SYBASE®
An SAP® Company

参考手册

Replication Server® 15.7.1

文档 ID： DC37893-01-1571-01
最后修订日期： 2012 年 4 月

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

目录

约定

Sybase® 文档中使用以下样式和语法约定。

样式约定

关键字	定义
等宽字体（固定宽度）	• SQL 和程序代码 • 完全按照所示输入的命令 • 文件名 • 目录名
等宽斜体	在 SQL 或程序代码段中，用户指定的值的占位符（请参见下面的示例）。
斜体	• 文件名和变量名 • 对其它主题或文档的交叉引用 • 在文本中，用户指定的值的占位符（请参见下面的示例） • 文本中的词汇表术语
粗体 san serif	• 命令、函数、存储过程、实用程序、类和方法的名称 • 词汇表条目（在词汇表中） • 菜单选项路径 • 在编号任务或过程步骤中，您单击的用户界面 (UI) 元素，如按钮、复选框、图标等

如有必要，接下来会在文本中对占位符（特定于系统或设置的值）进行说明。例如：运行：

```
installation directory\start.bat
```

其中 *installation directory* 是应用程序的安装位置。

语法约定

关键字	定义
{ }	大括号表示必须至少选择括号中的一个选项。不要在输入命令时键入大括号。
[ ]	中括号表示可以选择括号中的一个或多个选项，也可不选。不要在输入命令时键入中括号。

关键字	定义
( )	小括号应作为命令的一部分输入。
|	竖线表示只能选择一个显示的选项。
,	逗号表示可以选择任意多个显示的选项，逗号作为命令的一部分输入以分隔选项。
...	省略号（三点）表示可以将最后一个单元重复任意多次。不要在命令中包括省略号。

区分大小写

- 所有命令语法和命令示例都以小写形式显示。但是，复制命令名称不区分大小写。例如，**RA_CONFIG**、**Ra_Config** 和 **ra_config** 是等效的。
- 配置参数的名称区分大小写。例如，**Scan_Sleep_Max** 与 **scan_sleep_max** 不同，前者将被解释为无效参数名称。
- 复制命令中的数据库对象名称不区分大小写。但是，若要在复制命令中使用混合大小写的对象名（以与主数据库中混合大小写的对象名相匹配），请用引号字符分隔该对象名。例如： **pdb_get_tables** "*TableName*"
- 根据有效的排序顺序，标识符和字符数据可能要区分大小写。
  - 如果使用区分大小写的排序顺序（如“binary”），则必须用正确的大写和小写字母组合形式输入标识符和字符数据。
  - 如果使用不区分大小写的排序顺序（如“nocase”），则可以用任意大写或小写字母组合形式输入标识符或字符数据。

术语

Replication Agent™ 是用于描述 Replication Agent for Adaptive Server® Enterprise、Replication Agent for Oracle、Replication Agent for IBM DB2 UDB 和 Replication Agent for Microsoft SQL Server 的通用术语。特定名称包括:

- RepAgent – 用于 Adaptive Server Enterprise 的 Replication Agent 线程
- Replication Agent for Oracle
- Replication Agent for Microsoft SQL Server
- Replication Agent for UDB – 用于 Linux、Unix 和 Windows 上的 IBM DB2

复制命令语言简介

了解每个类别的命令。某些命令同时属于多个类别。

有关完整的命令语法和用法信息，请参见 Replication Server® 命令。

在使用复制命令语言 (RCL) 时，请遵循下面的格式设置规则：

- 除了在关键字或标识符中间，您可以在任何位置断行。
- 可以通过在行尾输入反斜杠 (\)，使字符串在下一行继续。除反斜杠之后的空格外，行上多余的空格字符将被忽略。
- 请不要在反斜杠之后输入任何空格。除非另外说明，否则可以在一个批处理中输入多个命令。
- RCL 命令是非事务型命令。Replication Server 执行批处理中的每个命令，而不考虑批处理中其它命令的完成状态。一个命令中的语法错误会导致 Replication Server 无法分析批处理中后续命令。

有关数据类型、标识符、保留字以及对 Adaptive Server 的支持的详细信息，请参见“主题”。

有关 Replication Server 体系结构的介绍，请参见《Replication Server 管理指南第一卷》中的“Replication Server 简介”和《Replication Server 管理指南第一卷》>“Replication Server 技术概述”。

有些 Replication Server 过程可能需要您执行 Adaptive Server 系统过程，例如 **sp_setreptable** 或 **sp_setrepproc**。有关完整的语法和用法信息，请参见“Adaptive Server 命令和系统过程”。

Replication Manager (RM) 提供了另一种方法来执行许多由 RCL 命令执行的任务。有关详细信息，请参见《Replication Server 管理指南第一卷》。

数据复制命令

数据复制命令创建并管理复制定义、发布和预订，这些使得复制表或存储过程成为可能。

表复制定义命令

表复制定义说明要复制的表和列。主表是复制的源；复制表是复制的目标。可以为每个主表创建一个或多个复制定义。

在管理存储主表的数据库的 Replication Server 中创建复制定义。

复制定义包括：

- 复制定义的名称
- 主表和复制表的名称（如果它们彼此不同，而且与复制定义名称也不相同）
- 主表的位置
- 要复制的主列的名称和数据类型以及相应复制列的名称
- 组成该表主键的列的名称

复制定义还可包括以下可选内容：

- 可在预订的 **where** 子句中引用的列的名称
- 是否将复制定义和它的列用于复制到备用数据库
- 复制所有列还是复制执行 **update** 和 **delete** 操作所需的最小列数
- *text*、*unitext*、*image* 和 *rawobject* 列的复制状态
- 是否将复制值的数据类型从主数据库的数据类型更改为复制数据库的数据类型。

创建复制定义时，不会分配任何数据。必须在每个复制数据库中创建表的副本，然后创建预订才能开始复制数据。

可以使用以下命令对表复制定义进行操作：

- **create replication definition** – 创建表的复制定义。
- **alter replication definition** – 更改复制定义。
- **drop replication definition** – 删除复制定义。

有关用于预订复制定义的命令，请参见“预订命令”。

另请参见

- 预订命令 （第 6 页）

函数复制定义命令

函数复制定义指定有关要复制的存储过程的信息。

在管理主数据库的 Replication Server 中创建函数复制定义。

函数复制定义包括：

- 函数复制定义的名称。
- 主数据的位置。
- 要复制的存储过程参数的名称和数据类型。

函数复制定义还可包括以下可选内容：

- 在源数据库中执行的存储过程的名称；如果该存储过程的名称不同于函数复制定义的名称，还包括要在目标数据库中执行的存储过程的名称。
- 可在预订的 **where** 子句中引用的参数的名称。
- 是否在复制到备份数据库时使用函数复制定义及其参数。

可以使用以下命令对函数复制定义进行操作：

- **create applied function replication definition** – 创建存储过程的应用函数复制定义。
- **alter applied function replication definition** – 更改应用函数复制定义。
- **create request function replication definition** – 创建存储过程的请求函数复制定义。
- **alter request function replication definition** – 更改请求函数复制定义。
- **drop function replication definition** – 删除函数复制定义。

创建函数复制定义时，不会分配任何数据。必须在主数据库和复制数据库中创建存储过程，并且必须在复制 Replication Server 上创建预订。

有关用于预订复制定义的命令，请参见“预订命令”。

数据库复制定义命令

数据库复制定义描述要复制的数据库或数据库对象。您可以选择复制整个数据库，也可以选择复制（或不复制）该数据库中的特定表、函数、事务、DDL 和系统存储过程。

数据库复制定义包括：

- 数据库复制定义的名称
- 要复制的数据库所在的主服务器的名称
- 要复制的数据库的名称

数据库复制定义还可包括以下可选内容：

- 一个指示是否将 DDL 复制到预订数据库的指示符
- 一个指示是否将表、存储过程、用户定义的函数、事务或系统过程复制到预订数据库的指示符

使用以下命令处理数据库复制定义：

- **create database replication definition** – 创建用于复制数据库或数据库对象的复制定义。
- **alter database replication definition** – 更改现有数据库复制定义。
- **drop database replication definition** – 删除现有数据库复制定义。

发布命令

使用 Replication Server 的发布功能可以将要预订的表和过程及它们的复制定义归为一组，并为该组创建一个预订。

*发布*是来自同一个主数据库的项目集。每个*项目*都由一个表或存储过程的复制定义，以及一组指定相关行的 **where** 子句组成。一个项目可以包含零个、一个或多个 **where** 子句。多个子句之间用 **or** 关键字分隔。

可以使用以下命令对发布和项目进行操作：

- **create publication** – 创建一个发布。

- **drop publication** – 删除一个发布及其项目。*drop_repdef* 选项删除相关的复制定义。
- **validate publication** – 验证发布是否至少具有一个项目，并标记该发布以便为其创建新的预订。
- **check publication** – 指出是否可以为一个发布创建预订，并报告它包含的项目数。
- **create article** – 创建项目并将其指派给发布。
- **drop article** – 删除发布中的项目。*drop_repdef* 选项还删除相关的复制定义。

另请参见

- 发布预订命令 （第 8 页）

预订命令

预订会启动数据或存储过程的复制。预订指定表的复制定义名称、函数的复制定义名称或指定一个发布，同时还会指定复制数据的目标数据库。

- 表复制定义的预订将复制数据。
- 函数复制定义的预订将复制存储过程。
- 数据库复制定义的预订将复制数据库或数据库对象。
- 发布的预订将复制该发布中的每个项目所表示的数据。发布也可以包含表示存储过程的项目。

表或函数复制定义的预订可包括 **where** 子句，该子句确定要复制的行或确定是否复制存储过程。

数据库复制定义的预订将预订所有数据。不能使用 **where** 子句为预订数据设置标准。如果需要预订特定的表或函数，可以添加表或函数预订。请参见《Replication Server 管理指南第一卷》的“使用多节点可用性管理复制对象”中的“Concurrent Use of Database, Table, and Function Replication Definitions in an MSA System”（在 MSA 系统中并发使用数据库、表和函数复制定义）。

注意： 发布的预订不能包含 **where** 子句。**where** 子句包含在发布的项目中。

另请参见

- 数据库复制定义命令 （第 5 页）

预订实现

创建表复制定义的预订时，符合预订要求的行将从主表复制到复制表，此过程称为*实现*。实现完成之后，Replication Server 可通过正常复制分发主数据库中的行更改。

如果预订涉及很多行，实现可能长时间保持锁定并造成网络过载。Replication Server 队列也可能被数据占满。为避免这些问题，Replication Server 提供了四种不同的方法来实现预订。

可以使用任何方法预订表复制定义或发布。也可对函数复制定义或数据库复制定义的预订使用非实现或批量实现。

- *原子实现*是表复制定义的缺省方法。Replication Server 使用锁定来选择主表中的行，并通过网络复制它们。在实现过程中主表处于锁定状态，从而保证了数据在主表和复制表之间的一致性。
- 在*非原子实现*中，Replication Server 将在不使用持有锁的情况下选择主表中的行，并通过网络复制它们。在进行非原子实现的过程中，因为主表未锁定，所以复制表可能会经历主表中不存在的可见步骤。
- 在非实现中，主数据和复制数据已处于同步状态。不需要通过网络复制数据或从介质装载数据。创建这类预订期间，不会进行任何更新。
- 在*批量实现*中，将数据从介质中手动卸载，然后再次装载。这是实现涉及大量数据的预订的最有效方法。

