
在 64 位 Windows 上部署 MobiLink 服务器, 754
指定位置（适用于不支持服务器组的重定向器）, 258
术语定义, 774
适用于 Apache Web 服务器的 Servlet 重定向器, 267
配置 (所有版本), 255
配置 MobiLink 客户端和服务器, 253
配置（适用于支持服务器组的重定向器）, 256
重置
MobiLink 上次下载时间, 123
CLASSPATH 环境变量
MobiLink Java 同步逻辑, 495
Clear 方法 [ML .NET]
DBParameterCollection 类语法, 577
Close 方法 [ML .NET]
DBCommand 语法, 569
DBConnection 语法, 570
DBRowReader 接口语法, 582
CLR
MobiLink 选项, 86
ColumnNames 属性 [ML .NET]
DBRowReader 接口语法, 582
ColumnTypes 属性 [ML .NET]
DBRowReader 接口语法, 582
CommandText 属性 [ML .NET]
DBCommand 语法, 569
Commit 方法 [ML .NET]
DBConnection 语法, 569
completed 属性
MobiLink 监控器同步统计信息, 181
conflicted_deletes 属性
MobiLink 监控器同步统计信息, 181
conflicted_inserts 属性
MobiLink 监控器同步统计信息, 181
conflicted_updates 属性
MobiLink 监控器同步统计信息, 181
从存储过程调用中下载结果集
同步技术, 153
connection_retries 属性
MobiLink 监控器同步统计信息, 181
Contains(object value) 方法 [ML .NET]
DBParameterCollection 类语法, 577
Contains(string parameterName) 方法 [ML .NET]
DBParameterCollection 类语法, 576
contd_timeout 协议选项
MobiLink 重定向器, 253
CopyTo(Array array, int index) 方法 [ML .NET]
DBParameterCollection 类语法, 579
Count 属性 [ML .NET]
DBParameterCollection 类语法, 580
CreateCommand 方法 [ML .NET]
DBConnection 语法, 570
参考数据库
术语定义, 774
参数
同步脚本, 299
参照完整性
术语定义, 774
操作员
监控器用户, 212
策略

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

术语定义, 774
插件模块
术语定义, 774
插入
MobiLink 中的脚本, 307
查看 MobiLink 日志
关于, 31
查询
术语定义, 774
查找详细信息并请求技术协助
技术支持, xix
程序集
在 MobiLink .NET 同步逻辑中定位, 553
在 MobiLink 中实施, 563
冲突
MobiLink, 137
MobiLink 冲突解决, 137
MobiLink 强制, 145
MobiLink 检测, 138
MobiLink 直接行处理, 615
MobiLink 缺省行为, 137
冲突处理
MobiLink 直接行处理, 615
冲突检测
MobiLink, 138
MobiLink 基于语句的上载, 138
冲突解决
MobiLink, 137
MobiLink 冲突检测, 138
MobiLink 检测, 138
MobiLink 缺省行为, 137
resolve_conflict 脚本, 140
upload_update 脚本, 142
在 MobiLink 中强制, 145
术语定义, 774
冲突是如何检测的
MobiLink, 138
抽取
术语定义, 774
出站启动器
中继服务器, 226
关于, 231
语法, 231
部署, 231
触发器
术语定义, 775
处理冲突
MobiLink, 137
处理错误
MobiLink 服务器, 414
处理单个 SQL 语句中的多个错误
MobiLink, 318
处理删除
MobiLink, 147
处理直接上载
MobiLink 直接行处理, 614
处理直接下载
MobiLink 直接行处理, 620
传输规则
术语定义, 775
窗口 (OLAP)
术语定义, 775
创建
MobiLink 统一数据库, 6
下载文件，用于 MobiLink 基于文件的下载, 278
文件定义数据库, 277
创建 Java 同步脚本
MobiLink Java 同步逻辑示例, 504
创建服务向导
MobiLink, 34
创建脚本版本向导
使用, 304
创建连接脚本向导
使用, 306
创建数据库
统一, 6
创建统一数据库
MobiLink 关于, 6
创建文件定义数据库
MobiLink, 277
创建下载文件
MobiLink 基于文件的下载, 278
创建者 ID
术语定义, 775
从 .NET 打印信息
MobiLink .NET 同步逻辑, 559
从自引用表上载数据
关于, 155
存储过程
MobiLink, 623
MobiLink 存储过程源代码, 307
术语定义, 775
用于下载数据, 153
用于添加或删除同步脚本, 307

错误
MobiLink modify_error_message 连接事件, 428
在 MobiLink 同步过程中处理, 317
提供反馈, xviii
记录, 317
错误处理
在 MobiLink 同步过程中, 317
错误日志
MobiLink 服务器 (mlsrv11), 61

D

daylight savings time
MobiLink, 123
DB2
MobiLink 隔离级别, 156
在 IBM 中标识符最大长度, 624, 656
针对 LUW 的 MobiLink 数据映射, 704
针对主机的 MobiLink 数据映射, 711
DB2 LUW
MobiLink 统一数据库, 12
DB2 主机
MobiLink 统一数据库, 15
在 IBM 中标识符的最大长度, 655
DBA 权限
术语定义, 776
DBCommand 接口 [ML .NET]
语法, 567
DBConnectionContext
构造函数, 556
DBConnectionContext 接口 [ML .NET]
语法, 570
DBConnectionContext 接口 [ML Java]
语法, 508
DBConnection 接口 [ML .NET]
语法, 569
dbmlsync 集成组件（不建议使用）
在 Windows 上部署, 763
dbmlsync 实用程序
在 Unix 上部署, 764
在 Windows 上部署, 763
部署, 763
DBMS
术语定义, 789
DBParameterCollection 方法 [ML .NET]
DBParameterCollection 类语法, 576
DBParameterCollection 类 [ML .NET]
语法, 576
DBParameter 类 [ML .NET]
语法, 573
DBRowReader 接口 [ML .NET]
语法, 581
dbspaces
术语定义, 776
DbType 属性 [ML .NET]
DBParameter 语法, 574
DCX
关于, xiv
DDL
术语定义, 788
Direction 属性 [ML .NET]
DBParameter 类语法, 574
DML
术语定义, 788
DocCommentXchange (DCX)
关于, xiv
download_bytes 属性
MobiLink 监控器同步统计信息, 181
download_cursor
MobiLink 无交集分区, 127
使用存储过程调用, 153
使用存储过程调用的示例, 153
关于, 313
基于时间戳的同步, 122
对子表进行分区, 129
性能, 162
有交集分区, 128
编写用于下载行的脚本, 312
表事件, 370
download_delete_cursor
使用存储过程调用, 153
使用存储过程调用的示例, 153
关于, 314
基于时间戳的同步, 121
对子表进行分区, 129
性能, 162
无交集分区, 127
有交集分区, 128
编写用于下载行的脚本, 312
表事件, 373
download_deleted_rows 属性
MobiLink 监控器同步统计信息, 181
download_errors 属性
MobiLink 监控器同步统计信息, 181
download_fetched_rows 属性

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

MobiLink 监控器同步统计信息, 181
download_filtered_rows 属性
MobiLink 监控器同步统计信息, 181
download_statistics
表事件, 379
连接事件, 376
download_timestamp
MobiLink 生成, 122
download_warnings 属性
MobiLink 监控器同步统计信息, 181
DownloadData 接口 [ML .NET]
语法, 582
DownloadData 接口 [ML Java]
语法, 513
DownloadTableData 接口 [ML .NET]
语法, 584
DownloadTableData 接口 [ML Java]
语法, 515
download 属性
MobiLink 监控器同步统计信息, 181
duration 属性
MobiLink 监控器同步统计信息, 181
代理 ID
术语定义, 775
代理 Web 服务器
MobiLink, 250
代理表
术语定义, 775
代码页
术语定义, 775
单向同步
关于, 130
电子表格
与 MobiLink 同步, 609
动态 SQL
术语定义, 776
度量
监控器, 200
编辑收集间隔, 207
断电
MobiLink 服务器, 30
对象
用于 .NET 的 MobiLink 服务器 API, 567
面向 Java 的 MobiLink 服务器 API, 508
对象树
术语定义, 776
对行进行分区
远程数据库之间的 MobiLink, 127
对子表进行分区
MobiLink, 129
多对多关系
分区, 128
同步, 128

E

EBF
术语定义, 776
ECC 协议选项
适用于 HTTPS 的 MobiLink 服务器 (mlsrv11) -x 选项, 105
适用于 TCP/IP 的 MobiLink 服务器 (mlsrv11) -x 选项, 103
end_connection
连接事件, 382
end_download
表事件, 386
连接事件, 384
end_download_deletes
表事件, 388
end_download_rows
表事件, 391
end_publication
连接事件, 394
end_synchronization
表事件, 399
连接事件, 397
end_sync 属性
MobiLink 监控器同步统计信息, 181
end_upload
表事件, 403
连接事件, 401
end_upload_deletes
表事件, 406
end_upload_rows
表事件, 408
ERROR [ML Java]
Java LogMessage 接口, 528
ERROR 字段 [ML .NET]
MessageType 枚举语法, 588
Excel
与 MobiLink 同步, 609
ExecuteNonQuery 方法 [ML .NET]
DBCommand 语法, 568
ExecuteReader 方法 [ML .NET]

DBCommand 语法, 568

F

FILE
术语定义, 777
file_authentication_code
authenticate_file_transfer 参数, 331
FILE 消息类型
术语定义, 777
FIPS
MobiLink 服务器 -x 选项, 102
使用 HTTPS 的 mlsrv11, 105
FIPS 协议选项
使用 TCP/IP 的 mlsrv10 -x 选项, 103
适用于 HTTPS 的 MobiLink 服务器 (mlsrv11) -x 选项, 105
FIPS 选项
MobiLink 服务器 (mlsrv11), 65
MobiLink 用户验证 (mluser), 650
FTP
MobiLink 基于文件的下载, 275
发布
术语定义, 776
发布更新
术语定义, 776
发布者
术语定义, 777
发送
监控器警告电子邮件, 217
发送电子邮件
监控器用户, 213
监控器警告通知, 217
反馈
报告错误, xviii
提供, xviii
文档, xviii
请求更新, xviii
反向代理
定义, 250
返回值
.NET 同步, 557
Java 同步, 499
方法
MobiLink .NET 同步逻辑, 557
MobiLink Java 同步逻辑, 499
防火墙
MobiLink 服务器, 253
路由 MobiLink 请求, 250
配置 MobiLink 客户端, 253
配置 MobiLink 服务器, 253
非关系数据库
与 MobiLink 同步, 609
非阻塞下载确认
MobiLink 服务器 (mlsrv11) -nba 选项, 70
nonblocking_download_ack 连接事件, 438
publication_nonblocking_download_ack 连接事件, 442
关于, 152
分区
MobiLink 无交集, 127
关于 MobiLink, 127
定义, 127
分析树
术语定义, 777
封锁
关于, 210
封装的下载
MobiLink 基于文件的下载, 275
服务
MobiLink, 34
MobiLink 服务器, 33
依赖性, 36
删除, 34
将 MobiLink 作为服务运行, 33
术语定义, 777
运行多个, 36
配置, 34
服务器
MobiLink 同步 [mlsrv11], 28
服务器管理请求
术语定义, 777
服务器监控
关于, 185
服务器启动的同步
术语定义, 777
服务器群
-zs 选项, 112
MobiLink, 37
中继服务器, 225
后端服务器, 226
负载平衡, 37
服务器系统过程
MobiLink, 623
服务器消息存储库