有关预订实现方法的详细信息，请参见《Replication Server 管理指南第一卷》。

原子和非原子实现命令
可以使用以下命令来创建预订并初始化复制数据库中的数据：

- **create subscription** – 使用原子实现创建和实现预订。
- **create subscription ... without holdlock** – 使用非原子实现创建和实现预订。

如果使用非原子实现（该方法在选择主数据时不使用锁定），则必须同时使用：

- **set autocorrection** – 防止由于复制表中出现丢失行或重复行而导致的故障。如果在不使用锁定的情况下选择主数据，该数据可能在实现完成之前和正常的事务复制开始之前被更新。

非实现命令
如果复制数据库的数据已经同步，应使用以下命令创建预订：

- **create subscription ... without materialization** – 创建预订，但不实现复制数据库上的数据。

批量实现命令
批量实现用于手动协调预订状态和传输函数复制定义或数据库复制定义的数据。

使用以下批量实现命令：

- **define subscription** – 将预订添加到主 Replication Server 和复制 Replication Server 的系统表。
- **activate subscription** – 启动从主数据库到复制数据库的更新的分配，并将预订状态设置为“活动”。
  使用此命令并检验状态之后，手动将初始数据从介质装载到复制数据库中。使用 **with suspension** 选项可以防止数据在从介质装载完成之前就被应用到复制数据库。
- **validate subscription** – 完成批量实现并将预订状态更改为“有效”。Replication Server 将接到实现已完成的通知。

其它预订命令

了解其它预订命令。

要监控预订的实现或取消实现，请使用：

- **check subscription** – 弄清主数据库或复制数据库中预订的状态。

要从复制数据库中删除预订，请使用：

- **drop subscription** – 从系统表中清除预订信息。

可以选择使用 **drop subscription with purge** 删除与预订相关的复制数据。此过程称为*取消实现*。

发布预订命令

发布预订与复制定义预订使用相同的命令。

- 要使用原子实现、非原子实现或非实现创建发布预订，请使用 **create subscription** 命令。
- 要使用批量实现创建发布预订，请使用 **define subscription** 和其它批量实现命令。

向具有预订的发布中添加项目时，必须刷新发布预订以便将新项目加入预订。此过程称为*重新实现*。

- 对于基本的或非基本的重新实现，应使用带有 **for new articles** 子句的 **create subscription** 命令。
- 如果数据在主数据库和复制数据库之间保持同步，应使用带有 **for new articles** 子句和 **without materialization** 关键字的 **create subscription**。
- 对于批量重新实现，应使用带有 **for new articles** 子句的 **define subscription**，然后使用其它批量实现命令。

另请参见

- 发布命令 （第 5 页）

用户命令

用户必须有 Replication Server 登录帐户才能执行 Replication Server 命令。帐户包括登录名和口令，要连接到 Replication Server 必须提供登录名和口令。

可以使用以下命令来管理用户登录帐户：

- **create user** – 将新用户添加到 Replication Server。
- **alter user** – 更改用户的口令。
- **drop user** – 删除 Replication Server 用户帐户。

可以使用以下命令来管理用户权限：

- **grant** – 授予权限。
- **revoke** – 撤消权限。

使用 **set proxy** 命令可以切换到另一个具有不同权限的用户登录帐户。

每一个权限都允许用户执行一组命令。例如，要创建复制定义，用户必须具有 **create object** 权限。具有“sa”权限的用户可以执行任何 Replication Server 命令。

数据库接口命令

Replication Server 提供了多种方法与数据库连接并自定义在数据库中执行的操作。

开放式体系结构支持由异构数据服务器（包括 Adaptive Server 和其它几种数据服务器）管理的主数据库或复制数据库。

对于每个数据库，您可以:

- 创建或修改 Replication Server 到数据库的连接。请参见“数据库连接命令”。
- 自定义错误处理方法。请参见“错误类命令”。
- 自定义数据库操作。请参见“函数和函数字符串命令”。
- 创建或修改在热备份应用程序中使用的逻辑数据库连接。请参见“热备份数据库命令”。
- 为连接或逻辑连接设置配置参数。请参见“配置命令”。

每个将成为复制事务或存储过程的源的数据库必须具有 Replication Agent。有关详细信息，请参见《Replication Server 管理指南第一卷》。

数据库连接命令

物理数据库*连接*将 Replication Server 连接到包含主数据或复制数据的本地数据库。Replication Server 通过连接与数据库进行消息的分配。

可以使用以下命令来管理数据库连接:

- **create connection** – 创建从 Replication Server 到非 Sybase 数据库的数据库连接。可以使用 **rs_init** 来添加 Adaptive Server 数据库连接。
- **create connection using profile** 子句 – 使用预定义的信息配置 Replication Server 和非 Adaptive Server 复制数据库之间的连接；如果需要，还会修改 RSSD 和复制数据服务器及数据库。
- **alter connection** – 更改或配置数据库连接。
- **drop connection** – 删除数据库连接。
- **suspend connection** – 挂起数据库连接。
- **resume connection** – 重新开始已挂起的连接。

错误类命令

*错误类*是一个名称，该名称之下的错误处理操作（如 **retry** 和 **ignore**）将被指派给特定的数据服务器错误。

使用 **create connection** 命令可以使错误类与数据库相关联。使用 **alter connection** 可以更改错误类。经常可以为指定数据服务器的所有数据库创建一个错误类。

注意： 使用 **rs_init** 添加连接时，将指派 Adaptive Server 数据库的缺省错误类。

使用以下命令管理错误处理操作和错误类：

- **create error class** – 创建错误类。
- **alter error class** – 通过从其它错误类中复制错误操作来修改现有错误类。
- **move primary** – 将错误类或函数字符串类及函数字符串类的任何派生类移动到一个不同的主节点。
- **drop error class** – 删除错误类。
- **assign action** – 向数据服务器错误代码指派操作。

使用存储过程 **rs_init_erroractions** 可以初始化使用现有错误类的错误操作创建的新错误类。有关详细信息，请参见“Adaptive Server 命令和系统过程”。

另请参见

- Adaptive Server 命令和系统过程 （第 445 页）

函数和函数字符串命令

可以使用函数字符串对 Replication Server 进行编程，以在目标数据库中执行自定义命令。

*函数*是与数据服务器操作关联的名称。例如，**rs_insert** 是在表中插入行的系统函数，而 **rs_begin** 是启动事务的系统函数。系统函数可以处理数据（如 **rs_insert**）或控制事务（如 **rs_begin**）。

Replication Server 使用称为*函数字符串* 的模板来构造其提交到数据库的命令。在运行时，函数字符串中的变量被替换为函数的值。

函数字符串类 将函数字符串分组以供数据库使用。例如，函数字符串类可以将供应商数据服务器或将部门表的所有函数字符串归为一组。Replication Server 为 Adaptive Server 和 DB2 数据库提供了函数字符串类。

使用 **create connection** 将函数字符串类与数据库相关联。使用 **alter connection** 更改函数字符串类。

注意： 使用 **rs_init** 添加连接时，将指派 Adaptive Server 数据库的缺省函数字符串类 **rs_sqlserver_function_class**。

可以创建一个从某现有类中继承函数字符串的新函数字符串类。然后，可以根据您的数据库或应用程序的要求，只自定义您要为其指定非缺省行为的函数字符串。

函数字符串类命令

了解函数字符串类命令。

可以使用以下命令对函数字符串类进行操作：

- **create function string class** – 创建函数字符串类。
- **alter function string class** – 更改函数字符串类的继承关系。
- **move primary** – 将错误类或函数字符串类及函数字符串类的任何派生类移动到一个不同的主节点。
- **drop function string class** – 删除函数字符串类。

函数字符串命令

了解函数字符串命令。

可以使用以下命令对函数字符串类中的函数字符串进行操作：

- **create function string** – 创建函数字符串。
- **alter function string** – 替换现有的函数字符串。
- **drop function string** – 删除函数字符串。

函数命令

只有在异步过程调用时才需要函数命令。

可以使用以下命令对用户定义的函数进行操作。

- **create function** – 创建函数。
- **alter function** – 将参数添加到用户定义的函数中。
- **drop function** – 删除函数。

热备份数据库命令

Replication Server *热备份应用程序*维护两个 Adaptive Server 数据库，其中一个数据库充当另一个数据库的备用或备份副本。Replication Server 与活动及备份数据库的连接称为*逻辑连接*。

可以使用以下命令来管理逻辑数据库连接：

- **create logical connection** – 创建逻辑连接。
- **alter logical connection** – 更改逻辑连接的特性。
- **drop logical connection** – 删除逻辑连接。
- **configure logical connection** – 配置逻辑连接。

可以使用以下命令来执行与热备份应用程序相关的任务：

- **switch active** – 更改活动数据库。
- **abort switch** – 如果可能，中止 **switch active** 命令。

- **wait for switch** – 在交互式或基于脚本的 Replication Server 会话中，避免在切换到新的活动数据库的过程完成之前执行命令。
- **wait for create standby** – 在交互式或基于脚本的 Replication Server 会话中，避免 Replication Server 在备份数据库做好操作准备之前接受命令。

网关命令

可以使用网关命令来管理 Replication Server 网关。

Replication Server 网关最大限度减少了显式登录到多个复制服务器、ID Server 和 RSSD 的次数。Replication Server 网关使用 RSSD 主用户名和口令登录到 RSSD；使用 ID Server 用户名和口令登录到 ID Server；使用远程服务器标识 (RSI) 登录到远程 Replication Server 以及使用维护用户 ID 登录到远程 Adaptive Server。在访问 Replication Server 本身时，您无需多次提供此信息。

Replication Server 网关还支持级联连接，Replication Server 可通过这些连接与未直接连接的服务器进行通信。通过使用该功能，您还可以使用单个客户端连接来管理复制域。

可以使用以下命令来管理 Replication Server 网关：

- **connect** – 将 Replication Server 变为其 RSSD、ID Server、远程 Replication Server 或远程数据服务器的网关。
- **show connection** – 列出连接堆栈的内容。
- **show server** – 显示当前的工作服务器。
- **disconnect** – 终止到服务器的连接。

路由命令

路由 是从源（主）Replication Server 到目标 Replication Server 的单向消息流。

Replication Server 通过路由将消息发送到另一个 Replication Server 或从另一个 Replication Server 接收消息。这类消息包括复制事务的数据。路由可以将局域网或广域网上的多个 Replication Server 连接起来。

可以使用以下命令来管理路由：

- **create route** – 创建和配置从当前 Replication Server 到另一个 Replication Server 的路由。
- **alter route** – 更改或重新配置从当前 Replication Server 到另一个 Replication Server 的路由。
- **drop route** – 删除到另一个 Replication Server 的路由。
- **suspend route** – 挂起到另一个 Replication Server 的路由。
- **resume route** – 重新开始已挂起的路由。

系统信息命令

系统信息命令提供有关 Replication Server 的信息。

可以使用以下命令来获取 Replication Server 的相关信息：

- **admin disk_space** – 显示 Replication Server 所访问的每一个磁盘分区的使用情况统计。
- **admin echo** – 返回您所输入的文本以检验 Replication Server 是否正在运行。
- **admin get_generation** – 检索主数据库的生成号。
- **admin health** – 显示 Replication Server 的整体状态。
- **admin log_name** – 显示当前日志文件的路径。
- **admin logical_status** – 显示热备份应用中逻辑连接的状态信息。
- **admin pid** – 显示 Replication Server 的进程 ID。
- **admin quiesce_check** – 确定 Replication Server 中的队列是否已停顿。
- **admin quiesce_force_rsi** – 确定 Replication Server 是否已停顿，并强制它传递出站消息。
- **admin rssd_name** – 显示 Replication Server 系统数据库 (RSSD) 的数据服务器和数据库的名称。
- **admin security_property** – 显示 Replication Server 所支持的基于网络的安全机制和功能。
- **admin security_setting** – 显示 Replication Server 所支持的基于网络的安全功能的状态。
- **admin set_log_name** – 关闭现有的 Replication Server 日志文件并打开一个新的日志文件。
- **admin show_connections** – 显示与 Replication Server 的所有连接的信息。
- **admin show_function_classes** – 显示现有的函数字符串类及其父类的名称，并指出继承的层数。
- **admin show_route_versions** – 显示起止于 Replication Server 的路由的版本号。
- **admin show_site_version** – 显示 Replication Server 的节点版本。
- **admin sqm_readers** – 显示正在读取入站队列的每一个 Replication Server 线程的读取点和删除点。
- **admin stats** – 显示有关 Replication Server 计数器的信息和统计信息。
- **admin stats, backlog** – 报告稳定队列中当前积压的事务。
- **admin stats, {md | mem | mem_in_use}** – 报告有关内存使用情况的信息。
- **admin stats, status** – 显示所有计数器的刷新状态。
- **admin stats, reset** – 重置所有可重置的计数器。
- **admin stats, {tps | cps | bps}** – 报告吞吐量，即每秒的事务数、命令数或字节数。
- **admin time** – 显示 Replication Server 的当前时间。

- **admin translate** – 对特定数据值执行数据类型转换，以带有分隔符的文本格式显示结果。
- **admin version** – 显示 Replication Server 软件版本。
- **admin who** – 显示关于正在 Replication Server 中运行的线程的信息。
- **admin who_is_down** – 显示关于未运行的 Replication Server 线程的信息子集。
- **admin who_is_up** – 显示关于正在运行的 Replication Server 线程的信息子集。

分区命令

Replication Server 将消息存储在稳定队列中，而稳定队列存储在磁盘分区上。入站队列存储从 Replication Agent 接收到的消息；出站队列存储要传输到数据服务器或其它 Replication Server 的消息。

使用 **rs_init** 创建 Replication Server 的初始分区。有关使用 **rs_init** 处理分区的详细信息，请参见 Replication Server 安装和配置指南。

使用以下命令添加、删除或更改分区大小：

- **create partition** – 使分区可供 Replication Server 使用。必须先创建分区，然后才能添加它。

  注意： **create partition** 替代了现有的 **add partition** 命令。为了保持向后兼容，仍然支持 **add partition** 并将其作为 **create partition** 的另一表示形式，但以后不再支持。
- **drop partition** – 从 Replication Server 中删除分区。
- **alter partition** – 更改分区的大小。

有关稳定队列和分区的详细信息，请参见《Replication Server 管理指南第一卷》。

配置命令

Replication Server 启动后，将从系统表或配置文件中读取配置参数。配置参数可能是静态的，也可能是动态的。可以在 Replication Server 运行期间更改动态参数，但更改静态参数之后必须重新启动 Replication Server。

可以使用以下命令来配置 Replication Server：

- **alter connection** 和 **configure connection** – 更改从 Replication Server 到数据库的连接的特性。
- **configure logical connection** – 为热备份应用中的逻辑连接更改 Replication Server 配置。
- **configure replication server** – 更改路由和连接的 Replication Server 参数和缺省参数。
- **alter route** 和 **configure route** – 更改路由的特性。路由将一个 Replication Server 与另一个 Replication Server 相连接。

使用 **create route** 和 **create connection** 创建路由和连接时，同时也会设置配置参数。

有关详细信息，请参见《Replication Server 管理指南第一卷》。

系统管理命令

可以使用以下命令来执行系统管理任务，并排除系统故障的遗留问题。必须具有“sa”权限才能执行这些命令。

警告! 这些命令中的许多命令只能在非常有限的情况下谨慎使用。使用它们之前，请仔细查阅相关文档。

- **alter queue** – 指定稳定队列在遇到超过 16K 个字节的大消息时的行为。只有在 Replication Server 版本是 12.5 或更高版本并且节点版本是 12.1 或更低版本时才可以使用。
- **resume distributor** – 为数据库连接恢复已挂起的分发器线程。
- **shutdown** – 关闭 Replication Server。
- **suspend distributor** – 挂起与数据库连接的分发器线程。
- **sysadmin apply_truncate_table** – 为特定表的现有预订打开或关闭“预订 Truncate Table”选项，以便启用或禁用 **truncate table** 的复制。
- **sysadmin dropdb** – 删除从 ID Server 到数据库的引用。
- **sysadmin dropldb** – 删除从 ID Server 到逻辑数据库的引用。
- **sysadmin drop_queue** – 删除稳定队列。
- **sysadmin droprs** – 删除从 ID Server 到 Replication Server 的引用。
- **sysadmin dump_file** – 指定备用日志文件以供转储稳定队列时使用。
- **sysadmin dump_queue** – 转储稳定队列的内容。
- **sysadmin erssd** – 用于检查 ERSSD 文件位置和备份配置，对 ERSSD 文件进行碎片整理，移动 ERSSD 文件或执行非预定 ERSSD 备份。
- **sysadmin fast_route_upgrade** – 将路由版本更新为主 Replication Server 或复制 Replication Server 两者中较低的节点版本。
- **sysadmin hibernate_off** – 关闭 Replication Server 休眠模式并将其恢复为活动状态。
- **sysadmin hibernate_on** – 打开 Replication Server 休眠模式（即挂起 Replication Server）。
- **sysadmin log_first_tran** – 将数据服务器接口 (DSI) 队列中的第一个事务写入例外日志中。
- **sysadmin purge_all_open** – 从入站队列中清除所有打开的事务。
- **sysadmin purge_first_open** – 从入站队列中清除第一个打开的事务。
- **sysadmin purge_route_at_replicate** – 删除复制节点处从 Replication Server 到主 Replication Server 的所有引用。
- **sysadmin restore_dsi_saved_segments** – 恢复积压的事务，以便将它们重新应用到数据库。

- **sysadmin set_dsi_generation** – 更改 RSSD 中的数据库生成号，以防止在恢复复制数据库之后 Replication Server 再次应用稳定队列中的事务。
- **sysadmin site_version** – 设置节点的版本级别。
- **sysadmin sqm_purge_queue** – 删除 Replication Server 接口 (RSI) 稳定队列中的所有消息。
- **sysadmin sqm_unzap_command** – 恢复稳定队列中已删除的消息。
- **sysadmin sqm_zap_command** – 删除稳定队列中的单个消息。
- **sysadmin sqt_dump_queue** – 转储每个入站队列或 DSI 队列的事务缓存。
- **sysadmin system_version** – 为复制系统设置最低 Replication Server 版本级别。

恢复命令

重新装载数据库之后或 Replication Server 稳定队列失败时，使用以下命令协调恢复。

警告！ 这些命令中的许多命令只能在非常有限的情况下谨慎使用。使用它们之前，请确保仔细查阅相关文档。

- **allow connections** – 为指定的数据库将 Replication Server 置于恢复模式下。
- **ignore loss** – 检测到丢失之后仍允许 Replication Server 接受消息。
- **rebuild queues** – 重建 Replication Server 稳定队列。
- **resume log transfer** – 允许 RepAgent 线程连接到 Replication Server。
- **resume queue** – 接收到超过 16K 个字节的消息后重新启动已停止的稳定队列。仅适用于 Replication Server 版本为 12.5 或更高版本且节点版本为 12.1 或更低版本的情况。
- **set log recovery** – 将数据库的 Replication Server 置于日志恢复模式。
- **suspend log transfer** – 将 Replication Server 与 RepAgent 断开连接，并禁止它们进行连接。

有关详细的恢复过程，请参见《Replication Server 管理指南第二卷》。

主题

了解数据类型、标识符、保留字、Adaptive Server 支持以及混合版本环境。

数据类型

了解 Replication Server 支持的 Sybase 数据类型。

表 1. Replication Server 所支持的数据类型

数据类型类	数据类型
精确数值（整数）	*bigint*、*int*、*smallint*、*tinyint*、*unsigned bigint*、*unsigned int*、*unsigned smallint*、*unsigned tinyint*、*rs_address*
精确数值（十进制）	*decimal*、*numeric*、*identity*
近似值（浮点数）	*float* 和 *real*
字符	*char(n)*、*varchar(n)*、*text*、*opaque*
货币	*money* 和 *smallmoney*
日期/时间	*datetime*、*smalldatetime*、*date*、*time*、*timestamp*、*bigdatetime*、*bigtime*
二进制	*binary(n)*、*varbinary(n)*、*image*、*rawobject*、*rawobject in row*
位	*bit*
Unicode	*unichar(n)*、*univarchar(n)*、*unitext*
Java	*rawobject*、*rawobject in row*
数据类型定义	请参见“数据类型定义”。

RCL 间接支持下列 Sybase 数据类型：

- *double precision*
- *nchar*、*nvarchar*

不支持下列数据类型：

- *float* 数据类型的可选精度参数
- 精确十进制数据类型的可选精度和标度参数

在使用 Replication Server 支持的数据类型表中所示的一种支持的数据类型创建复制定义时，可以复制具有不支持的数据类型的列中的数据。例如，要复制 *double precision*

列，请在复制定义中将该列定义为 *float* 类型。要复制具有用户定义的数据类型的列，请使用复制定义中的基本数据类型。

要复制 Adaptive Server 上存储在具有 *nchar* 或 *nvarchar* 类型的列中的数据，请分别使用 *char* 和 *varchar* Replication Server 数据类型。唯一的区别在于 *nchar* 和 *nvarchar* 中的长度单位是指 Adaptive Server 的本机字符集中的字符数，而 *char* 和 *varchar* 中的长度单位始终为字节。

要得到对应的 Replication Server *char* 和 *varchar* 数据类型的长度，可用 Adaptive Server 全局变量 *@@ncharsize* 的值乘以 *nchar* 或 *nvarchar* 数据类型的声明长度。

例如，如果 *@@ncharsize* 为 1（适用于所有的单字节字符集，如 iso_1、cp850、cp437、roman8 和 mac），则存在一对一的对应关系，而且声明的长度相同。如果 *@@ncharsize* 为 2（适用于某些多字节字符集，如 Shift-JIS 和 EUC-JIS），应将 *nchar* 和 *nvarchar* 数据类型的声明长度乘 2，并在复制定义中将它们声明为 *char* 和 *varchar*。

下面各节对支持的数据类型进行了说明。有关 Adaptive Server 数据类型的详细信息，请参见《Adaptive Server Enterprise 参考手册》。

Replication Server 支持非 Sybase 数据服务器的一组数据类型定义，通过这些定义您可以将一个数据类型的列值复制到复制数据库中其它数据类型的列。有关异构数据类型支持 (HDS) 的详细信息，请参见《Replication Server 管理指南第一卷》。