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

术语定义, 777
服务器组
MobiLink, 255
服务依赖性
MobiLink, 36
复制
术语定义, 777
复制代理
术语定义, 777
复制服务器
术语定义, 778
复制频率
术语定义, 778
复制消息
术语定义, 778
负载平衡
重定向器示例（适用于支持服务器组的重定向器）, 258
重定向器示例（适用于不支持服务器组的重定向器）, 260
MobiLink 服务器群, 37
MobiLink 重定向器, 250
负载平衡器
HTTP, 226

G

GetConnection 方法 [ML .NET]
DBConnectionContext 语法, 571
getConnection 方法 [ML Java]
DBConnectionContext 语法, 509
GetDeleteCommand 方法 [ML .NET]
DownloadTableData 接口语法, 585
getDeletePreparedStatement 方法 [ML Java]
DownloadTableData 语法, 517
GetDeletes 方法 [ML .NET]
UploadedTableData 接口语法, 604
getDeletes 方法 [ML Java]
UploadedTableData 语法, 546
GetDownloadData 方法 [ML .NET]
DBConnectionContext 语法, 571
getDownloadData 方法 [ML Java]
DBConnectionContext 语法, 509
GetDownloadTableByName 方法 [ML .NET]
DownloadData 接口语法, 584
getDownloadTableByName 方法 [ML Java]
DownloadData 语法, 514
GetDownloadTables 方法 [ML .NET]
DownloadData 接口语法, 583
getDownloadTables 方法 [ML Java]
DownloadData 语法, 514
GetEnumerator 方法 [ML .NET]
DBParameterCollection 类语法, 579
GetInserts 方法 [ML .NET]
UploadedTableData 接口语法, 605
getInserts 方法 [ML Java]
UploadedTableData 语法, 547
GetLastDownloadTime 方法 [ML .NET]
DownloadTableData 接口语法, 586
getLastDownloadTime 方法 [ML Java]
DownloadTableData 语法, 521
getMetaData 方法 [ML Java]
DownloadTableData 语法, 520
UploadedTableData 语法, 549
GetName 方法 [ML .NET]
DownloadTableData 接口语法, 586
UploadedTableData 接口语法, 606
getName 方法 [ML Java]
DownloadTableData 语法, 519
UploadedTableData 语法, 549
getPropertiesByVersion 方法 [ML .NET]
ServerContext 接口语法, 592
getPropertiesByVersion 方法 [ML Java]
ServerContext 语法, 534
getProperties 方法 [ML .NET]
ServerContext 接口语法, 591
GetProperties 方法 [ML .NET]
DBConnectionContext 语法, 572
getProperties 方法 [ML Java]
DBConnectionContext 语法, 510
ServerContext 语法, 533
getPropertySetNames 方法 [ML .NET]
ServerContext 接口语法, 592
getPropertySetNames 方法 [ML Java]
ServerContext 语法, 534
GetRemoteID 方法 [ML .NET]
DBConnectionContext 语法, 572
getRemoteID 方法 [ML Java]
DBConnectionContext 语法, 511
GetSchemaTable 方法 [ML .NET]
DownloadTableData 接口语法, 586
UploadedTableData 接口语法, 607
GetServerContext 方法 [ML .NET]
DBConnectionContext 语法, 572
getServerContext 方法 [ML Java]

DBConnectionContext 语法, 511
GetStartClassInstances 方法 [ML .NET]
ServerContext 接口语法, 590
getStartClassInstances 方法 [ML Java]
ServerContext 语法, 535
getText 方法 [ML Java]
LogMessage 语法, 529
getType 方法 [ML Java]
LogMessage 语法, 529
GetUpdates 方法 [ML .NET]
UploadedTableData 接口语法, 607
getUpdates 方法 [ML Java]
UploadedTableData 语法, 548
GetUploadedTableByName 方法 [ML .NET]
UploadData 接口语法, 602
getUploadedTableByName 方法 [ML Java]
UploadData 语法, 544
GetUploadedTables 方法 [ML .NET]
UploadData 接口语法, 603
getUploadedTables 方法 [ML Java]
UploadData 语法, 545
GetUpsertCommand 方法 [ML .NET]
DownloadTableData 接口语法, 587
getUpsertPreparedStatement 方法 [ML Java]
DownloadTableData 语法, 518
getUser 方法 [ML Java]
LogMessage 语法, 529
getValue 方法 [ML Java]
InOutInteger 语法, 522
InOutString 语法, 524
GetVersion 方法 [ML .NET]
DBConnectionContext 语法, 572
getVersion 方法 [ML Java]
DBConnectionContext 语法, 512
global_database_id 选项
在 MobiLink 中设置, 133
GUID
(*参见* UUID)
过程调用
SQL Server MobiLink 统一数据库, 19
高可用性
MobiLink 重定向器, 250
隔离级别
MobiLink, 156
术语定义, 778
个人服务器
术语定义, 778
更改上次下载时间
MobiLink, 123
工具
MobiLink 监控器选取框工具, 176
工作表
术语定义, 778
工作线程
MobiLink, 163
MobiLink 性能, 160
公共语言运行库
MobiLink 选项, 86
共享程序集
在 MobiLink 中实施, 563
共享服务器状态
-zs 选项, 112
共享规则 (*见* 分区)
共享状态
MobiLink 服务器群, 37
构造函数
MobiLink .NET 同步逻辑, 556
MobiLink Java 同步逻辑, 498
故障排除
MobiLink 服务器启动, 38
MobiLink 服务器日志, 31
MobiLink 可重新启动的下载, 149
同步错误, 61
处理失败的下载, 149
故障切换
术语定义, 778
故障转移
MobiLink 服务器群, 37
MobiLink 重定向器, 250
挂接
(*参见* 事件挂接)
关闭
MobiLink 停止实用程序 (mlstop), 649
MobiLink 服务器, 30
关键连接
术语定义, 788
管理员
监控器用户, 212
管理员用户
监控器关于, 212
广播下载
MobiLink 基于文件的下载, 275
规范化
术语定义, 779

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

归类
术语定义, 779
归类序列
MobiLink 同步, 740
过程
MobiLink, 623
过程返回结果
Oracle 驱动程序选项, 745

H

handle_DownloadData
连接事件, 410
handle_error
同步脚本, 317
连接事件, 414
handle_odbc_error
连接事件, 418
handle_UploadData
连接事件, 422
host 协议选项
MobiLink 重定向器, 253
适用于 HTTP 的 MobiLink 服务器 (mlsrv11) -x 选项, 104
适用于 HTTPS 的 MobiLink 服务器 (mlsrv11) -x 选项, 105
适用于 TCP/IP 的 MobiLink 服务器 (mlsrv11) -x 选项, 102
适用于通过 TCP/IP 的 TLS 的 MobiLink 服务器 (mlsrv11) -x 选项, 103
HotSync
在 Windows 上部署 MobiLink 客户端, 763
HTTP
mlsrv11 -x 选项, 104
MobiLink 服务器 -x 选项, 102
httpd.conf
Apache 本机重定向器, 269
HTTPS
mlsrv11 -x 选项, 105
MobiLink 服务器 -x 选项, 102
HTTP 负载平衡器
中继服务器, 226
后端服务器部分
中继服务器配置文件, 228
后端群部分
中继服务器配置文件, 228
环境变量
命令 shell, xvii
命令提示符, xvii
恢复失败的下载
MobiLink, 149
回退日志
术语定义, 779
会话范围的变量
DB2 MobiLink 统一数据库, 13
Oracle MobiLink 统一数据库, 23
会话属性
MobiLink 监控器, 176
获取帮助
技术支持, xviii

I

iAnywhere JDBC 驱动程序
术语定义, 779
iAnywhere Solutions ODBC 驱动程序
支持, 743
iAnywhere Solutions Oracle 驱动程序
关于, 745
iAnywhere 开发人员社区
新闻组, xix
iaredirect.dll
在 Unix 上配置 NSAPI 重定向器, 263
在 Windows 上配置 NSAPI 重定向器, 261
配置 ISAPI 重定向器, 265
iaredirect.so
在 Unix 上配置 NSAPI 重定向器, 263
在 Windows 上配置 NSAPI 重定向器, 261
IBM DB2
DB2 LUW 作为 MobiLink 统一数据库, 12
标识符最大长度, 624, 656
针对 LUW 的 MobiLink 数据映射, 704
针对主机的 MobiLink 数据映射, 711
IBM DB2 LUW
MobiLink 统一数据库, 12
IBM DB2 LUW 统一数据库
MobiLink, 12
IBM DB2 主机
MobiLink 统一数据库, 15
标识符最大长度, 655
IBM DB2 主机统一数据库
MobiLink, 15
IBM DB2 主机系统表名称转换
ml_active_rid, 656, 657
ml_conn_script, 656, 659
ml_pt, 656, 665, 686

ml_pt_repair, 656, 666
ml_pt_script, 656, 667
ml_pt_status, 656, 669
ml_script_modified, 656, 686
identity_password 协议选项
适用于 HTTPS 的 MobiLink 服务器 (mlsrv11) -x 选项, 105
identity 协议选项
适用于 HTTPS 的 MobiLink 服务器 (mlsrv11) -x 选项, 105
identity 选项
适用于 HTTPS 的 MobiLink 服务器 (mlsrv11) -x 选项, 105
ignored_deletes 属性
MobiLink 监控器同步统计信息, 181
ignored_inserts 属性
MobiLink 监控器同步统计信息, 181
ignored_updates 属性
MobiLink 监控器同步统计信息, 181
ignore 协议选项
适用于 TCP/IP 的 MobiLink 服务器 (mlsrv11) -x 选项, 102
适用于通过 TCP/IP 的 TLS 的 MobiLink 服务器 (mlsrv11) -x 选项, 103
IIS
为 ISAPI 配置, 265
IndexOf(object value) 方法 [ML .NET]
DBParameterCollection 类语法, 578
IndexOf(string parameterName) 方法 [ML .NET]
DBParameterCollection 类语法, 576
INFO [ML Java]
Java LogMessage 接口, 528
InfoMaker
术语定义, 779
INFO 字段 [ML .NET]
MessageType 枚举语法, 588
InOutInteger 接口 [ML Java]
语法, 521
InOutString 接口 [ML Java]
语法, 523
Insert(int index, object value) 方法 [ML .NET]
DBParameterCollection 类语法, 578
install-dir
文档用法, xvi
Interactive SQL
术语定义, 780
iPlanet
在 Unix 上为 NSAPI 重定向器进行配置, 263
在 Windows 上为 NSAPI 重定向器进行配置, 261
ISAPI 重定向器
调用, 265
ISAPI 重定向器
配置, 265
IsFixedSize 属性 [ML .NET]
DBParameterCollection 类语法, 579
IsNullable 属性 [ML .NET]
DBParameter 类语法, 574
IsReadOnly 属性 [ML .NET]
DBParameterCollection 类语法, 580
IsSynchronized 属性 [ML .NET]
DBParameterCollection 类语法, 580

J

JAR 文件
术语定义, 780
Java
MobiLink 同步脚本, 493
MobiLink 基于对象的数据流, 609
MobiLink 数据类型, 498
MobiLink 服务器 API 参考, 508
Javadoc
MobiLink, 508
Java MobiLink 服务器 API (*见* 用于 Java 的 MobiLink 服务器 API)
Java VM
MobiLink 选项, 87
Java 类
Java 同步逻辑实例化, 497
术语定义, 780
Java 同步
MobiLink Java 同步逻辑, 504
Java 同步逻辑
Java 类实例化, 497
MobiLink 性能, 162
MobiLink 服务器 API, 508
在 32 位 Unix 上部署, 757
在 32 位 Windows 上部署, 751
在 64 位 Unix 上部署, 760
在 64 位 Windows 上部署, 754
在 MobiLink 服务器命令行中指定, 495
方法, 499
示例, 504
设置, 495

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

Java 与 SQL 同步逻辑
MobiLink 性能, 162
jConnect
术语定义, 780
JDBC
术语定义, 780
校验
MobiLink 基于文件的下载, 280
MobiLink 自动, 280
MobiLink 自定义, 282
术语定义, 782
校验和
术语定义, 782
校验检查
MobiLink 基于文件的下载, 280
基表
术语定义, 780
基础规则
MobiLink, 120
基于 SQL 的同步
术语定义, 781
基于对象的数据流 (*见* 直接行处理)
基于会话的同步
术语定义, 780
基于脚本的上载
术语定义, 781
基于时间戳的同步
download_cursor 脚本, 122
download_delete_cursor 脚本, 121
关于, 121
基于时间戳的下载
关于, 121
基于文件的下载
关于, 275
术语定义, 781
示例, 283
基于语句的脚本
上载行, 309
基于语句的上载
冲突检测, 138
集成登录
术语定义, 781
技术支持
新闻组, xix
记录
MobiLink 性能, 161
MobiLink 服务器操作, 31
记录 MobiLink 服务器操作
关于, 31
加密口令
Oracle 驱动程序选项, 745
监控
MobiLink 中的同步, 167
MobiLink 性能, 166
MobiLink 服务器, 185
监控 MobiLink 性能
概述, 166
监控器
[度量] 选项卡, 201
MobiLink 监控器, 167
使用封锁停止监控, 210
使用户与资源相关联, 213
停止监控器, 196
停止监控资源, 209
关于, 185
创建用户, 212
删除度量, 204
删除用户, 214
删除警告, 217
删除资源, 210
启用电子邮件通知, 218
在单独的计算机上启动, 195
在单独的计算机上安装, 220
在生产环境中运行, 186
安全, 214
封锁, 210
度量, 200
开始监控资源, 206
快速入门, 189
手工停止监控, 209
收集间隔, 206
教程, 190
断开连接, 198
本地启动, 195
查看警告, 216
添加资源, 206
状态, 200
用户类型, 212
电子邮件通知, 217
疑难解答, 221
监控资源, 206
管理员用户, 212
编辑用户, 213
网络配置, 187

要求, 186
要求登录, 214
警告, 216
警告错误报告, 219
资源, 199, 206
连接, 197
退出, 196
选项卡, 201
错误报告, 219
限制, 186
监控器度量
[同步] 选项卡, 203
[度量] 选项卡, 201
[统一数据库] 选项卡, 203
[警告] 选项卡, 202
[计算机资源] 选项卡, 204
删除, 204
当在指定时间内发生相同情况时取消发出警告, 208
指定要收集的度量, 207
收集间隔, 206
服务器选项卡, 202
警告, 208
监听器
术语定义, 781
监听器实用程序
在 Windows 上部署 MobiLink 客户端, 763
检测冲突
MobiLink, 138
MobiLink 使用 upload_fetch 脚本, 138
MobiLink 使用 upload_update 脚本, 139
检查点
术语定义, 781
键池
MobiLink 同步应用程序, 134
健康和统计信息
监控器, 185
建立
MobiLink 基于文件的下载, 277
MobiLink 统一数据库, 6
建立 Oracle 统一数据库
MobiLink, 23
建立 SQL Anywhere 统一数据库
MobiLink, 26
建立 Sybase ASE 统一数据库
MobiLink, 10
建立线程
MobiLink 性能, 160
将表分区
示例, 127
脚本
MobiLink ml_active_remote_id 系统表, 657
MobiLink ml_column 系统表, 658
MobiLink ml_connection_script 系统表, 659
MobiLink ml_database 系统表, 660
MobiLink ml_device 系统表, 661
MobiLink ml_device_address 系统表, 662
MobiLink ml_listening 系统表, 663
MobiLink ml_passthrough 系统表, 665
MobiLink ml_passthrough_repair 系统表, 666
MobiLink ml_passthrough_script 系统表, 667
MobiLink ml_passthrough_status 系统表, 669
MobiLink ml_property 系统表, 670
MobiLink ml_script 系统表, 684
MobiLink ml_script_version 系统表, 685
MobiLink ml_scripts_modified 系统表, 686
MobiLink ml_server 系统表, 687
MobiLink ml_sis_sync_state 系统表, 688
MobiLink ml_subscription 系统表, 689
MobiLink ml_table_script 系统表, 692
MobiLink ml_user 系统表, 693
MobiLink 事件, 320
MobiLink 事件概述, 321
MobiLink 需要, 305
全局脚本版本, 303
关于 MobiLink, 293
支持的 DBMS 脚本策略, 8
术语定义, 781
添加到 MobiLink 中的统一数据库, 306
添加和删除 .NET 表脚本, 629
添加和删除 .NET 连接脚本, 628
添加和删除 Java 表脚本, 631
添加和删除 Java 连接脚本, 630
添加和删除 SQL 表脚本, 639
添加和删除 SQL 连接脚本, 627
版本, 303
编写用于上载行的脚本, 309
编写用于下载行的脚本, 312
表脚本, 298
连接脚本, 298
脚本版本
保留名称, 303
全局, 303
有关 MobiLink 同步, 303

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

术语定义, 782
添加, 304
脚本参数
last_download, 122
last_table_download, 122
关于 MobiLink, 299
脚本的直接插入
MobiLink, 307
脚本类型
MobiLink, 298
角色
术语定义, 782
角色名
术语定义, 782
教程
监控器, 190
结束
MobiLink 服务器, 30
解决
MobiLink 冲突, 137
MobiLink 冲突解决, 137
解决冲突
MobiLink 使用 resolve_conflict 脚本, 140
MobiLink 使用 upload_update 脚本, 142
MobiLink 概述, 140
resolve_conflict 脚本, 140
upload_update 脚本, 142
仅上载同步
关于, 130
必需的脚本, 305
仅上载同步和仅下载同步
关于, 130
仅下载同步
关于, 130
必需的脚本, 305
进度
ml_subscription 表的 progress 列, 689
进度计数器
ml_subscription 表的 progress 列, 689
进度偏移
ml_subscription 表的 progress 列, 689
警告
监控器, 216
监控器取消, 219
监控器电子邮件通知, 217
镜像日志
术语定义, 782
旧的行参数
MobiLink 脚本, 299
局部临时表
术语定义, 782

K

可重新启动的下载
MobiLink, 149
keep partial download 同步参数
可重新启动的下载, 150
开发人员社区
新闻组, xix
开发提示
MobiLink 同步, 120
Mobilink 直接行处理, 612
开关
MobiLink 服务器 (mlsrv11), 41
MobiLink 用户验证 (mluser), 650
抗锯齿功能
MobiLink 监控器选项, 174
可分配的下载
MobiLink 基于文件的下载, 275
客户端
连接到中继服务器群, 245
客户端/服务器
术语定义, 782
客户端事件挂接过程
(*参见* 事件挂接)
客户端消息存储库
术语定义, 782
客户端消息存储库 ID
术语定义, 783
空字符串
Oracle MobiLink 统一数据库, 23
Oracle 不支持, 23
口令
MobiLink mluser 实用程序, 650
在 Oracle 驱动程序中加密, 745
监控器用户, 212
快速入门
MobiLink 直接行处理, 611
快照隔离
MobiLink, 156
MobiLink -dsd 选项禁用, 59
MobiLink -esu 选项为上载启用, 62
术语定义, 783
用于 SQL Server 的 MobiLink -dt 选项, 60

快照同步
关于, 125

L

last_download
MobiLink 命名参数, 122
modify_last_download_timestamp 连接事件, 430
last_download_timestamp
MobiLink 命名参数, 122
MobiLink 生成, 122
last_table_download
MobiLink 命名参数, 122
modify_last_download_timestamp 连接事件, 430
Linux 上的 Apache
部署中继服务器, 239
LOG_LEVEL
重定向器属性（适用于不支持服务器组的重定向器）, 259
重定向器属性（适用于支持服务器组的重定向器）, 257
LogCallback ErrorListener 事件 [ML .NET]
ServerContext 接口语法, 590
LogCallback InfoListener 事件 [ML .NET]
ServerContext 接口语法, 590
LogCallback WarningListener 事件 [ML .NET]
ServerContext 接口, 590
LogCallback 委派 [ML .NET]
DBRowReader 接口语法, 587
LogListener 接口 [ML Java]
语法, 524
LogMessage 类 [ML .NET]
语法, 587
LogMessage 类 [ML Java]
语法, 525
LONG 数据类型
Oracle 同步, 736
LTM
术语定义, 784
LUW
DB2 LUW 作为 MobiLink 统一数据库, 12
类实例
Java 同步逻辑, 497
MobiLink .NET 同步逻辑, 555
联机手册
PDF, xiv
连接
MobiLink mlsrv11 -c 选项, 50
MobiLink 服务器 -x 选项, 102
术语定义, 783
版本 10 以前的 MobiLink 客户端-服务器, 107
连接 ID
术语定义, 783
连接参数
MobiLink 服务器 -x 选项, 102
连接级脚本
定义, 298
连接脚本
ml_global, 303
使用 Sybase Central 添加, 306
关于, 298
删除 .NET 脚本, 628
删除 Java 脚本, 630
删除 SQL 脚本, 627
定义, 298
按字母顺序排列的 MobiLink 脚本列表, 320
添加 .NET 脚本, 628
添加 Java 脚本, 630
添加 SQL 脚本, 627
连接类型
术语定义, 783
连接配置
术语定义, 783
连接启动的同步
术语定义, 783
连接属性
MobiLink 服务器 -x 选项, 102
连接条件
术语定义, 783
连接字符串
MobiLink mlsrv11, 50
列大小
ASE MobiLink 统一数据库, 10
临时表
术语定义, 783
路由请求
MobiLink 同步, 249
轮询
术语定义, 784
逻辑删除
编写 download_delete_cursor 脚本, 314
逻辑索引
术语定义, 784