精确数值（整数）数据类型

了解精确数值（整数）数据类型。

Replication Server 支持下列精确数值（整数）数据类型：

- *bigint* – -263 和 +263 - 1（-9,233,372,036,854,775,808 和 +9,233,372,036,854,775,807）之间的整数，含这两个数
- *int* – -2^{31} 和 $+2^{31} - 1$（-2,147,483,648 和 +2,147,483,647）之间的整数，含这两个数
- *smallint* – -2^{15} 和 $+2^{15} - 1$（-32,768 和 +32,767）之间的整数，含这两个数
- *tinyint* – 0 和 255 之间的正整数，含这两个数
- *unsigned bigint* – 0 和 18,446,744,073,709,551,615 之间的整数，含这两个数
- *unsigned int* – 0 和 4,294,967,295 之间的整数，含这两个数
- *unsigned smallint* – 0 和 65535 之间的整数，含这两个数
- *unsigned tinyint* – 0 和 255 之间的整数，含这两个数

使用基本数据类型 *int* 的 *rs_address* 数据类型用于一种特殊的预订解析方法。请参见《Replication Server 管理指南第一卷》以了解 *rs_address* 数据类型的详细信息。

另请参见

- create subscription （第 280 页）

精确数值（十进制）数据类型

了解精确数值（十进制）数据类型。

Replication Server 支持下列精确数值（十进制）数据类型：

- *decimal* – -10^{38} 和 $10^{38}-1$ 之间的精确十进制数值，含这两个数。
- *numeric* – -10^{38} 和 $10^{38}-1$ 之间的精确十进制数值，含这两个数。

创建复制定义时，请在 *numeric* 数据类型声明中省略长度和精度。Replication Server 可以在不影响精度的情况下处理 *numeric* 值。

注意： 如果您要在复制定义的 **where** 子句中使用 numeric 数据类型，该值必须包括精度信息。

Identity 列使用 *numeric* 作为基本数据类型，包括 1 到 $10^{38}-1$ 之间标度为 0 的精确十进制数，含这两个数。

为包含 *identity* 列的表创建复制定义时，应将“identity”指定为该列的数据类型。

以下命令将在执行 **insert** 命令之前应用到被复制表：

```
set identity_insert table_name on
```

以下命令将在执行 **insert** 命令之后应用到被复制表：

```
set identity_insert table_name off
```

从不使用 **update** 命令更新 *Identity* 列。

如果复制数据服务器为 Adaptive Server，并且表中包含 *identity* 列，维护用户在复制数据库中必须是该表的所有者（或者必须是“dbo”用户，或者使用“dbo”登录名作为别名），才可以使用 Transact-SQL® **identity_insert** 选项。

近似值（浮点数）数据类型

了解数值（浮点数）数据类型。

共有两种近似值（浮点数）数据类型：

- *float* – 正的或负的浮点数值。精度和有效数字的位数取决于计算机。存储大小为 8 个字节。
- *real* – 与 *float* 相似，但其存储大小为 4 个字节。

字符数据类型

了解字符数据类型。

注意： Unicode 数据类型 *unichar*、*univarchar* 和 *unitext* 具有与等效 *char*、*varchar* 和 *text* 相同的属性。

- *char(n)* – 最多 32,768 个单字节字母、符号和数字的任意组合。可以使用 *n* 来指定字符串的最大大小。*char* 值可以包含 0 个字符，但是 *n* 必须介于 1 到 32,768 之间。多字节字符串不能超过 32,768 个字节。
- *varchar(n)* – 最多 32,768 个单字节字母、符号和数字的任意组合。*varchar* 值定义为允许空值时可以包含 0 个字符，但是 *n* 必须介于 1 和 32,768 之间。

  char 和 *varchar* 数据的差别在于其值在 Adaptive Server 数据库中的存储方式不同。Replication Server 将它们视为等效类型，但仍然保留其间的差别，以便在主数据库和复制数据库中使用相同的存储方法。
- *text* – 长度最大可达 2,147,483,647 个字节的可变长度字符列。

  Replication Server 15.1 支持大对象 (LOB) 数据类型（如带有文本指针的 *text*、*unitext* 和 *image* 数据类型以及不带文本指针的 *text*、*unitext* 和 *image* 数据类型）之间的数据类型转换。

字符数据的条目格式

char、*varchar* 和 *text* 的实际值（或等效数据类型）必须用单引号引起来。

您可以通过两种方式在 *char* 和 *varchar* 的实际值中嵌入单引号。使用两个连续的引号可以代表一个字符串本身带有的引号，如下例所示：

```
'''You can have cake if you bake it,'' Ed claims.'
```

首尾的引号表示该字符串的首尾分界。里面的两对引号将解释为该字符串本身带有的单引号。

Replication Server 用字符值替换函数字符串模板中的变量时，将生成单引号。

另请参见

- create function string （第 236 页）

货币数据类型

货币数据类型对货币值规定了固定的精度值。

- *money* – -922,337,203,685,477.5808 和 922,337,203,685,477.5807 之间的货币值，精确到货币单位的 1/10000。存储大小为 8 个字节。
- *smallmoney* – -214,748.3648 和 214,748.3647 之间的货币值，精确到一个货币单位的 1/10000。存储大小为 4 个字节。

货币数据的条目格式

在 *money* 和 *smallmoney* 实际值之前加上美元符号 ($) 可以将它们与浮点数据类型区分开来。如果是负值，应将负号放在美元符号后。

Replication Server 在将 *money* 和 *smallmoney* 值替换到函数字符串输出模板中时，会输出美元符号。

日期/时间以及日期和时间数据类型

了解日期和时间数据类型。

Replication Server 支持下列日期和时间数据类型：

- *datetime* – 1753 年 1 月 1 日到 9999 年 12 月 31 日之间的日期和时间。存储大小为 8 个字节：4 个字节代表基准日期 1900 年 1 月 1 日之前或之后的天数，另 4 个字节代表时间，精确到 1/300 秒。基准日期之前的日期存储为负值。
- *smalldatetime* – 1900 年 1 月 1 日到 2079 年 6 月 6 日之间的日期和时间，精确到分钟。存储大小为 4 个字节：一个小整型数值代表 1900 年 1 月 1 日之后的天数，另一个小整型数值代表午夜以后的分钟数。
- *date* – 0001 年 1 月 1 日到 9999 年 12 月 31 日之间的日期。存储大小为 4 个字节。基准日期之前的日期存储为负值。
- *time* – 12:00:00 AM 到 11:59:59.999 PM 之间的时间。存储大小为 4 个字节。
- *bigtime* – 每日时刻，其中包含小时、分钟、秒钟和秒的小数部分，对应于 Sybase IQ 中的 *TIME* 数据类型。小数部分存储为 6 位小数。*bigtime* 值需要 8 个字节的存储空间。ODBC 标准将 *bigtime* 数据类型限制为精确到秒。因此，不要在要求的精度比秒更高的 **WHERE** 子句比较中使用 *bigtime* 数据类型。

  bigtime 的有效范围为 12:00:00.000000AM 到 11:59:59.999999PM
- *bigdatetime* – 时间点，其中包含年、月、日、小时、分钟、秒钟和秒的小数部分，对应于 Sybase IQ 中的 *TIMESTAMP* 数据类型。小数部分存储为 6 位小数。日必须为非零值。*bigdatetime* 值需要 8 个字节的存储空间。

  bigdatetime 的有效范围为从 0001 年 1 月 1 日到 9999 年 12 月 31 日以及从 12:00:00.000000AM 到 11:59:59.999999PM。超出范围 1600-02-28 23:59:59 到 7911-01-01 00:00:00 的 *bigdatetime* 数据的显示可能不完整，但完整的 *bigdatetime* 值存储在数据库中。
- *timestamp* – 将 *varbinary*(8) 作为基本数据类型。可通过状态位区分 *timestamp* 和 *varbinary*。

  对于 Replication Server 15.1，*timestamp* 作为 *timestamp* 传播，对于 Replication Server 15.0.1 或更低版本，则作为 *varbinary* 传播。

注意： 只有 ASE 15.0.2 或更高版本支持向 timestamp 列复制。

日期/时间值的条目格式

应将 *datetime* 和 *smalldatetime* 的值作为字符串输入，并用单引号引起来。

Replication Server 在将 *datetime* 值替换到函数字符串输出模板中时，会用单引号将 *datetime* 值引起来。在创建包括 *datetime* 变量的函数字符串时，请务必考虑到这一点。

数据的日期和时间部分是分别识别的，因此，时间可以在日期之前，也可以在日期之后。如果省略时间，Replication Server 将假定时间为午夜 (12:00:00:000AM)。如果省略日期，Replication Server 将假定日期为 1900 年 1 月 1 日。

请按照下面的一般规则输入时间：

- 小时的范围是 0 到 23；分钟和秒的范围是 0 到 59；毫秒的范围是 0 到 999。
- 值中必须有冒号、“AM”或“PM”指示符才能识别为时间值。
- 添加“AM”或“PM”时留不留空格都可以。12AM 代表午夜，12PM 代表中午。如果指定 AM，小时必须在 1 到 12 之间（可以用 0 代替 12）。如果指定 PM，小时必须在 13 到 23 之间。
- 毫秒前可以使用冒号或句点。如果使用冒号，该数表示千分之几秒。如果使用句点，一个数位表示十分之几秒，两个数位表示百分之几秒，三个数位表示千分之几秒。例如，“12:30:20:1”表示 12 点 30 分 20 又千分之一秒；而“12:30:20.1”则表示 12 点 30 分 20 又十分之一秒。
- 您可以省略时间值的任何部分。如果省略秒，必须同时省略毫秒。如果省略分钟，必须同时省略秒和毫秒。Replication Server 将任何省略的部分都假定为零。

以下是一些实际的时间值示例：

```
2:00
14.30
14:30:20
14:30:20:500
4pm
11:41:36 AM
12:48:5.333 pm
```

输入日期时，可以按任何顺序输入月、日和年，但应遵循以下规则：

- 可以使用 1 – 12 的数字输入月份，或者使用英语中的月份名称或其三个字符的缩写形式。
- 如果使用数字月份，必须用斜杠 (/)、连字符 (-) 或句点 (.) 分开日期的各部分。日期部分的顺序必须为月-日-年。
- 以下示例列出了输入日期 1998 年 3 月 15 日的不同方法：

  ```
  3-15-1998
  March-15-1998
  March 15 1998
  15/March/1998
  March.15.1998
  ```
- 您可以将英语的月份缩写为 3 个字符。不需要区分大小写。

  ```
  JAN 9 1998
  31 oct 1997
  ```
- 使用字母月份时，月份和日期之后可以跟一个逗号。以下是有效的日期：

  ```
  Nov 17, 1997
  1997 Nov, 17,
  17 Nov, 1997
  ```
- 您可以用一、二或四位数字输入年份。小于 50 的一位或两位数年份被假定为在本（二十一）世纪。大于或等于 50 的两位数年份在上一个（二十）世纪。
- 四位数的年份放在日期值中的任何位置上都可以识别。两位数的年份必须放在日期（月份中的某一天）之后。
- 如果使用字母月份和四位数的年份，可以省略日期（月份中的某一天）。日期（月份中的某一天）缺省为该月的第一天。不能在月份名称后使用逗号之外的其它分隔符。