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

M

magnus.conf
  在 Unix 上为 NSAPI 重定向器进行配置, 263
  在 Windows 上为 NSAPI 重定向器进行配置, 261
MakeConnection 方法 [ML .NET]
  ServerContext 接口语法, 590
makeConnection 方法 [ML Java]
  ServerContext 语法, 535
Manage Anywhere
  MobiLink 基于文件的下载, 275
M-Business Anywhere
  重定向器, 271
  为同步配置, 271
M-Business Anywhere 重定向器
  配置, 271
messageLogged 方法 [ML Java]
  LogListener 语法, 525
MessageType 枚举 [ML .NET]
  语法, 588
Microsoft Excel
  与 MobiLink 同步, 609
Microsoft SQL Server
  MobiLink 数据映射, 720
  MobiLink 隔离级别, 156
  作为 MobiLink 统一数据库, 19
Microsoft SQL Server 统一数据库
  MobiLink, 19
Microsoft 分布式事务协调器
  Oracle 驱动程序选项, 745
ML
  重定向器属性（适用于不支持服务器组的重定向器）, 259
  重定向器属性（适用于支持服务器组的重定向器）, 257
ml_active_remote_id
  MobiLink 系统表, 657
ml_active_rid
  IBM DB2 主机系统表名称转换, 656, 657
ml_add_column 系统过程
  语法, 626
ml_add_connection_script 系统过程
  语法, 627
ml_add_cs 系统过程
  IBM DB2 主机系统过程名称转换, 627
  IBM DB2 转换, 624
ml_add_dcs 系统过程
  IBM DB2 主机系统过程名称转换, 628
  IBM DB2 转换, 624
ml_add_dnet_connection_script 系统过程
  语法, 628
ml_add_dnet_table_script 系统过程
  语法, 629
ml_add_dts 系统过程
  IBM DB2 主机系统过程名称转换, 629
  IBM DB2 转换, 624
ml_add_java_connection_script 系统过程
  语法, 630
ml_add_java_table_script 系统过程
  语法, 631
ml_add_jcs 系统过程
  IBM DB2 主机系统过程名称转换, 630
  IBM DB2 转换, 624
ml_add_jts 系统过程
  IBM DB2 主机系统过程名称转换, 631
  IBM DB2 转换, 624
ml_add_lang_connection_script_chk 系统过程
  语法, 633
ml_add_lang_connection_script 系统过程
  语法, 633
ml_add_lang_table_script_chk 系统过程
  语法, 633
ml_add_lang_table_script 系统过程
  语法, 633
ml_add_lcs_chk 系统过程
  IBM DB2 转换, 624
  语法, 633
ml_add_lcs 系统过程
  IBM DB2 转换, 624
  语法, 633
ml_add_lts_chk 系统过程
  IBM DB2 转换, 624
  语法, 633
ml_add_lts 系统过程
  IBM DB2 转换, 624
  语法, 633
ml_add_passthrough_repair 系统过程
  语法, 634
ml_add_passthrough_script 系统过程
  语法, 635
ml_add_passthrough 系统过程
  语法, 633
ml_add_property 系统过程

语法, 637
ml_add_pt_repair 系统过程
IBM DB2 主机系统过程名称转换, 634
IBM DB2 转换, 624
ml_add_pt_script 系统过程
IBM DB2 主机系统过程名称转换, 635
IBM DB2 转换, 624
ml_add_pt 系统过程
IBM DB2 主机系统过程名称转换, 633
IBM DB2 转换, 624
ml_add_table_script 系统过程
语法, 639
ml_add_ts 系统过程
IBM DB2 主机系统过程名称转换, 639
IBM DB2 转换, 624
ml_add_user 系统过程
语法, 640
ML_CLIENT_TIMEOUT
重定向器属性（适用于不支持服务器组的重定向器）, 259
重定向器属性（适用于支持服务器组的重定向器）, 257
ml_column
MobiLink 系统表, 658
ml_conn_script
IBM DB2 主机系统表名称转换, 656, 659
ml_connection_script
MobiLink 系统表, 659
ml_database
MobiLink 系统表, 660
ml_del_pt_repair 系统过程
IBM DB2 主机系统过程名称转换, 641
IBM DB2 转换, 624
ml_del_pt_script 系统过程
IBM DB2 主机系统过程名称转换, 642
IBM DB2 转换, 624
ml_del_pt 系统过程
IBM DB2 主机系统过程名称转换, 641
IBM DB2 转换, 624
ml_del_sstate_b4 系统过程
IBM DB2 主机系统过程名称转换, 644
IBM DB2 转换, 624
ml_del_sstate 系统过程
IBM DB2 主机系统过程名称转换, 643
IBM DB2 转换, 624
ml_delete_passthrough_repair 系统过程
语法, 641
ml_delete_passthrough_script 系统过程
语法, 642
ml_delete_passthrough 系统过程
语法, 641
ml_delete_sync_state_before 系统过程
语法, 644
ml_delete_sync_state 系统过程
语法, 643
ml_delete_user 系统过程
语法, 645
ml_device
MobiLink 系统表, 661
ml_device_address
MobiLink 系统表, 662
ml_global 脚本版本
关于, 303
ml_listening
MobiLink 系统表, 663
ml_passthrough
MobiLink 系统表, 665
ml_passthrough_repair
MobiLink 系统表, 666
ml_passthrough_script
MobiLink 系统表, 667
ml_passthrough_status
MobiLink 系统表, 669
ml_property
MobiLink 系统表, 670
ml_pt
IBM DB2 主机系统表名称转换, 656, 665
ml_pt_repair
IBM DB2 主机系统表名称转换, 656, 666
ml_pt_script
IBM DB2 主机系统表名称转换, 656, 667
ml_pt_status
IBM DB2 主机系统表名称转换, 656, 669
ml_qa_clients
QAnywhere 客户端系统表, 671
ml_qa_delivery
QAnywhere 客户端系统表, 672
ml_qa_delivery_archive
QAnywhere 客户端系统表, 673
ml_qa_global_props
QAnywhere 客户端系统表, 674
ml_qa_notifications
QAnywhere 客户端系统表, 675
ml_qa_repository

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

QAnywhere 客户端系统表, 676
ml_qa_repository_archive
QAnywhere 客户端系统表, 677
ml_qa_repository_props
QAnywhere 客户端系统表, 678
ml_qa_repository_props_archive
QAnywhere 客户端系统表, 679
ml_qa_repository_staging
QAnywhere 客户端系统表, 680
ml_qa_status_history
QAnywhere 客户端系统表, 681
ml_qa_status_history_archive
QAnywhere 客户端系统表, 682
ml_qa_status_staging
QAnywhere 客户端系统表, 683
ml_reset_sstate 系统过程
IBM DB2 主机系统过程名称转换, 645
IBM DB2 转换, 624
ml_reset_sync_state 系统过程
语法, 645
ml_script
MobiLink 系统表, 684
ml_script_modified
IBM DB2 主机系统表名称转换, 656, 686
ml_script_version
MobiLink 系统表, 685
ml_scripts_modified
MobiLink 系统表, 686
ml_server
MobiLink 系统表, 687
ml_server_delete 系统过程
语法, 646
ml_server_update 系统过程
语法, 646
ml_set_sis_state 系统过程
IBM DB2 转换, 624
ml_sis_sync_state
MobiLink 系统表, 688
ml_subscription
MobiLink 系统表, 689
ml_table
MobiLink 系统表, 691
ml_table_script
MobiLink 系统表, 692
ml_user
MobiLink 用户验证 (mluser), 650
MobiLink 系统表, 693
mlDomConfig.xml
关于, 564
mlmon
关于 MobiLink 监控器, 167
开始, 169
mlMonitorSettings
MobiLink 监控器设置, 176
mlscript.jar
MobiLink Java 同步逻辑, 495
mlsrv11
(*参见* MobiLink 服务器)
-nc 选项, 71
QAnywhere, 69
停止, 30
启动, 28
在消息日志中报告错误上下文, 73
记录, 31
语法, 41
连接字符串, 50
选项, 41
通告程序, 72
mlsrv11 选项
按字母顺序排序的列表, 41
mlsrv11 语法
关于, 41
mlstop 实用程序
停止 MobiLink 服务器的方法, 30
在 32 位 Unix 上部署, 757
在 32 位 Windows 上部署, 751
在 64 位 Unix 上部署, 760
在 64 位 Windows 上部署, 754
语法, 649
选项, 649
mluser 实用程序
在 32 位 Unix 上部署, 757
在 32 位 Windows 上部署, 751
在 64 位 Unix 上部署, 760
在 64 位 Windows 上部署, 754
语法, 650
选项, 650
MobiLink
.NET 同步逻辑, 551
重定向器, 249
Java 同步逻辑, 493
mlsrv11 的连接参数, 102
mlsrv11 选项, 40
ODBC 驱动程序支持, 744

Web 服务器配置, 249
事件概述, 321
停止 MobiLink 服务器, 30
同步技术, 119
启动, 28
在当前会话外运行, 33
基于文件的下载, 275
处理冲突, 137
多个同步服务器, 251
字符集注意事项, 739
开发提示, 120
性能, 159
按字母顺序排列的事件列表, 320
数据类型, 695
术语定义, 784
监控器, 167
监控器连接参数, 169
系统表, 654
系统过程, 623
统一数据库, 3
脚本, 293
运行同步服务器, 27
部署应用程序, 749
MobiLink 存储过程 (*见* MobiLink 系统过程)
MobiLink 服务器
(*参见* mlsrv11)
停止实用程序, 649
启动, 28
术语定义, 784
监控, 185
语法, 41
运行, 27
选项, 41
部署, 751
MobiLink 服务器共享状态
服务器群, 37
MobiLink 服务器监控
关于, 185
MobiLink 服务器群
-lsc 选项, 68
故障转移, 37
负载平衡, 37
MobiLink 服务器日志文件查看器
MobiLink 服务器日志, 32
MobiLink 服务器选项
关于, 40
MobiLink 服务器组
关于, 255
MobiLink 监控器
[一览表] 窗格, 175
[图表] 窗格, 174
[详细信息表] 窗格, 171
[选取框] 工具, 175
会话属性, 176
使用, 171
保存数据, 179
关于, 167
启动, 169
图形窗格, 172
在 32 位 Unix 上部署, 757
在 32 位 Windows 上部署, 751
在 64 位 Unix 上部署, 760
在 64 位 Windows 上部署, 754
在 MS Excel 中查看, 179
恢复缺省设置, 176
指定监视, 180
术语定义, 784
用户界面, 171
监视项目管理器, 180
示例属性, 177
统计信息属性, 181
缩放, 174
选项, 176
MobiLink 脚本
列出的, 320
MobiLink 客户端
术语定义, 785
部署, 763
MobiLink 连接
调试, 38
MobiLink 日志文件查看器
MobiLink 服务器日志, 32
MobiLink 实用程序
MobiLink 停止实用程序 (mlstop), 649
MobiLink 用户验证 (mluser), 650
关于, 647
服务器, 647
MobiLink 世代号
基于文件的下载, 281
MobiLink 事件
列出的, 320
MobiLink 事件序列
伪代码, 324
MobiLink 数据类型

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

.NET 和 SQL, 556
Java 和 SQL, 498
MobiLink 数据映射
关于, 695
MobiLink 停止实用程序 (mlstop)
语法, 649
MobiLink 同步
.NET 同步逻辑, 551
Java 同步逻辑, 493
可重新启动的下载, 149
Web 服务器配置, 249
事件概述, 321
基于文件的下载, 275
性能, 159
统一数据库, 3
编写 .NET 类, 557
编写 Java 类, 499
MobiLink 同步服务器 (*见* MobiLink 服务器)
MobiLink 同步脚本
调试 Java 类, 499
关于, 293
在 .NET 中保留数据库事务, 555
在 Java 中保留数据库事务, 497
按字母顺序排列的脚本列表, 320
数据库事务和 .NET 类, 555
数据库事务和 Java 类, 497
构造 .NET 类, 556
构造 Java 类, 498
编写 Java 类, 499, 557
MobiLink 同步逻辑
.NET, 551
.NET 和 SQL 的数据类型, 556
Java, 493
Java 和 SQL 数据类型, 498
同步技术, 119
按字母顺序排列的脚本列表, 320
编写脚本, 293
MobiLink 统计信息属性
MobiLink 监控器, 181
MobiLink 统一数据库
ASE, 10
IBM DB2 LUW 作为, 12
IBM DB2 主机作为, 15
MySQL 作为, 21
Oracle, 23
SQL Anywhere 作为, 26
SQL Server 作为, 19
关于, 3
MobiLink 文件传输实用程序 (mlfiletransfer)
mlsrv11 -ftr 选项, 67
MobiLink 系统表
ml_active_remote_id, 657
ml_column, 658
ml_connection_script, 659
ml_database, 660
ml_device, 661
ml_device_address, 662
ml_listening, 663
ml_passthrough, 665
ml_passthrough_repair, 666
ml_passthrough_script, 667
ml_passthrough_status, 669
ml_property, 670
ml_script, 684
ml_script_version, 685
ml_scripts_modified, 686
ml_server, 687
ml_sis_sync_state, 688
ml_subscription, 689
ml_table, 691
ml_table_script, 692
ml_user, 693
关于, 654
在统一数据库中创建, 6
术语定义, 785
MobiLink 系统过程
关于, 623
MobiLink 系统数据库
关于, 7
MobiLink 性能
关于, 159
关键因素, 163
监控, 166
MobiLink 用户
ml_user 系统表, 693
MobiLink 用户验证 (mluser), 650
向 mluser 实用程序注册, 650
术语定义, 785
MobiLink 用户验证实用程序 (mluser)
语法, 650
MobiLink 支持的 ODBC 驱动程序
关于, 744
MobiLink 中的行处理 (*见* 直接行处理)
mod_iaredirect.dll

配置 Apache 重定向器, 269
配置 M-Business Anywhere 重定向器, 271
mod_iaredirect.so
配置 M-Business Anywhere 重定向器, 271
modify_error_message
连接事件, 428
modify_last_download_timestamp
连接事件, 430
modify_next_last_download_timestamp
连接事件, 433
modify_user
连接事件, 436
MySQL
MobiLink 数据映射, 726
MobiLink 统一数据库, 21
MySQL 统一数据库
MobiLink, 21
面向 .NET 的 MobiLink 服务器 API
ml_property 系统表, 670
面向 Java 的 MobiLink 服务器 API
ml_property 系统表, 670
命令 shell
大括号, xvii
引号, xvii
括号, xvii
环境变量, xvii
约定, xvii
命令提示符
大括号, xvii
引号, xvii
括号, xvii
环境变量, xvii
约定, xvii
命令文件
术语定义, 784
命令行
启动 mlsrv11, 41
命令行实用程序
MobiLink 停止实用程序 (mlstop), 649
MobiLink 同步, 647
MobiLink 用户验证 (mluser), 650
命名参数
last_download, 122
last_table_download, 122
关于 MobiLink, 299
命名的脚本参数
关于 MobiLink, 299
命名的行参数
关于 MobiLink 脚本, 299
命名脚本参数
ml_add_column 系统过程, 626
命名行参数
将列信息添加到统一数据库中, 626
模式
将统一表与 MobiLink 远程表相关联, 5
术语定义, 785

N

Netscape Web 服务器
在 Unix 上配置 NSAPI 重定向器, 263
在 Windows 上配置 NSAPI 重定向器, 261
NextRow 方法 [ML .NET]
DBRowReader 接口语法, 582
nonblocking_download_ack
连接事件, 438
NSAPI 重定向器
在 Unix 上配置, 263
在 Windows 上配置, 261
内连接
术语定义, 785

O

o.
MobiLink 命名参数前缀, 299
obj.conf
在 Unix 上为 NSAPI 重定向器进行配置, 263
在 Windows 上为 NSAPI 重定向器进行配置, 261
ODBC
MobiLink 中的多个错误, 318
MobiLink 驱动程序, 744
Oracle 驱动程序, 745
术语定义, 785
ODBC 管理器
术语定义, 785
ODBC 驱动程序
MobiLink 字符集转换，通过, 741
MobiLink 支持的, 744
Oracle, 745
ODBC 数据源
术语定义, 785
Oracle
MobiLink 同步中的序列, 23
MobiLink 数据映射, 731

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

MobiLink 隔离级别, 156
ODBC 驱动程序, 745
作为 MobiLink 统一数据库, 23
同步 LONG 型数据, 736
Oracle varray
在存储过程中使用, 24
示例, 24
限制, 25
Oracle 驱动程序
ODBC, 745
加密口令, 745
Oracle 统一数据库
MobiLink, 23

P

ParameterName 属性 [ML .NET]
DBParameter 类语法, 574
Parameters 属性 [ML .NET]
DBCommand 语法, 569
partial download retained 同步参数
可重新启动的下载, 150
PDB
术语定义, 785
PDF
文档, xiv
配置重定向器属性
不支持服务器组的重定向器, 258
支持服务器组的重定向器, 256
port 协议选项
MobiLink 重定向器, 253
适用于 HTTP 的 MobiLink 服务器 (mlsrv11) -x 选项, 104
适用于 HTTPS 的 MobiLink 服务器 (mlsrv11) -x 选项, 105
适用于 TCP/IP 的 MobiLink 服务器 (mlsrv11) -x 选项, 102
适用于通过 TCP/IP 的 TLS 的 MobiLink 服务器 (mlsrv11) -x 选项, 103
PowerDesigner
术语定义, 785
PowerJ
术语定义, 786
Precision 属性 [ML .NET]
DBParameter 类语法, 575
prepare_for_download
连接事件, 440
prepare_for_download 属性
MobiLink 监控器同步统计信息, 181
Prepare 方法 [ML .NET]
DBCommand 语法, 568
ProcResults
Oracle 驱动程序选项, 745
publication_nonblocking_download_ack
连接事件, 442
排序顺序
字符与 MobiLink 同步, 740
配置
Apache Web 服务器, 269
M-Business Anywhere, 271
Microsoft Web 服务器, 265
MobiLink 统一数据库, 6
MobiLink 重定向器关于, 252
Tomcat, 267
Unix 上的 NSAPI Web 服务器, 263
Windows 上的 NSAPI Web 服务器, 261
适用于 Apache Web 服务器的 Servlet 重定向器, 267
配置 NSAPI 重定向器以用于 Netscape/Sun Web 服务器
Windows, 261
配置 Servlet 重定向器
Apache Web 服务器, 267
偏移
ml_subscription 表的 progress 列, 689
瓶颈
MobiLink 性能, 163

Q

QAnywhere
MobiLink 系统表, 654
属性, 69
术语定义, 786
部署, 766
QAnywhere 代理
术语定义, 786
QAnywhere 客户端
部署, 766
QAnywhere 客户端系统表
ml_qa_clients, 671
ml_qa_delivery, 672
ml_qa_delivery_archive, 673
ml_qa_global_props, 674
ml_qa_notifications, 675
ml_qa_repository, 676

ml_qa_repository_archive, 677
ml_qa_repository_props, 678
ml_qa_repository_props_archive, 679
ml_qa_repository_staging, 680
ml_qa_status_history, 681
ml_qa_status_history_archive, 682
ml_qa_status_staging, 683
企业数据库
与 MobiLink 同步, 609
启动
MobiLink 服务器, 28
MobiLink 监控器 (mlmon), 169
通告程序, 72
启动 MobiLink 监控器
关于, 169
启动类
.NET 的 MLStartClasses 选项, 86
Java 的 DMLStartClasses 选项, 87
MobiLink .NET 同步逻辑, 557
MobiLink Java 同步逻辑, 501
启用 Microsoft 分布式事务
Oracle 驱动程序选项, 745
前缀
MobiLink 命名参数, 299
嵌入式 SQL
术语定义, 786
强制冲突
MobiLink, 145
请求
在 MobiLink 中路由, 249
驱动程序
MobiLink 支持的, 744
取消未提交的错误报告
监控器, 219
权限
MobiLink 服务器, 28
全局
MobiLink 中的脚本版本, 303
全局程序集高速缓存
在 MobiLink 中实施, 563
全局脚本版本
MobiLink, 303
全局临时表
术语定义, 786
全局自动增量
MobiLink 唯一主键, 132
MobiLink 声明, 132
为 MobiLink 设置 global_database_id, 133
算法, 133
缺省隔离级别
MobiLink, 156
缺省全局自动增量
MobiLink 声明, 132

R

r.
MobiLink 命名参数前缀, 299
RDBMS
术语定义, 778
READPAST 表提示
download_cursor 问题, 371
download_delete_cursor 问题, 373
upload_fetch 问题, 467
redirector_server_group.config
示例（适用于支持服务器组的重定向器）, 258
redirector.config
配置（适用于不支持服务器组的重定向器）, 258
示例（适用于不支持服务器组的重定向器）, 260
位置（适用于不支持服务器组的重定向器）, 258
位置（适用于支持服务器组的重定向器）, 256
配置（适用于支持服务器组的重定向器）, 256
REMOTE DBA 权限
术语定义, 798
Remove(object value) 方法 [ML .NET]
DBParameterCollection 类语法, 578
RemoveAt(int index) 方法 [ML .NET]
DBParameterCollection 类语法, 579
RemoveAt(string parameterName) 方法 [ML .NET]
DBParameterCollection 类语法, 577
removeErrorListener 方法 [ML Java]
ServerContext 语法, 536
removeInfoListener 方法 [ML Java]
ServerContext 语法, 536
removeShutdownListener 方法 [ML Java]
ServerContext 语法, 537
removeWarningListener 方法 [ML Java]
ServerContext 语法, 537
report_error
语法, 317
连接事件, 444
report_odbc_error

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

连接事件, 447
resolve_conflict
使用, 140
表事件, 450
resume partial download 同步参数
可重新启动的下载, 150
Rollback 方法 [ML .NET]
DBConnection 语法, 570
rsa 协议选项
适用于 HTTPS 的 MobiLink 服务器 (mlsrv11) -x 选项, 105
适用于 TCP/IP 的 MobiLink 服务器 (mlsrv11) -x 选项, 103
rshost (*见* 中继服务器状态管理器)
日志
(*参见* 日志文件)
日志文件
MobiLink 服务器, 31
MobiLink 服务器查看, 31
术语定义, 786
日志文件查看器
MobiLink 服务器日志, 32
如何选择缺省值
MobiLink 全局自动增量, 133
软关机
MobiLink 停止实用程序 (mlstop), 649

S

s.
MobiLink 命名参数前缀, 299
samples-dir
文档用法, xvi
Scale 属性 [ML .NET]
DBParameter 类语法, 575
ServerContext [ML Java]
语法, 530
ServerContext 接口 [ML .NET]
语法, 589
ServerException 构造函数 [ML .NET]
语法, 592
ServerException 构造函数 [ML Java]
语法, 539
ServerException 类 [ML .NET]
语法, 592
ServerException 类 [ML Java]
语法, 538
Servlet
为 Apache Web 服务器安装重定向器, 267
servlet 重定向器
Apache Tomcat, 267
Servlet 重定向器
Apache Web 服务器, 267
session_key
适用于 HTTP 的 MobiLink 服务器 (mlsrv11) -xo 选项, 109
session_key 协议选项
适用于 HTTPS 的 MobiLink 服务器 (mlsrv11) -xo 选项, 110
SetNewRowValues 方法 [ML .NET]
UpdateDataReader 接口语法, 604
setNewRowValues 方法 [ML Java]
SynchronizationException 语法, 542
SET NOCOUNT
SQL Server MobiLink 统一数据库, 19
SetOldRowValues 方法 [ML .NET]
UpdateDataReader 接口语法, 604
setOldRowValues 方法 [ML Java]
面向 Java 的 MobiLink 服务器 API
SynchronizationException 类, 543
setValue 方法 [ML Java]
InOutInteger 语法, 523
InOutString 语法, 524
设置重定向器
关于, 252
ShutdownCallback 委派 [ML .NET]
语法, 593
ShutdownListener 方法 [ML .NET]
ServerContext 接口语法, 591
ShutdownListener 接口 [ML Java]
语法, 539
shutdownPerformed 方法 [ML Java]
ShutdownListener 语法, 540
ShutDown 方法 [ML .NET]
ServerContext 接口语法, 591
shutdown 方法 [ML Java]
ServerContext 语法, 537
SID
Oracle 驱动程序选项, 745
SLEEP
重定向器属性（适用于不支持服务器组的重定向器）, 259
重定向器属性（适用于支持服务器组的重定向器）, 257
SQL