Replication Server 将下面的日期解释为 1998 年 5 月 1 日:

```
May 1998
1998 MAY
may, 1998
```

下面的示例显示了如何在复制定义、函数复制定义和预订中使用 *bigdatetime* 和 *bigtime*。在下面的示例中:

- PDS – 主数据服务器
- pdb1 – 主数据库
- RDS – 复制数据服务器
- rdb1 – 复制数据库
- tb1 – 表
- col1、col2、col3 – 列
- rep1 – 复制定义
- func1 – 函数复制定义
- sub1 – 预订

示例 1
在复制定义中使用数据类型。

```
create replication definition rep1
with primary at PDS.pdb1
with all tables named tb1
(col1 int, col2 bigdatetime, col3 bigtime)
primary key (col1)
```

示例 2
在函数复制定义中使用数据类型。

```
create function replication definition func1
with primary at PDS.pdb1
(@par1 int, @par2 bigdatetime, @par3 bigtime)
searchable parameters (@par1)
```

示例 3
在预订中使用数据类型。

```
create subscription sub1 for rep1
with replicate at RDS.rdb1
where col3 = '14:20:00.010101'
without materialization
```

二进制数据类型

了解二进制数据类型。

二进制数据类型有:

- *binary(n)* – 最多 32,768 个字节的固定长度二进制数据。*binary* 数据类型用于存储编程代码或图像，不存储数字值。可用 *n* 指定值的最大字节长度。*binary* 值可以包含 0 个字节，但是 *n* 必须介于 1 到 32,768 之间。
- *varbinary(n)* – 最多 32,768 个字节的可变长度二进制数据。*varbinary* 数据类型用于存储编程代码或图像，不存储数字值。可用 *n* 指定值的最大字节长度。*varbinary* 值可以包含 0 个字节，但是 *n* 必须介于 1 到 32,768 之间。

  binary 和 *varbinary* 数据的差别在于其值在 Adaptive Server 数据库中的存储方式不同。Replication Server 将它们视为等效类型，但仍然保留其间的差别，以便在主数据库和复制数据库中使用相同的存储方法。
- *rawobject in row* – 255 个字节的可变长度二进制数据。*rawobject in row* 数据类型用于存储分配给表的数据页内的序列化 Java 值。

  Replication Server 处理 *rawobject in row* 数据的方式与处理 *varbinary* 数据完全相同。*rawobject in row* 的基本数据类型为 *varbinary(255)*。
- *rawobject large in row* – 32,768 个字节的可变长度二进制数据。*rawobject large in row* 数据类型用于存储分配给表的数据页内的序列化 Java 值。

  Replication Server 处理 *rawobject large in row* 数据的方式与处理 *varbinary* 数据的方式相同。*rawobject large in row* 的基本数据类型为 *varbinary(32768)*。
- *image* – 长度最大为 2,147,483,647 个字节的可变长度二进制列。

  Replication Server 15.1 支持 LOB 数据类型（如带有文本指针的 *text*、*unitext* 和 *image* 数据类型和不带文本指针的 *text*、*unitext* 和 *image* 数据类型）之间的数据类型转换。
- *rawobject* – 长度最大为 2,147,483,647 个字节的可变长度二进制列。*rawobject* 数据类型用于存储序列化的 Java 值。Replication Server 不支持 *rawobject* 数据的数据类型转换。这意味着如果复制定义将某列声明为 *rawobject* 类型，则主表的列必须为 *rawobject* 类型。

  Replication Server 处理 *rawobject* 数据的方式与处理 *image* 数据完全相同。*rawobject* 的基本数据类型为 *image*。

另请参见

- Java 数据类型 （第 26 页）

二进制数据的条目格式

您可以使用十六进制数 0-9 和 A-F（或者 a-f）输入 *binary*、*varbinary*、 *image*、*rawobject*、*rawobject in row* 及 *rawobject large in row* 的实际值。

每个字节由两个十六进制数字表示，并在整个值前加上“0x”。下面的示例是一个 10 个字节的 *binary* 字符串：

```
0x010305070B0D1113171D
```

Replication Server 在将 *binary* 值替换到函数字符串输出模板中时输出前缀“0x”。

bit 数据类型

bit 数据类型用于布尔值。

- *bit* – 1 或 0。1 和 0 以外的其它整数值都解释为 1。

Unicode 数据类型

Replication Server 支持三种 Unicode 数据类型：*unichar(n)*、*univarchar(n)* 和 *unitext*。利用 Unicode，可以在同一台数据服务器上混用不同语言组中的语言。

Unicode 数据类型的作用与其等效 Replication Server 数据类型完全一样。

- *unichar* –> *char*
- *univarchar* –> *varchar*
- *unitext* –> *text*

除了 Unicode 值始终存储在 UTF-16 中之外，Unicode 数据类型共享其等效数据类型的语法和语义，不管 Replication Server 的缺省字符集为何都均为如此。*unichar(n)* 是固定宽度、不可为空的数据类型。*univarchar(n)* 是可变宽度、可为空的数据类型。对于 *unichar(n)* 和 *univarchar(n)*，使用 *n* 指定 Unicode 字符数。*unitext* 是可变宽度、可为空的数据类型。

您可以：

- 将 *unichar(n)*、*univarchar(n)* 和 *unitext* 列复制到复制数据库和备用数据库中
- 在复制定义的主键中使用 *unichar(n)* 和 *univarchar(n)* 列
- 在复制定义以及相关预订和项目的 **where** 子句中将 *unichar(n)* 和 *univarchar(n)* 列用作可搜索列
- 在函数复制定义以及相关预订和项目的 **where** 子句中将 *unichar(n)* 和 *univarchar(n)* 列用作可搜索列
- 在复制到异构数据服务器或从中复制时，使用 *unichar(n)*、*univarchar(n)* 和 *unitext* 列

使用与 *text* 相同的方式：

- *unitext* 列不能是复制定义中的主键的一部分。
- 不能在复制定义中将 *unitext* 列指定为可搜索列。
- 不能在函数复制定义中将 *unitext* 列指定为可搜索列。
- 不能将 *unitext* 数据类型作为基本数据类型或数据类型定义使用，也不能将其作为列级转换或类级别转换的源或目标使用。

要正确复制 *unichar* 和 *univarchar* 列，必须对 Replication Server 进行如下配置：

```
RS_charset=utf8
```

如果 Replication Server 的缺省字符集不是 UTF-8，Replication Server 只能复制 ASCII-7 代码范围内的 *unichar* 和 *univarchar* 字符。

升级问题

要完全支持 *unichar* 和 *univarchar* 数据类型，主 Replication Server 和复制 Replication Server 必须运行 12.5 版或更高版本。

为了完全支持 *unitext* 数据类型，主 Replication Server 和复制 Replication Server 的版本都必须为 15.0.1 版或更高版本，路由版本必须是 15.0.1 或更高版本，LTL 的版本必须是 700 或更高版本。如果在连接源时，LTL 的版本低于 700，RepAgent 会将 *unitext* 列转换为 *image*。

RM 路由升级功能从上游 Replication Server 复制引用 *unichar*、*univarchar* 和 *unitext* 数据类型的复制定义。

混合版本问题

在混合版本环境中，主 Replication Server 和复制 Replication Server 之间的路由版本决定了所支持的功能。

- 仅 Adaptive Server 15.5 及更高版本支持 *bigdatetime* 和 *bigtime*。如果主数据服务器至少为 Adaptive Server 15.5，并且：
  - 主 Replication Server 和复制 Replication Server 是版本 15.5 或更高版本，而复制 Adaptive Server 不支持这些数据类型，您可以创建包含这两个数据类型中的每一个到 *varchar* 数据类型的映射的复制定义。也可以在复制定义中使用 *varchar* 数据类型代替这两个数据类型。
  - 主 Replication Server 是版本 15.5 或更高版本，而复制 Replication Server 和 Adaptive Server 不支持这些数据类型，请在复制定义中使用 *varchar* 数据类型代替这两个数据类型。
  - 主 Replication Server、复制 Replication Server 和复制 Adaptive Server 不支持这些数据类型，RepAgent 会自动将 *varchar* 数据类型发送给 Replication Server。
- 要成功复制带引号的标识符，连接到复制数据服务器的主 Replication Server 和 Replication Server 版本必须为 15.2。不过，路由中的中间 Replication Server 可以是较早的版本。
- 使用 *unitext* 列创建的复制定义不会传播到 Replication Server 12.6 版和更低版本。
- 不能更改 Replication Server 12.6 版和更低版本所预订的复制定义以添加 *unitext* 列。
- 如果删除了 *unitext* 列，则会将使用 *unitext* 列创建的复制定义传播到 Replication Server 12.6 版或更低版本。

Java 数据类型

了解 Java 数据类型。

Java 列作为下列三种 Replication Server 数据类型中的任何一种通过复制系统进行传递：

- 作为 *rawobject*，在此数据类型中，信息存储在数据库中的一个单独位置，与存储 *image* 数据的方式相同。*rawobject* 的基本数据类型为 *image*。*rawobject* 是 Replication Server 中的 Java 列的缺省数据类型。
- 作为 *rawobject in row*，在此数据类型中，信息存储在数据库中分配给该表的连续数据页上，与存储 *char* 数据的方式相同。*rawobject in row* 的基本数据类型是 *varbinary(255)*。
- 作为 *rawobject large in row*，在此数据类型中，信息存储在数据库中分配给该表的连续数据页上，与存储 *char* 数据的方式相同。*rawobject large in row* 的基本数据类型为 *varbinary(32768)*。

rawobject、*rawobject in row* 和 *rawobject large in row* 数据类型仅与其基本数据类型兼容。它们互不兼容。不能将一种 Java 数据类型复制为另一种 Java 数据类型，反之亦然。

rs_subcmp 调和实用程序将 Java 数据类型作为其基本数据类型处理。

opaque 数据类型

opaque 数据类型处理 Replication Server 当前不支持的数据类型。RepAgent 为 Replication Server 提供可直接应用到目标数据服务器的格式化数据。此类数据类型的示例有 Oracle 的 *anydata* 数据类型，以及 Microsoft SQL Server 的 *sql_variant* 数据类型。

限制

opaque 数据类型的限制包括：

- 无法在复制定义、预订和项目的可搜索列和 **where** 子句中使用 *opaque* 数据类型。
- 无法对 *opaque* 数据类型使用 **map to** 子句。
- *opaque* 数据类型列或参数存在于复制定义中时，无法使用动态 SQL 功能。
- 如果函数字符串具有远程过程调用 (RPC)，则无法使用 *opaque* 数据类型。
- 无法将字符转换或字节顺序转换应用于 *opaque* 数据。

混合版本支持

若要支持 opaque 数据类型，主 Replication Server 和复制 Replication Server 的节点版本不可低于 15.1，LTL 的版本不可低于 710。

数据类型定义

Sybase 提供了一组用户定义的数据类型和数据类型类。在以下服务器之间复制数据时，可以使用它们更改列值的数据类型：

- Sybase 数据服务器
- Sybase 数据服务器和非 Sybase 数据服务器
- 同构的非 Sybase 数据服务器
- 异构的非 Sybase 数据服务器

数据类型定义使用 Replication Server 本机的基本数据类型来描述非 Sybase 数据类型。基本数据类型决定与数据类型定义相关联的最大和最小长度，并提供其它数据类型属性的缺省值。基本数据类型还定义了与数据类型定义相关联的分隔符。