术语定义, 790
SQL_ARD_TYPE 字段 [ML .NET]
SQLType 枚举语法, 598
SQL_BIGINT 字段 [ML .NET]
SQLType 枚举, 599
SQL_BINARY 字段 [ML .NET]
SQLType 枚举语法, 599
SQL_BIT 字段 [ML .NET]
SQLType 枚举语法, 598
SQL_CHAR 字段 [ML .NET]
SQLType 枚举语法, 594
SQL_DATETIME 字段 [ML .NET]
SQLType 枚举语法, 596
SQL_DATE 字段 [ML .NET]
SQLType 枚举语法, 596
SQL_DECIMAL 字段 [ML .NET]
SQLType 枚举语法, 594
SQL_DEFAULT 字段 [ML .NET]
SQLType 枚举语法, 598
SQL_DOUBLE 字段 [ML .NET]
SQLType 枚举语法, 596
SQL_FLOAT 字段 [ML .NET]
SQLType 枚举语法, 595
SQL_GUID 字段 [ML .NET]
SQLType 枚举语法, 600
SQL_INTEGER 字段 [ML .NET]
SQLType 枚举语法, 595
SQL_INTERVAL 字段 [ML .NET]
SQLType 枚举语法, 596
SQL_LONGVARBINARY 字段 [ML .NET]
SQLType 枚举语法, 599
SQL_LONGVARCHAR 字段 [ML .NET]
SQLType 枚举语法, 600
SQL_NUMERIC 字段 [ML .NET]
SQLType 枚举语法, 594
SQL_REAL 字段 [ML .NET]
SQLType 枚举语法, 595
SQL_SMALLINT 字段 [ML .NET]
SQLType 枚举语法, 595
SQL_TIMESTAMP 字段 [ML .NET]
SQLType 枚举语法, 597
SQL_TIME 字段 [ML .NET]
SQLType 枚举语法, 596
SQL_TINYINT 字段 [ML .NET]
SQLType 枚举语法, 598
SQL_TXN_READ_COMMITTED
MobiLink 隔离级别, 156
SQL_TYPE_DATE 字段 [ML .NET]
SQLType 枚举语法, 597
SQL_TYPE_NULL 字段 [ML .NET]
SQLType 枚举语法, 594
SQL_TYPE_TIMESTAMP 字段 [ML .NET]
SQLType 枚举语法, 598
SQL_TYPE_TIME 字段 [ML .NET]
SQLType 枚举语法, 597
SQL_UNKNOWN_TYPE 字段 [ML .NET]
SQLType 枚举语法, 594
SQL_VARBINARY 字段 [ML .NET]
SQLType 枚举语法, 599
SQL_VARCHAR 字段 [ML .NET]
SQLType 枚举语法, 597
SQL_WCHAR 字段 [ML .NET]
SQLType 枚举, 600
SQL_WLONGVARCHAR 字段 [ML .NET]
SQLType 枚举语法, 600
SQL_WVARCHAR 字段 [ML .NET]
SQLType 枚举语法, 600
SQL-.NET 数据类型
MobiLink .NET 同步逻辑, 556
SQL Anywhere
MobiLink 隔离级别, 156
作为 MobiLink 统一数据库, 26
文档, xiv
术语定义, 790
SQL Anywhere 统一数据库
MobiLink, 26
SQL-java 数据类型
关于, 498
SQL Remote
术语定义, 790
SQL Server
(*参见* Microsoft SQL Server)
MobiLink 数据映射, 720
作为 MobiLink 统一数据库, 19
SQLType 枚举 [ML .NET]
ServerException 类语法, 593
SQL 同步逻辑
MobiLink, 293
MobiLink 性能, 162
SQL 语法
MobiLink 服务器 (mlsrv11), 41
SQL 语句
术语定义, 790
start_time 属性

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

MobiLink 监控器同步统计信息, 181
StaticCursorLongColBuffLen
ASE, 10
STOP SYNCHRONIZATION DELETE 语句
SQL Anywhere 客户端, 147
用法, 314
Sun Java System Web 服务器
在 Unix 上配置 NSAPI 重定向器, 263
在 Windows 上配置 NSAPI 重定向器, 261
Sun One
在 Unix 上为 NSAPI 重定向器进行配置, 263
在 Windows 上为 NSAPI 重定向器进行配置, 261
Sun Web 服务器
在 Unix 上配置 NSAPI 重定向器, 263
在 Windows 上配置 NSAPI 重定向器, 261
Sybase Adaptive Server Enterprise (*见* Adaptive Server Enterprise)
Sybase Central
在 32 位 Unix 上部署 MobiLink 服务器, 757
在 64 位 Unix 上部署 MobiLink 服务器, 760
术语定义, 791
sync_deadlocks 属性
MobiLink 监控器同步统计信息, 181
sync_errors 属性
MobiLink 监控器同步统计信息, 181
sync_request 属性
MobiLink 监控器同步统计信息, 181
sync_tables 属性
MobiLink 监控器同步统计信息, 181
sync_warnings 属性
MobiLink 监控器同步统计信息, 181
sync.conf
M-Business Anywhere 重定向器, 271
syncase.sql
关于, 10
syncdb2long.sql
关于, 12
synchronization_statistics
表事件, 456
连接事件, 453
SynchronizationException 构造函数 [ML .NET]
SynchronizationException 类语法, 601
SynchronizationException 构造函数 [ML Java]
SynchronizationException 语法, 541
SynchronizationException 类 [ML .NET]
语法, 601
SynchronizationException 类 [ML Java]
语法, 540
syncmss.sql
关于, 19
syncora.sql
关于, 23
SyncRoot 属性 [ML .NET]
DBParameterCollection 类语法, 580
syncsa.sql
关于, 26
sync 属性
MobiLink 监控器同步统计信息, 181
SYS
术语定义, 791
散列
术语定义, 786
删除
MobiLink .NET 表脚本, 629
MobiLink .NET 连接脚本, 628
MobiLink Java 表脚本, 631
MobiLink Java 连接脚本, 630
MobiLink SQL 表脚本, 639
MobiLink SQL 连接脚本, 627
MobiLink 下载, 314
MobiLink 属性, 637
停止 SQL Anywhere 客户端的上载, 147
远程 MobiLink 数据库中的行, 314
删除表中的所有行
MobiLink, 314
删除脚本
MobiLink 关于, 306
删除行
MobiLink 技术, 147
MobiLink 远程数据库, 314
上次下载的时间戳
modify_last_download_timestamp 连接事件, 430
modify_next_last_download_timestamp 连接事件, 433
上次下载时间
关于 MobiLink, 122
上次下载时间戳
MobiLink 生成, 122
关于 MobiLink, 122
上载
MobiLink 事务, 322
MobiLink 暂停, 147
术语定义, 787

用于上载行的 MobiLink 脚本, 309
上载过程中的事件
关于, 326
编写用于上载行的脚本, 309
上载事件
MobiLink 同步, 326
关于, 309
上载事务
MobiLink, 322
上载行
.NET 同步技术, 562
MobiLink 性能, 162
编写脚本, 309
设备跟踪
术语定义, 787
设置
MobiLink .NET 同步逻辑, 553
MobiLink Java 同步逻辑, 495
MobiLink 统一数据库, 3
MobiLink 重定向器关于, 252
设置 .NET 同步逻辑
关于, 553
设置 IBM DB2 LUW 统一数据库
MobiLink, 12
设置 IBM DB2 主机统一数据库
MobiLink, 15
设置 MySQL 统一数据库
MobiLink, 21
设置直接行处理
关于, 611
生成的连接条件
术语定义, 788
升级应用程序
使用多个 MobiLink 脚本版本, 303
失败的下载
MobiLink, 149
同步技术, 149
时间戳
MobiLink 下载, 122
时间更改
MobiLink, 123
实例化视图
术语定义, 787
实用程序
MobiLink, 647
MobiLink 停止实用程序 (mlstop), 649
MobiLink 服务器 (mlsrv11), 41
MobiLink 用户验证 (mluser), 650
MobiLink 重定向器, 249
使用 .NET 编写同步脚本
关于, 551
使用 .NET 处理 MobiLink 服务器错误
MobiLink .NET 同步逻辑, 559
使用 download_delete_cursor 脚本删除行
MobiLink, 314
使用 Java 编写同步脚本
关于, 493
使用 Java 处理 MobiLink 服务器错误
MobiLink Java 同步逻辑, 500
示例
.NET 同步逻辑, 565
Java 同步逻辑, 504
MobiLink 基于文件的下载, 283
示例属性
MobiLink 监控器, 177
示例域配置文件
MobiLink, 564
世代号
MobiLink 基于文件的下载, 281
术语定义, 787
事件
MobiLink, 320
MobiLink 直接行处理, 610
关于 MobiLink, 293
关于 MobiLink 事件, 295
关于 MobiLink 同步, 321
事件模型
MobiLink 伪代码, 324
术语定义, 787
事务
MobiLink, 321
MobiLink .NET 同步逻辑, 555
MobiLink Java 同步逻辑, 497
术语定义, 787
事务日志
术语定义, 787
事务日志镜像
术语定义, 788
事务完整性
术语定义, 788
视图
术语定义, 787
守护程序
运行 MobiLink 作为, 33

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

授权选项
术语定义, 788
受保护的功能
术语定义, 788
属性
DBParameter 类语法, 575
QAnywhere 服务器, 69
术语表
SQL Anywhere 术语列表, 773
数据不一致
MobiLink 冲突处理, 137
数据操作语言
术语定义, 788
数据交换 (*见* 同步)
数据库
MobiLink 统一数据库, 3
术语定义, 789
数据库对象
术语定义, 789
数据库服务器
术语定义, 789
数据库工作线程
MobiLink, 163
MobiLink 性能, 160
数据库管理员
术语定义, 789
数据库连接
MobiLink 性能, 165
MobiLink 性能设置最大, 161
术语定义, 789
数据库名称
术语定义, 789
数据库模式
将统一表与 MobiLink 远程表相关联, 5
数据库所有者
术语定义, 790
数据库文件
术语定义, 790
数据类型
MobiLink .NET 和 SQL, 556
MobiLink Java 和 SQL, 498
MobiLink 中的 ASE, 696
MobiLink 中的 IBM DB2 LUW, 704
MobiLink 中的 IBM DB2 主机, 711
MobiLink 中的 Microsoft SQL Server, 720
MobiLink 中的 MySQL, 726
MobiLink 中的 Oracle, 731
MobiLink 统一数据库映射, 695
术语定义, 788
数据类型映射
MobiLink 统一数据库, 695
数据立方体
术语定义, 789
数据流 (MobiLink) (*见* 直接行处理)
数据输入
MobiLink, 146
数据映射
关于, 695
数据源名称
Oracle 驱动程序选项, 745
数组大小
Oracle 驱动程序选项, 745
双向同步
关于, 130
必需的脚本, 305
死锁
术语定义, 790
碎片
(*参见* 分区)
索引
MobiLink 性能, 162
术语定义, 791
锁定
术语定义, 790

T

TCP/IP
MobiLink 服务器 -x 选项, 102
Text 属性 [ML .NET]
DBRowReader 接口语法, 589
this[int index] 属性 [ML .NET]
DBParameterCollection 类语法, 581
this[string parameterName] 属性 [ML .NET]
DBParameterCollection 类语法, 580
调试
.NET 同步逻辑, 560
MobiLink 服务器日志, 31
MobiLink 连接, 38
使用 Java 类的 MobiLink 同步, 499
调试 .NET 同步逻辑
关于, 560
调试 Java 类
MobiLink Java 同步逻辑, 499
调优性能