每个数据类型类都包含特定数据服务器的数据类型定义。这些数据类型类有：

- Adaptive Server – **rs_sqlserver_udd_class**
- SQL Anywhere® – **rs_asa_udd_class**
- DB2 – **rs_db2_udd_class**
- Microsoft SQL Server – **rs_msss_udd_class**
- Oracle – **rs_oracle_udd_class**

有关每种数据类型类支持的数据类型定义的列表和说明，请参见《Replication Server 异构复制指南》。

标识符

标识符是对象的符号名，这些对象包括数据库、表、复制定义、发布、预订、函数、参数、函数字符串变量等。

这些对象的标识符长度为 1 到 255 个字节：

- 表
- 列
- 过程
- 参数
- 函数 – 作为函数复制定义或内部函数的一部分

  注意： **create function**、**alter function** 和 **drop function** 命令不支持长标识符。函数名称和这些命令参数均不能超过 30 个字节。
- 函数字符串
- 复制定义 – 包括表复制定义、函数复制定义和数据库复制定义
- 项目
- 发布
- 预订ン

所有其它标识符的长度为 1 到 30 个字节。

如果标识符未用引号引起来，其第一个字符必须为 ASCII 字母。后续的字符可以是 ASCII 字母、数字、$ 或 _ 字符。不允许嵌入空格。

以字符“rs_”开头的标识符是 Replication Server 的保留标识符。有关其它保留字的列表，请参见“保留字”。

Replication Server 函数和 Adaptive Server 存储过程的参数名称是唯一能够以 @ 字符开头的标识符。

- Replication Server 函数参数名称最多包含 256 个字节（包括 @ 字符）。
- Adaptive Server 存储过程参数名称最多可包含 255 个字节（包括 @ 字符）。

要用保留字作标识符，可将标识符用双引号引起来。使用引号时，还可以嵌入空格，或使用在其它情况下禁用的字符，如 !@#$%^&*() 以及 8 位和多字节字符。Replication Server 会删去标识符末尾的所有尾随空白，即使括在引号里的也不例外。例如：

```
check subscription "publishers_sub"
     for "publishers_rep"
with replicate at "SYDNEY_DS"."pubs2"
```

警告！ Adaptive Server 允许您在将 **quoted_identifier** 设置为 on 时将标识符放在引号内。这使您能够使用保留字作为 Adaptive Server 对象名。但是，Replication Server 不能识别它发往 Adaptive Server 的命令中括在引号内的标识符，因此您不能使用 Transact-SQL 关键字作为复制的 Adaptive Server 对象的名称。如有必要，可以更改函数字符串，将复制对象的标识符用引号引起来。

应在函数字符串模板中的变量名两边加上问号。例如，可在函数字符串中使用下面这个变量名指代主数据库：

```
?rs_origin_db!sys?
```

或者，使用带引号的标识符：

```
?”rs_origin_db”!sys?
```

标识符的名称空间

标识符的名称空间是指 Replication Server 能够识别它的范围。

例如，数据服务器名具有全局名称空间，这是因为在整个复制数据系统中只有一台数据服务器可以使用该名称。而对于列名，这个范围就是表；必须用表名来限定它，这是因为多个表可以具有同名的列。

Replication Server 标识符名称空间表显示了每个标识符的 Replication Server 名称空间。

表 2. Replication Server 标识符的名称空间

标识符类型	名称空间
项目	发布
列	表
数据服务器	全局
database	数据服务器
错误类	全局
函数字符串类	全局

标识符类型	名称空间
功能	复制定义。如果用户定义的函数用于在 Adaptive Server 数据库中执行的异步过程，这些函数就必须具有全局唯一的名称，除非在该过程中指定表复制定义。
函数复制定义	全局
参数	功能
发布	主数据服务器和主数据库
复制定义	全局
Replication Server	全局
预订	复制定义、复制数据服务器和数据库。预订必须具有全局唯一的名称。
user	Replication Server
变量	函数或表

对于复制定义和其它具有全局范围的 Replication Server 对象，应当采用适当的命名约定，以确保其名称在全局名称空间中保持唯一。

警告！ 必须谨慎管理具有全局名称空间的标识符。Replication Server 不能立即检测到全局名称空间中所有重复的名称，但稍后可能出现错误。

有时，必须限定具有非全局名称空间的标识符。例如，许多 Replication Server 命令的语法中都包括一个 **at** 子句，该子句标识表所在的数据服务器和数据库：

```
at data_server.database
```

在配置正确的系统中，所有的服务器都使用相同的排序顺序。如果服务器使用的排序顺序不同，不同的服务器比较标识符时将不一致，这可能会导致网络中出现异常。

保留字

了解 Replication Server 保留字。

Replication Server 的保留字表中的保留字是保留的 Replication Server 关键字。虽然这些字以小写显示，但 Replication Server 并不区分大小写。因此，大小写字母的所有组合都是保留字。Replication Server 还保留了所有以“rs_”开头的关键字和标识符。

表 3. Replication Server 保留字

	保留字
A	**abort**、**_aco**、**action**、**activate**、**active**、**add**、**_add_recov_pending**、**admin**、**_af**、**after**、**all**、**allow**、**alter**、**always_rep**、**always_replicate**、**_alt_attr2**、**_alter_attributes2**、**_alter_col_objid**、**and**、**_ap**、**_apd**、**article**、**articles**、**_apd**、**append**、**applied**、**_ar**、**_arp**、**article**、**articles**、**as**、**assign**、**at**

	保留字
B	**before、begin、_bf、_bg**
C	**changed、_ch、check、ci、class、_cm、columns、commit、configure、connect、connection、connections、connector、controller、create**
D	**database、datarow、dataserver、ddl、debug、define、definition、deletelen、deliver、description、disconnect、display_only、distribute、distribution、distributor、_dln、_dr、drop、drop_repdef、_ds、dsi_suspended、dump、dynamic**
E	**enable、error、exec、execute、expand**
F	**_fi、first、for、from、function、functions**
G	**get、grant**
H	**_ha、hastext、holdlock**
I	**ignore、in、incrementally、init、installjava、internal_use_only、into、_instj、_isb、_isbinary**
J	**_jar**
K	**key**
L	**language、large、last、load、log、logical、loss**
M	**maintenance、map、marker、materialization、message、_mbf、min_before、min_row、minimal、move、_mr**
N	**name、named、_ne、never_rep、new、next、no、no_password、none、not、notrep、nowait、npw、_nr、_nu、null、nullable**
O	**of、off、offset、on、only、open_xact、or、_os、osid、output、overwrite、owner**
P	**parameters、parent、partialupd、partition、passthru、password、primary、procedure、procedures、profile、_pu、public、publication、purge**
Q	**queue、queues、quoted**
R	**_rar、rebuild、reconfigure、recover、recovery、references、reject、remove、_rename_phystable_name、_reorder_columns、repfunc、replay、rep_if_changed、replicate、replicate_if_changed、replication、request、_resetq、_resetqueue、resetqueue、resume、resync、retry、revoke、_rc、_rf、_rl、_roc、rollback、route、row、rpc、_rpn、_rs_alterrepdef、rs_rcl、rs_ticket、_rsc、rsrpc**
S	**scan、schedule、searchable、segment、select、send、sendallxacts、seq、server、set、shutdown、site、size、skip、source、sql、sqlddl、sqldml、_st、standby、starting、status、stdb、string、subscribe、subscription、suspend、suspension、switch、sys_sp、sysadmin、system**
T	**table、tables、template、textcol、textlen、_tl、_tn、to、_tp、tpinit、tpnull、_tr、trace、tran、transaction、transactions、transfer、truncate、truncation、twosave**
U	**_up、unsigned、update、use、user、username、using**

	保留字
V	**validate、verify、verify_repserver_cmd、vers**
W	**wait、warmstdb、_wh、where、with、without、withouttp、_wo、writetext**
Y	**_yd、yielding**
Z	**_zl、zerolen**

对 Adaptive Server 的支持

概述 Replication Server 对 Adaptive Server 的特定支持。

Replication Server 向国际客户提供以下支持：

- 支持 Sybase 所支持的所有字符集，包括 8 位、多字节字符集和 Unicode 字符集
- 支持 Sybase 所支持的所有排序顺序，包括非二进制排序顺序和 Unicode 排序顺序
- 已将 Replication Server 中的消息本地化为英语、法语、德语和日语
- 对 Replication Server 逻辑页大小、列数量和列大小以及存储过程参数数量的支持

下面的信息对这些功能进行了说明。有关在国际环境下设计复制系统的指导，请参见《Replication Server 设计指南》中的“国际复制设计注意事项”。

字符集支持

Replication Server 支持 Sybase 所支持的所有字符集，并且可以根据需要执行数据和标识符的字符集转换。

适用于字符集转换的准则：

- Replication Server 与所有 Sybase 软件一样，不能在单字节字符数据和多字节字符数据之间进行转换。
- 如果标识符（例如，表名和列名）包含多字节字符或带高位字符的单字节字符，必须用双引号将其括起来。
- XML 文本数据必须用单字节字符集进行编码，或者必须用与 Adaptive Server 相同的字符集进行编码。

指定字符集

您可以在 Replication Server 配置文件中使用 *rs_charset* 参数指定字符集。还可以指定编写 Replication Server 配置文件的字符集。该参数为 *CONFIG_charset*。

要使复制能够正常进行，Replication Server 的字符集必须与它所控制的数据服务器的字符集相同。同时，该字符集应当与系统中所有其它 Replication Server 的字符集兼容。

字符集转换

Replication Server 在主数据库和复制数据库之间执行数据和标识符的字符集转换。但是，Replication Server 不在不兼容的字符集之间执行字符集转换。如果两个字符集兼

容，但其中一个字符集中有一个或多个字符不属于另外一个字符集，那么将使用问号 (?) 替换未识别的字符。

使用 *rs_config* 系统表中的配置参数 **dsi_charset_convert**，可以选择 Replication Server 处理字符集转换的方法。此参数用 **alter connection** 命令来设置。

rs_get_charset 系统函数
Replication Server 每次连接到数据服务器时，都会执行 **rs_get_charset**，以获取该数据服务器所使用的字符集。如果该字符集不是预期的，Replication Server 将向错误日志文件中写入一条警告消息。

另请参见

- rs_get_charset （第 410 页）
- alter connection （第 109 页）

排序顺序支持

Replication Server 使用排序顺序或归类序列来决定如何比较字符数据和标识符并进行排序。Replication Server 支持 Sybase 所支持的所有排序顺序，包括非二进制排序顺序。对于欧洲语言，非二进制排序顺序对字符数据和标识符的正确排序非常必要。

可以在 Replication Server 配置文件中使用 *RS_sortorder* 参数指定排序顺序。您可以指定 Sybase 支持的且与您的字符集兼容的任何排序顺序。

要使复制正常进行，复制系统中的所有排序顺序应相同。

rs_get_sortorder 系统函数
Replication Server 每次连接到数据服务器时，都会执行 **rs_get_sortorder**，以获取该数据服务器所使用的排序顺序。如果该字符集不是预期的，Replication Server 将向错误日志文件中写入一条警告消息。

另请参见

- rs_get_sortorder （第 413 页）

消息语言支持

Replication Server 可以使用法语、德语和日语在错误日志和客户端中写入消息。可以在 Replication Server 配置文件中使用 *RS_language* 参数指定语言。

您可以指定 Replication Server 已经进行本地化的、与您的字符集兼容的任何语言。英语是缺省语言，与所有的 Sybase 字符集兼容。

存储过程消息
rs_msgs 系统表存储在安装过程中使用的以及管理 RSSD 的 Replication Server 存储过程使用的本地化错误消息。有关 *rs_msgs* 系统表的详细信息，请参见 *rs_queuemsg*。

扩展页大小和列大小支持

了解对扩展限制的支持。

Replication Server 12.5 版和更高版本支持 Adaptive Server 12.5 版和更高版本所支持的扩展限制。Replication Server 支持：

- 选择逻辑页大小：2K、4K、8K 或 16K
- 更大的行（最高为页大小的限制）
- 更宽的列（最高为页大小的限制）
- 更宽的索引键
- 每个表内更多的列
- 更大的消息（大于 16K 字节）

有关 Replication Server 中的扩展限制的详细信息，请参见《Replication Server 管理指南第一卷》。

混合版本复制系统

一个复制系统可以包括不同版本的 Replication Servers 或 Adaptive Servers。每个系统都存在不同的问题。

- 如果复制系统域包含 Replication Server 15.5 及更高版本，则复制系统域中的系统版本及所有节点和路由版本必须为版本 12.6 及更高版本。

  必须将 Replication Server 升级至版本 12.6 或更高版本，将节点版本设置为 12.6 或更高版本，并将路由升级至 12.6 或更高版本，然后才能升级到版本 15.5 或更高版本。

  请参见《Replication Server 配置指南》中的“升级或降级 Replication Server”。
- 当所有的 Replication Server 都至少是 12.6 版而且系统版本也已设为 12.6 版时，每一个 Replication Server 都可使用符合其 *节点版本*的功能。例如，运行 15.5 版的各个 Replication Server 能在彼此之间使用 15.5 版的所有功能，而运行 15.0 版的 Replication Server 就只能使用 15.0 版的功能。这样的系统被称为*混合版本系统*，每个 Replication Server 都可以使用其自身的所有功能。

混合版本系统中的限制

不同版本的 Replication Server 之间的交互作用会受到最旧版本的 Replication Server 的限制。

因此较早版本的 Replication Server 可能无法使用与新功能相关的信息。有关混合版本环境下的使用限制的其它信息，请参见说明新版本引进的每项功能（例如函数字符串继承或多个复制定义）的文档。

有关混合版本系统以及设置节点版本和系统版本的详细信息，请参见适用于您的平台的安装指南、配置指南以及发行公告。

Replication Server 命令

提供 RCL 命令的简要说明。

表 4. RCL 命令

命令	说明
abort switch （第 43 页）	中止 **switch active** 命令，除非 Replication Server 的活动切换进程已经进行到相当的程度而无法中止。
activate subscription （第 44 页）	对于预订复制定义或发布，启动从主数据库到复制数据库的更新分发，并将预订状态设置为“active”。
add partition （第 47 页）	使 Replication Server 可以使用某一分区。分区既可以是一个磁盘分区，也可以是一个操作系统文件。请参见 create partition。
admin config （第 47 页）	检索 Replication Server 参数，如 **global**、**connection**、**logical connection** 和 **route parameters**。
admin disk_space （第 50 页）	显示 Replication Server 所访问的每个磁盘分区的使用情况。
admin echo （第 51 页）	返回用户输入的字符串。
admin get_generation （第 52 页）	检索主数据库的生成号。
admin health （第 52 页）	显示 Replication Server 的状态。
admin log_name （第 54 页）	显示当前日志文件的路径。
admin logical_status （第 54 页）	显示逻辑连接的状态信息。
admin pid （第 56 页）	显示 Replication Server 的进程 ID。
admin quiesce_check （第 57 页）	确定 Replication Server 中的队列是否已停顿。
admin quiesce_force_rsi （第 58 页）	确定 Replication Server 是否已停顿，并强制 Replication Server 传递 RSI 队列中的消息并获取确认。
admin rssd_name （第 59 页）	显示 RSSD 的数据服务器和数据库的名称。
admin schedule （第 59 页）	显示有关调度的信息。
admin security_property （第 60 页）	显示有关所支持的基于网络的安全性机制和安全服务的信息。

命令	说明
admin security_setting （第 61 页）	显示 Replication Server 的基于网络的安全性参数和值。
admin set_log_name （第 62 页）	关闭现有的 Replication Server 日志文件并打开新的日志文件。
admin show_connection_profiles （第 63 页）	列出 Replication Server 中定义的每个配置文件的配置文件名称、版本和注释。
admin show_connections （第 66 页）	显示有关 Replication Server 与数据服务器和其它 Replication Server 的所有连接的信息。
admin show_function_classes （第 69 页）	显示现有的函数字符串类及其父类的名称，并指出继承的层数。
admin show_route_versions （第 70 页）	显示起始/终止于 Replication Server 的路由的版本号。
admin show_site_version （第 71 页）	显示 Replication Server 的节点版本。
admin sqm_readers （第 71 页）	显示正在读取稳定队列的线程的读取点和删除点。
admin stats （第 73 页）	显示有关 Replication Server 计数器的信息和统计数据。
admin stats, backlog （第 76 页）	报告当前积压在稳定队列中的事务。
admin stats, cancel （第 78 页）	取消当前运行的异步命令。
admin stats, {md | mem |mem_in_use} （第 78 页）	报告与内存使用情况有关的信息。
admin stats, reset （第 79 页）	重置所有可重置的计数器。
admin stats, status （第 79 页）	显示所有计数器的刷新状态。
admin stats, {tps | cps | bps} （第 80 页）	报告吞吐量，即每秒的事务数、命令数或字节数。
admin time （第 82 页）	显示 Replication Server 的当前时间。
admin translate （第 82 页）	对某个值执行数据类型转换，以带有分隔符的文本格式显示结果。
admin verify_repserver_cmd （第 84 页）	检验 Replication Server 能否成功执行复制定义请求。
admin version （第 86 页）	显示 Replication Server 软件的版本号。

命令	说明
admin version, route （第 87 页）	报告从当前 Replication Server 升级到目标 Replication Server 或从源 Replication Server 升级到当前 Replication Server 的路由，并检查路由升级的状态。
admin version, "connection" （第 86 页）	升级 Replication Server 后用户数据库的升级状态列表。
admin who （第 88 页）	显示有关 Replication Server 中正在运行的线程的信息。
admin who_is_down （第 104 页）	显示有关未运行的 Replication Server 线程的信息。
admin who_is_up （第 105 页）	显示有关正在运行的 Replication Server 线程的信息。
allow connections （第 106 页）	为指定数据库将 Replication Server 置于恢复模式。
alter applied function replication definition （第 107 页）	更改现有的应用函数复制定义。
alter connection （第 109 页）	更改数据库连接的属性。
alter connector （第 134 页）	更改数据库连接器的属性。
alter database replication definition （第 135 页）	更改现有的数据库复制定义。
alter encryption key （第 137 页）	重新生成加密密钥。
alter error class （第 138 页）	通过从其它错误类中复制错误操作来更改现有错误类。
alter function （第 139 页）	将参数添加到用户定义的函数中。
alter function replication definition （第 141 页）	更改现有的函数复制定义。
alter function string （第 143 页）	替换现有的函数字符串。
alter function string class （第 145 页）	更改函数字符串类，指定此类应是基类还是派生类。
alter logical connection （第 146 页）	启用或禁用逻辑连接的分配器线程，更改逻辑连接的属性，以及允许或禁止将 **truncate table** 复制到备用数据库。
alter partition （第 149 页）	更改分区的大小。
alter queue （第 150 页）	指定遇到超过 16K 个字节的大消息时稳定队列的行为。
alter replication definition （第 152 页）	更改现有的复制定义。

命令	说明
alter request function replication definition （第 159 页）	更改现有的请求函数复制定义。
alter route （第 161 页）	更改从当前 Replication Server 到远程 Replication Server 的路由的属性。
alter schedule （第 168 页）	启用或禁用执行命令的日程表。
alter subscription （第 169 页）	在使用同一 Replication Server 的同一复制数据库的复制连接之间移动预订而无需重新实现。
alter user （第 170 页）	更改用户口令。
assign action （第 172 页）	将 Replication Server 错误处理操作指派给 DSI 线程接收的数据服务器错误。
check publication （第 176 页）	查找发布状态以及发布所包含的项目数。
check subscription （第 177 页）	查找复制定义或发布的预订的实现状态。
configure connection （第 180 页）	更改数据库连接的属性。
configure logical connection （第 180 页）	更改逻辑连接的属性。
configure replication server （第 181 页）	设置 Replication Server 的特性，包括基于网络的安全性。
configure route （第 200 页）	更改从当前 Replication Server 到远程 Replication Server 的路由的属性。
connect （第 200 页）	将 Replication Server 变为其 RSSD、ID Server、远程 Replication Server 或远程数据服务器的网关。
create alternate connection （第 202 页）	添加替代主连接或复制连接，或替代活动连接或备用连接，并设置连接的配置参数。
create alternate logical connection （第 205 页）	向缺省逻辑连接添加替代逻辑连接。Replication Server 使用逻辑连接来管理热备份应用程序。
create applied function replication definition （第 206 页）	为要复制的存储过程创建应用函数复制定义和用户定义的函数。
create article （第 211 页）	创建表或函数复制定义的项目，并指定要包含此项目的发布。
create connection （第 214 页）	将数据库添加到复制系统，并设置此连接的配置参数。要创建 Adaptive Server 数据库的连接，请使用 Sybase Central™ 或 **rs_init**。

命令	说明
create connection using profile （第 219 页）	使用预定义的信息配置 Replication Server 和非 Adaptive Server 数据库之间的连接；如果需要，还会修改 RSSD 和命名的 *data_server.database*。
create database replication definition （第 225 页）	创建用于复制数据库或数据库对象的复制定义。
create error class （第 228 页）	创建错误类。
create function （第 231 页）	创建用户定义的函数。
create function replication definition （第 232 页）	为要复制的存储过程创建函数复制定义和用户定义的函数。
create function string （第 236 页）	将函数字符串添加到函数字符串类中。Replication Server 使用函数字符串来生成数据服务器的指令。
create function string class （第 249 页）	创建函数字符串类。
create logical connection （第 252 页）	创建逻辑连接。Replication Server 使用逻辑连接来管理热备份应用程序。
create partition （第 253 页）	使 Replication Server 可以使用某一分区。分区既可以是一个磁盘分区，也可以是一个操作系统文件。
create publication （第 254 页）	为表或存储过程创建发布，以将其作为一组复制到一个或多个预订复制数据库。
create replication definition （第 258 页）	为要复制的表创建复制定义。
create request function replication definition （第 269 页）	为要复制的存储过程创建请求函数复制定义和用户定义的函数。
create route （第 273 页）	指定用于当前 Replication Server 到远程 Replication Server 的连接的路由。
create schedule （第 277 页）	创建日程表以便在指定的时间执行 shell 命令。
create subscription （第 280 页）	创建和初始化预订，并实现预订数据。预订的对象可以是数据库复制定义、表复制定义、函数复制定义或发布。
create user （第 288 页）	将新的用户登录名添加到 Replication Server。
define subscription （第 290 页）	将预订添加到 Replication Server 系统表，但不实现或激活预订。预订的对象可以是数据库复制定义、表复制定义、函数复制定义或发布。此命令是批量预订实现进程或刷新发布预订进程的开始。
disconnect （第 295 页）	终止到服务器的连接。
drop article （第 296 页）	删除项目，也可以选择同时删除其复制定义。