MobiLink, 163
time_statistics
表事件, 462
连接事件, 459
TLS
(*参见* 传送层安全)
MobiLink 服务器 -x 选项, 102
在 32 位 Unix 上部署 MobiLink 服务器, 757
在 32 位 Windows 上部署 MobiLink 服务器, 751
在 64 位 Unix 上部署 MobiLink 服务器, 760
在 64 位 Windows 上部署 MobiLink 服务器, 754
在 Unix 上部署 MobiLink 客户端, 764
在 Windows 上部署 MobiLink 客户端, 763
tls_type 协议选项
适用于 HTTPS 的 MobiLink 服务器 (mlsrv11) -x 选项, 105
适用于 TCP/IP 的 MobiLink 服务器 (mlsrv11) -x 选项, 103
Tomcat
重定向器支持的版本, 267
配置 servlet 重定向器, 267
Type 属性 [ML .NET]
DBRowReader 接口语法, 589
提示
MobiLink 性能, 159
同步技术, 120
添加
MobiLink .NET 表脚本, 629
MobiLink .NET 连接脚本, 628
MobiLink Java 表脚本, 631
MobiLink Java 连接脚本, 630
MobiLink SQL 表脚本, 639
MobiLink SQL 连接脚本, 627
MobiLink 属性, 637
在 MobiLink 中添加用户名, 650
用 Sybase Central 同步脚本, 306
监控器用户, 212
添加表脚本向导
使用, 306
添加或删除脚本
MobiLink, 306
添加脚本
MobiLink 关于, 306
添加脚本版本
MobiLink, 304
添加同步脚本
使用存储过程, 307
停止
MobiLink 停止实用程序 (mlstop), 649
MobiLink 服务器, 30
SQL Anywhere 客户端的删除上载, 147
停止实用程序 (mlstop)
语法, 649
通告程序
在 32 位 Unix 上部署, 757
在 32 位 Windows 上部署, 751
在 64 位 Unix 上部署, 760
在 64 位 Windows 上部署, 754
术语定义, 791
通过 Web 服务器同步
重定向器（不建议使用）, 249
中继服务器, 223
通信
MobiLink mlsrv11 -c 选项, 50
MobiLink 服务器 -x 选项, 102
通信流
术语定义, 791
同步
可重新启动的下载, 149
mlsrv11 的协议选项, 102
MobiLink 中的 ASE 数据类型, 696
MobiLink 中的 IBM DB2 LUW 数据类型, 704
MobiLink 中的 IBM DB2 主机数据类型, 711
MobiLink 中的 Microsoft SQL Server 数据类型, 720
MobiLink 中的 MySQL 数据类型, 726
MobiLink 中的 Oracle 数据类型, 731
MobiLink 字符集, 740
MobiLink 字符集转换, 740
MobiLink 实用程序, 647
MobiLink 系统表, 654
MobiLink 系统过程, 623
Web 服务器配置, 249
下载行, 312
事件概述, 321
使用 .NET 编写 MobiLink 脚本, 551
使用 Java 编写 MobiLink 脚本, 493
冲突解决, 137
删除行, 314
在 MobiLink 中的数据类型映射, 695
多对多关系, 128
快照, 125
性能提示, 159
技术, 119

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

按字母顺序排列的脚本列表, 320
术语定义, 791
监控器连接参数, 169
统一数据库, 3
编写脚本, 293
过程, 297
运行 MobiLink 服务器, 27
同步表
MobiLink ml_table 系统表, 691
同步参数
HTTP 同步, 102
HTTPS 同步, 102
TCP/IP 同步, 102
同步除统一数据库之外的数据源
关于, 609
同步错误
处理 MobiLink, 317
故障排除, 61
同步服务器 (*见* MobiLink 服务器)
同步技术
上载行, 309
主键池, 134
关于, 119
分区, 127
删除行, 147
基于快照的同步, 125
基于时间戳的同步, 121
失败的下载, 149
存储过程用于下载, 153
数据输入, 146
同步脚本
.NET, 551
.NET 方法, 557
DBMS 依赖性, 8
download_cursor, 313
handle_error 事件, 317
Java, 493
Java 实现, 495
Java 方法, 499
MobiLink 事件, 320
report_error, 317
使用 Sybase Central 添加, 306
使用存储过程添加或删除, 307
关于, 293
参数, 299
实施 .NET, 553
执行时间, 297
支持的 DBMS 脚本策略, 8
添加和删除, 306
示例, 295
类型, 298
编写用于上载行的脚本, 309
编写用于下载行的脚本, 312
脚本版本, 303
表脚本, 298
连接脚本, 298
同步流库
在 32 位 Unix 上部署 MobiLink 服务器, 757
在 32 位 Windows 上部署 MobiLink 服务器, 751
在 64 位 Unix 上部署 MobiLink 服务器, 760
在 64 位 Windows 上部署 MobiLink 服务器, 754
同步逻辑
MobiLink, 293
同步事件
ASE begin_connection_autocommit 连接事件, 344
MobiLink 上载, 326
MobiLink 下载, 329
关于, 320
关于 MobiLink 同步, 321
按字母顺序排列的事件脚本列表, 320
同步属性
MobiLink 监控器, 177
同步新的远程数据库
MobiLink 基于文件的下载, 278
同步序列号
ml_subscription 表的 progress 列, 689
同步用户
MobiLink 用户验证 (mluser), 650
同步预订
(*参见* 预订)
同步自引用表
MobiLink, 155
统计信息
MobiLink, 181
统计信息属性
MobiLink, 181
统一数据库
ASE 作为 MobiLink, 10
DBMS 依赖性, 8
IBM DB2 LUW 作为 MobiLink, 12
IBM DB2 主机作为 MobiLink, 15
MobiLink 系统表, 654
MobiLink 隔离级别, 156

MySQL 作为 MobiLink, 21
Oracle 作为 MobiLink, 23
SQL Anywhere 作为 MobiLink, 26
SQL Server 作为 MobiLink, 19
关于, 3
创建 MobiLink, 6
在 MobiLink 中的数据类型映射, 695
将同步脚本添加到, 306
将表与 MobiLink 远程表相关联, 5
术语定义, 791
除 SQL Anywhere 之外的数据库, 8
图标
此帮助文档中使用的, xvii
图形窗格
MobiLink 监控器, 172
推式请求
术语定义, 792
推式通知
术语定义, 792
退出
MobiLink 服务器, 30
托管中继服务器
关于, 243
添加服务器群, 243
登录, 243
预订, 243

U

u.
MobiLink 用户定义的参数前缀, 300
ULRollbackPartialDownload 函数
可重新启动的下载, 150
UltraLite
术语定义, 792
部署, 765
UltraLite 运行时
术语定义, 792
Unix
MobiLink 服务器作为守护程序, 33
unknown_timeout 协议选项
跨防火墙同步, 253
UpdateDataReader 接口 [ML .NET]
语法, 603
UpdateData 接口 [ML .NET]
语法, 602
UpdateResultSet [ML Java]
SynchronizationException 语法, 542
UPDATE 冲突
MobiLink, 137
upload_bytes 属性
MobiLink 监控器同步统计信息, 181
upload_deadlocks 属性
MobiLink 监控器同步统计信息, 181
upload_delete
表事件, 465
upload_deleted_rows 属性
MobiLink 监控器同步统计信息, 181
upload_errors 属性
MobiLink 监控器同步统计信息, 181
upload_fetch
冲突检测概述, 138
检测冲突, 138
表事件, 467
upload_fetch_column_conflict
冲突检测概述, 138
检测冲突, 138
表事件, 469
upload_insert
表事件, 471
upload_inserted_rows 属性
MobiLink 监控器同步统计信息, 181
upload_new_row_insert
表事件, 473
upload_old_row_insert
表事件, 476
upload_statistics
表事件, 483
连接事件, 479
upload_update
使用, 142
冲突检测概述, 138
检测冲突, 139
表事件, 488
upload_updated_rows 属性
MobiLink 监控器同步统计信息, 181
upload_warnings 属性
MobiLink 监控器同步统计信息, 181
UploadData 接口 [ML Java]
语法, 543
UploadedTableData 接口 [ML .NET]
语法, 604
UploadedTableData 接口 [ML Java]
语法, 545
upload 属性

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

MobiLink 监控器同步统计信息, 181
url_suffix 协议选项
MobiLink 重定向器, 253
user 属性
MobiLink 监控器同步统计信息, 181
User 属性 [ML .NET]
DBRowReader 接口, 589
UUID
MobiLink 同步应用程序, 131

V

Value 属性 [ML .NET]
DBParameter 类语法, 575
VARBIT 数据类型
ASE MobiLink 统一数据库的限制, 10
VARCHAR 数据类型
MobiLink 和其它 DBMS, 8
varray (Oracle)
在存储过程中使用, 24
示例, 24
限制, 25
version 属性
MobiLink 监控器同步统计信息, 181
version 协议选项
适用于 HTTP 的 MobiLink 服务器 (mlsrv11) -x 选项, 104
适用于 HTTPS 的 MobiLink 服务器 (mlsrv11) -x 选项, 105
Visual Basic
MobiLink .NET 中的支持, 552
MobiLink 同步脚本, 551
Visual Studio
MobiLink 同步脚本, 551

W

WARNING [ML Java]
Java LogMessage 接口, 528
WARNING 字段 [ML .NET]
MessageType 枚举语法, 588
WebLogic
MobiLink 和, 609
Web 服务
与 MobiLink 同步, 609
Web 服务器
MobiLink 客户端, 253
MobiLink 的配置选项, 251
MobiLink 重定向器, 249
为 MobiLink 进行配置（适用于不支持服务器组的重定向器）, 258
与 MobiLink 同步, 609
为 MobiLink 进行配置（适用于支持服务器组的重定向器）, 256
为同步配置 Apache, 269
为同步配置 ISAPI Microsoft, 265
为同步配置 M-Business Anywhere, 271
在 Unix 上配置 NSAPI 以用于同步, 263
在 Windows 上配置 NSAPI 以用于同步, 261
Web 扩展
中继服务器, 223
Windows
术语定义, 794
Windows Mobile
术语定义, 794
Windows 上的 IIS
中继服务器，性能提示, 238
部署中继服务器, 236
外表
术语定义, 792
外部登录
术语定义, 792
外键
术语定义, 792
外键约束
术语定义, 793
外连接
术语定义, 793
完全备份
术语定义, 793
完整的事件模型
MobiLink, 321
MobiLink 伪代码, 324
完整性
术语定义, 793
网关
术语定义, 793
网络参数
MobiLink 服务器 -x 选项, 102
网络服务器
术语定义, 793
网络协议
MobiLink 服务器, 102
使用 HTTP 的 mlsrv11, 104
使用 HTTPS 的 mlsrv11, 105
使用 TCP/IP 的 mlsrv11 -x 选项, 102

使用基于 TCP/IP 的 TLS 的 mlsrv11 -x 选项, 103
术语定义, 794
唯一
MobiLink 中的主键, 131
唯一键
MobiLink, 131
唯一约束
术语定义, 794
唯一主键
MobiLink, 131
使用 UUID 为 MobiLink 生成, 131
使用组合键为 MobiLink 生成, 131
使用键池为 MobiLink 生成, 134
用于 MobiLink 的全局自动增量, 132
为 Apache Web 服务器配置 Apache 重定向器
关于, 269
为 Microsoft Web 服务器配置 ISAPI 重定向器
关于, 265
为 Netscape/Sun Web 服务器配置 NSAPI 重定向器
Unix, 263
维护版本
术语定义, 794
维护唯一主键
UUID, 131
主键池, 134
全局自动增量, 132
关于, 131
组合键, 131
伪代码
MobiLink 事件, 321
未提交的错误报告
监控器, 202
监控器警告, 219
位数组
术语定义, 794
谓语
术语定义, 794
文本文件
与 MobiLink 同步, 609
文档
SQL Anywhere, xiv
约定, xv
文件
MobiLink 基于文件的下载, 275
文件定义数据库
关于, 277
创建, 277
术语定义, 794
问号
MobiLink 脚本参数, 299
无交集分区
MobiLink, 127
定义, 127
物理索引
术语定义, 795

X

Xusage.txt
位置, 87
系统表
MobiLink 同步, 654
在 MobiLink 统一数据库中创建, 6
术语定义, 795
系统参数
MobiLink 脚本, 299
系统对象
术语定义, 795
系统过程
ml_add_cs, 627
ml_add_dcs, 628
ml_add_dts, 629
ml_add_jcs, 630
ml_add_lcs, 633
ml_add_lcs_chk, 633
ml_add_lts, 633
MobiLink, 623
MobiLink IBM DB2 主机系统过程名称转换, 624
按字母顺序排序的 MobiLink 系统过程列表, 624
系统视图
术语定义, 795
系统数据库
MobiLink, 7
下载
MobiLink 事务, 323
MobiLink 失败的下载, 149
MobiLink 性能, 162
基于文件的 MobiLink, 275
基于时间戳, 121
术语定义, 795
用于下载行的 MobiLink 脚本, 312
下载故障
MobiLink 可重新启动的下载, 149
下载过程中的事件
关于, 329

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

编写用于下载行的脚本, 312
下载缓冲区
MobiLink 性能, 160
下载确认
MobiLink 性能, 161
关于, 152
下载删除
MobiLink download_delete_cursor 脚本, 314
下载时间戳
MobiLink 生成, 122
关于 MobiLink, 122
下载事件
MobiLink 同步, 329
下载事务
MobiLink, 323
下载数据
MobiLink 中基于文件的下载, 275
下载文件
为 MobiLink 基于文件的下载创建, 278
下载行
同步脚本, 312
限制
监控器, 186
线程
(*参见* 线程)
MobiLink 工作线程和性能, 160
相关名
术语定义, 795
详细程度
MobiLink 性能, 161
MobiLink 服务器 (mlsrv11) -v 选项, 97
项目
术语定义, 795
消息传递
MobiLink QAnywhere 系统表, 654
消息存储库
术语定义, 795
消息类型
术语定义, 795
消息日志
术语定义, 796
消息属性文件
10.0.0 版中不建议使用, 69
消息系统
术语定义, 796
协议
MobiLink 服务器, 102
使用 HTTP 的 mlsrv11, 104
使用 HTTPS 的 mlsrv11, 105
使用 TCP/IP 的 mlsrv11 -x 选项, 102
使用基于 TCP/IP 的 TLS 的 mlsrv11 -x 选项, 103
卸载
术语定义, 796
新闻组
技术支持, xix
行
分区, 127
在远程 MobiLink 数据库上删除, 314
行参数
MobiLink 脚本, 299
行级触发器
术语定义, 779
性能
MobiLink, 159
MobiLink -sm 选项, 89
MobiLink 强制冲突, 145
下载, 162
性能统计
术语定义, 796
序列
MobiLink 同步中的主键唯一性, 23
序列号
ml_subscription 表的 progress 列, 689
选项
mlsrv11, 41
MobiLink 停止实用程序 (mlstop), 649
MobiLink 服务器 (mlsrv11), 41
MobiLink 用户验证 (mluser), 650
选项部分
中继服务器配置文件, 229
选择性的共享 (*见* 分区)

Y

验证
MobiLink mluser 实用程序, 650
验证参数
MobiLink, 301
MobiLink 脚本, 299
业务规则
术语定义, 796
一览表窗格
MobiLink 监控器, 175
移动设备
连接到中继服务器群, 245

疑难解答
MobiLink 远程数据丢失, 122
新闻组, xix
监控器, 221
引号
DB2 MobiLink 统一数据库, 13
引用对象
术语定义, 796
应用程序
部署 MobiLink, 749
应用程序池
创建, 237
应用程序服务器
与 MobiLink 同步, 609
影子表
编写 download_delete_cursor 脚本, 314
硬关机
MobiLink 停止实用程序 (mlstop), 649
映射
MobiLink 统一数据库数据类型, 695
用户
监控器创建, 212
监控器删除, 214
监控器发送电子邮件, 213
监控器只读用户, 212
监控器安全, 214
监控器操作员, 212
监控器管理员, 212
监控器管理员用户, 212
监控器类型, 212
监控器缺省用户, 212
用户参数
MobiLink, 300
用户定义的参数
MobiLink, 300
用户定义的过程
DB2 MobiLink 统一数据库, 13
用户定义的启动类
MobiLink .NET 同步逻辑, 557
MobiLink Java 同步逻辑, 501
用户定义数据类型
术语定义, 797
用户名
MobiLink 用户验证实用程序 (mluser), 650
用户验证实用程序 (mluser)
语法, 650
用于 .NET 的 MobiLink 服务器 API
API 参考, 567
用于 Java 和 .NET 的 MobiLink 基于对象的数据流
关于, 609
用于 MobiLink 的中继服务器
方案示例, 246
用于添加或删除脚本的系统过程
MobiLink 服务器, 624
用于添加或删除属性的系统过程
MobiLink 服务器, 624
优先级同步
MobiLink 性能, 162
游标
术语定义, 797
游标脚本
定义, 298
游标结果集
术语定义, 797
游标位置
术语定义, 797
有交集
分区, 127
有交集分区
MobiLink, 128
语法
MobiLink 停止实用程序 (mlstop), 649
MobiLink 同步实用程序, 647
MobiLink 服务器 (mlsrv11), 41
MobiLink 用户验证 (mluser), 650
MobiLink 系统过程, 623
MobiLink 脚本, 320
语句级触发器
术语定义, 797
语言库
在 32 位 Unix 上部署 MobiLink 服务器, 757
在 32 位 Windows 上部署 MobiLink 服务器, 751
在 64 位 Unix 上部署 MobiLink 服务器, 760
在 64 位 Windows 上部署 MobiLink 服务器, 754
域
术语定义, 797
域配置文件
MobiLink, 564
预订
ml_subscription 系统表, 689
术语定义, 798
元数据
术语定义, 798
原子事务

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

术语定义, 798
远程 ID
ml_database 系统表, 660
MobiLink Java API 中的 getRemoteID 方法, 511
术语定义, 798
远程表
删除 MobiLink 中的行, 314
远程数据库
在 MobiLink 中的数据类型映射, 695
将统一表与 MobiLink 远程表相关联, 5
术语定义, 798
远程数据库和统一数据库之间的 MobiLink 数据映射
关于, 695
约定
命令 shell, xvii
命令提示符, xvii
文档, xv
文档中的文件名, xvi
约束
术语定义, 799
约束错误 (*见* 冲突)
运行
MobiLink 服务器, 27
运行 .NET 同步逻辑
关于, 553
运行 Java 同步逻辑
关于, 495
运行 MobiLink
作为守护程序, 33
关于, 27
在当前会话外, 33
运行 MobiLink 服务器
作为服务, 33
关于, 28
在服务器群中, 37
运营公司
术语定义, 799

Z

自定义校验
MobiLink 基于文件的下载, 282
自动校验
MobiLink 基于文件的下载, 280
增量备份
术语定义, 799
争用
MobiLink 性能, 160
MobiLink 性能解释, 163
术语定义, 799
正则表达式
术语定义, 799
支持
新闻组, xix
支持新旧客户端
MobiLink, 255
直方图
术语定义, 799
直接上载的冲突处理
MobiLink 直接行处理, 615
直接同步事件
关于, 610
直接行处理
DownloadData 接口 [ML Java], 513
DownloadTableData 接口 [ML Java], 515
handle_DownloadData 连接事件, 410
handle_UploadData 连接事件, 422
SendColumnNames, 612
UpdateResultSet 接口, 542
UploadData 接口 [ML Java], 543
UploadedTableData 接口 [ML Java], 545
上载, 614
下载, 620
关于, 609
开发提示, 612
快速入门, 611
术语定义, 799
只读用户
监控器, 212
监控器要求登录, 214
中继服务器
HTTP 负载平衡器, 226
rshost 命令行语法, 235
rsoe 语法, 231
中继服务器群, 225
体系结构, 224
使用 MobiLink, 245
关于, 224
出站启动器, 226, 231
出站启动器语法, 231
出站启动器部署, 231
后端服务器群, 226
托管服务, 243
更新配置, 241

状态管理器, 234
通过 Web 服务器同步, 223
部署, 236
配置文件, 227
配置文件格式, 229
中继服务器 web 扩展
启用, 237
中继服务器 Web 扩展
关于, 223
中继服务器部分
中继服务器配置文件, 227
中继服务器部署
Linux 上的 Apache, 239
Linux 上的文件, 239
web 服务器扩展, 237, 239
Windows 上的 IIS, 236
Windows 上的文件, 236
应用程序池，创建, 237
状态管理器, 237, 239
中继服务器配置文件
中继服务器部分, 227
关于, 227
后端服务器部分, 228
后端群部分, 228
格式, 229
选项部分, 229
中继服务器群
连接客户端, 245
连接移动设备, 245
中继服务器群配置
更新, 241
中继服务器托管服务
预订, 243
中继服务器状态管理器
作为 Windows 服务启动, 234
关于, 234
命令行语法, 235
自动启动, 235
通过自定义选项自动启动, 235
中央数据库
MobiLink 统一数据库, 3
重定向器
指定位置（适用于支持服务器组的重定向器）, 256
主表
术语定义, 800
主机
DB2 作为 MobiLink 统一数据库, 15
主键
MobiLink 关于, 120
MobiLink 唯一性技术, 131
Oracle 序列, 23
术语定义, 800
主键池
MobiLink 唯一主键, 134
MobiLink 示例, 135
使用缺省全局自动增量为 MobiLink 生成唯一值, 132
主键约束
术语定义, 800
主题
图标, xvii
注册
方法作为 MobiLink 脚本, 624
注册 MobiLink 用户
mluser 实用程序, 650
注册方法
用于 .NET 的 MobiLink 服务器 API, 557
用于 Java 的 MobiLink 服务器 API, 499
专用程序集
在 MobiLink 中实施, 563
转换
字符集通过 ODBC 驱动程序, 741
装载程序集
MobiLink .NET 同步逻辑, 563
状态
ml_subscription 表的 progress 列, 689
监控器, 200
状态管理
中继服务器, 234
状态管理器
命令行语法, 235
资源
监控器, 199
子查询
术语定义, 800
子集
将数据子集下载到远程, 127
自定义您的统计信息
MobiLink 监控器, 180
自动增量方法
Oracle MobiLink 统一数据库, 23
自然连接
术语定义, 788

 Copyright © 2009, iAnywhere Solutions, Inc. - SQL Anywhere 11.0.1

自引用表
MobiLink, 155
字符串
术语定义, 800
字符集
MobiLink 同步, 740
术语定义, 800
字符集之间的转换
MobiLink 同步, 740
字符集注意事项
MobiLink, 740
字符集转换
MobiLink 同步期间, 740
通过 ODBC 驱动程序, 741
阻塞下载确认
关于, 152
组合键
MobiLink 唯一主键, 131
最近修改的列
MobiLink, 121