命令	说明
drop connection （第 297 页）	从复制系统中删除数据库。
drop database replication definition （第 298 页）	删除现有的数据库复制定义。
drop error class （第 299 页）	删除错误类以及与其关联的所有操作。
drop function （第 300 页）	删除用户定义的函数及其函数字符串。
drop function replication definition （第 301 页）	删除函数复制定义及其用户定义的函数。
drop function string （第 302 页）	删除函数字符串类的函数字符串。
drop function string class （第 304 页）	删除函数字符串类。
drop logical connection （第 305 页）	删除逻辑连接。逻辑连接用于管理热备份应用程序。
drop partition （第 306 页）	从 Replication Server 删除磁盘分区。
drop publication （第 307 页）	删除发布及其所有项目，并且可以选择删除这些项目的复制定义。
drop replication definition （第 308 页）	删除复制定义及其函数。
drop route （第 309 页）	关闭到另一个 Replication Server 的路由。
drop schedule （第 311 页）	删除执行命令的日程表。
drop subscription （第 312 页）	删除对数据库复制定义、表复制定义、函数复制定义、项目或发布的预订。
drop user （第 315 页）	删除 Replication Server 用户登录名。
grant （第 316 页）	将权限指派给用户。
ignore loss （第 317 页）	允许 Replication Server 在检测到丢失后接受消息。
move primary （第 318 页）	更改错误类或函数字符串类的主 Replication Server。
rebuild queues （第 320 页）	重建 Replication Server 稳定队列。
resume connection （第 321 页）	恢复挂起的连接。
resume distributor （第 323 页）	为与数据库的连接恢复挂起的分配器线程。
resume log transfer （第 324 页）	使 RepAgent 能够连接到 Replication Server。

命令	说明
resume queue （第 325 页）	重新启动在传送超过 16K 个字节的消息后停止的稳定队列。
resume route （第 326 页）	恢复挂起的路由。
revoke （第 327 页）	撤消用户的权限。
set （第 328 页）	控制复制连接的复制定义属性。
set log recovery （第 330 页）	指定要从脱机转储恢复其日志的数据库。
set proxy （第 331 页）	切换到另一个用户。
show connection （第 332 页）	列出连接堆栈的内容。
show server （第 333 页）	显示当前的工作服务器。
shutdown （第 333 页）	关闭 Replication Server。
suspend connection （第 334 页）	挂起与某一数据库的连接。
suspend distributor （第 335 页）	挂起与主数据库连接的分配器线程。
suspend log transfer （第 336 页）	将 RepAgent 与 Replication Server 断开连接，并禁止 RepAgent 进行连接。
suspend route （第 337 页）	挂起到另一个 Replication Server 的路由。
switch active （第 338 页）	更改热备份应用程序中的活动数据库。
sysadmin apply_truncate_table （第 339 页）	为特定表的所有现有预订打开或关闭“subscribe to truncate table”选项，以启用或禁用 **truncate table** 的复制。
sysadmin cdb （第 340 页）	在向 Sybase IQ 的实时装载 (RTL) 复制和到 Adaptive Server 的高容量自适应复制 (HVAR) 中管理净更改数据库。
sysadmin dropdb （第 346 页）	从 ID Server 中删除数据库。
sysadmin dropldb （第 347 页）	从 ID Server 中删除逻辑数据库。
sysadmin drop_queue （第 348 页）	删除稳定队列。此命令用于删除失败的实现队列。
sysadmin droprs （第 349 页）	从 ID Server 中删除 Replication Server。
sysadmin dump_file （第 349 页）	指定转储 Replication Server 稳定队列时使用的备用日志文件名。
sysadmin dump_queue （第 350 页）	转储 Replication Server 稳定队列的内容。

命令	说明
sysadmin dump_thread_stacks（第 353 页）	转储 Replication Server 堆栈。
sysadmin dump_tran （第 355 页）	将稳定队列事务的命令转储到日志文件中。
sysadmin erssd （第 357 页）	显示 ERSSD 名称、日程表、备份目录和 ERSSD 文件位置。使用选项时，此命令执行非预定备份并移动 ERSSD 文件。
sysadmin fast_route_upgrade（第 359 页）	将路由版本更新为主 Replication Server 或复制 Replication Server 版本较低者的节点版本。
sysadmin hibernate_off （第 360 页）	关闭 Replication Server 的休眠模式并使其返回活动状态。
sysadmin hibernate_on （第 361 页）	打开 Replication Server 的休眠模式（即挂起 Replication Server）。
sysadmin issue_ticket （第 362 页）	将 **rs_ticket** 标记插入到入站队列中。
sysadmin lmconfig （第 364 页）	在 Replication Server 中配置和显示与许可证管理相关的信息。
sysadmin log_first_tran （第 366 页）	将 DSI 队列中的第一个事务写入例外日志。
sysadmin purge_all_open （第 367 页）	从 Replication Server 的入站队列中清除所有打开的事务。
sysadmin purge_first_open（第 368 页）	从 Replication Server 的入站队列清除第一个打开的事务。
sysadmin purge_route_at_replicate （第 369 页）	从复制 Replication Server 中删除对主 Replication Server 的所有引用。
sysadmin restore_dsi_saved_segments （第 370 页）	恢复积压的事务。
sysadmin set_dsi_generation（第 371 页）	更改 Replication Server 中的数据库生成号，以防止在复制数据库恢复后应用 DSI 稳定队列中的事务。
sysadmin site_version （第 372 页）	设置 Replication Server 的节点版本号。通过此命令，可以使用相应版本中的软件功能，防止降级到更低的版本。
sysadmin skip_bad_repserver_cmd （第 374 页）	指示 Replication Server 在 Replication Agent 下次启动时跳过失败的复制定义请求。
sysadmin sqm_purge_queue（第 375 页）	清除稳定队列中的所有消息。

命令	说明
sysadmin sqm_unzap_command （第 376 页）	将消息恢复到稳定队列中。
sysadmin sqm_unzap_tran （第 377 页）	将事务恢复到稳定队列中。
sysadmin sqm_zap_command （第 379 页）	删除稳定队列中的单个消息。
sysadmin sqm_zap_tran （第 380 页）	删除稳定队列中的事务。
sysadmin sqt_dump_queue （第 382 页）	转储入站队列或 DSI 队列的事务高速缓存。
sysadmin system_version （第 385 页）	显示或设置复制系统的系统范围版本号，以便使用相应版本级别中的软件功能。
sysadmin upgrade, route （第 388 页）	将路由从当前 Replication Server 升级到目标 Replication Server 并恢复任何失败的升级路由
sysadmin upgrade, "database" （第 387 页）	升级由 Replication Server 提供服务的用户数据库。
validate publication （第 389 页）	将发布的状态设置为“VALID”，以便为此发布创建新的预订。
validate subscription （第 390 页）	对于复制定义或发布的预订，将预订状态设置为“VALID”。此命令是批量实现进程的一部分，或刷新发布预订进程的一部分。
wait for create standby （第 392 页）	一个阻塞命令，使 Replication Server 中的客户端会话等待备用数据库创建进程完成。
wait for delay （第 392 页）	指定阻塞此命令的时间间隔。
wait for switch （第 393 页）	一个阻塞命令，使 Replication Server 中的客户端会话等待切换至新活动数据库的操作完成。
wait for time （第 394 页）	指定一天中取消阻塞此命令的时间。

abort switch

中止 **switch active** 命令，除非 Replication Server 的活动切换进程已经进行到相当的程度而无法中止。**switch active** 命令用于更改热备份应用程序中的活动数据库。

语法

```
abort switch for logical_ds.logical_db
```

参数

- **logical_ds** – 逻辑连接的数据服务器名称。
- **logical_db** – 逻辑连接的数据库名称。

示例

- **示例 1** – Replication Server 的活动切换进程已经进行到相当程度，无法取消。请等待切换完成，然后输入另一个 **switch active** 命令返回到原始活动数据库。

```
abort switch for LDS.pubs2
```

```
Switch for logical connection LDS.pubs2 is beyond the
point where it can be aborted. Abort command fails.
```

- **示例 2** – Replication Server 已中止活动切换。活动数据库尚未更改。

```
abort switch for LDS.pubs2
```

```
Switch for logical connection LDS.pubs2 has been aborted.
```

用法

- **abort switch** 命令尝试取消 **switch active** 命令。
- 如果逻辑连接没有正在进行的切换，Replication Server 会返回一条错误消息。
- 如果此命令成功取消了活动切换，您可能必须重新启动活动数据库的 RepAgent。
- 在执行到一定程度后，将无法取消 **switch active** 命令。在这种情况下，必须等待 **switch active** 完成。然后再次使用 **switch active** 返回到原始活动数据库。

权限

abort switch 需要“sa”权限。

另请参见

- switch active （第 338 页）
- admin logical_status （第 54 页）
- wait for switch （第 393 页）

activate subscription

对于预订复制定义或发布，启动从主数据库到复制数据库的更新分发，并将预订状态设置为“ACTIVE”。**activate subscription** 命令是批量实现进程的一部分，或刷新发布预订进程的一部分。

语法

```
activate subscription sub_name
for {table_rep_def | function_rep_def |
```

```
publication pub_name
with primary at data_server.database}
with replicate at data_server.database
[with suspension [at active replicate only]]
```

参数

- **sub_name** – 要激活的预订的名称。
- **for table_rep_def** – 指定所预订的表复制定义的名称。
- **for function_rep_def** – 指定所预订的函数复制定义的名称。
- **for publication pub_name** – 指定所预订的发布的名称。
- **with primary at data_server.database** – 指定主数据的位置。如果主数据库是热备份应用程序的一部分，则 *data_server.database* 是逻辑数据服务器和数据库的名称。只有对发布的预订才使用此子句。
- **with replicate at data_server.database** – 指定复制数据的位置。如果复制数据库为使用逻辑连接的热备份应用程序的一部分，则 *data_server.database* 是逻辑数据服务器和数据库的名称。
- **with suspension** – 更改预订状态之后，挂起复制数据库的数据服务器接口 (DSI)。挂起 DSI 时，Replication Server 将对复制数据库的更新保存在稳定队列中。装载初始数据并恢复 DSI 后，Replication Server 将应用这些更新。在热备份应用程序中，此子句挂起活动数据库 DSI 和备用 DSI。
- **with suspension at active replicate only** – 在热备份应用程序中，挂起活动数据库 DSI，但不挂起备用 DSI。

示例

- **示例 1** – 激活对表复制定义 *titles_rep* 的预订 *titles_sub*，其中复制数据库是 SYDNEY_DS.*pubs2*。此命令挂起 DSI。

```
activate subscription titles_sub
 for titles_rep
 with replicate at SYDNEY_DS.pubs2
 with suspension
```

- **示例 2** – 激活对函数复制定义 *myproc_rep* 的预订 *myproc_sub*，其中复制数据库是 SYDNEY_DS.*pubs2*。

```
activate subscription myproc_sub
 for myproc_rep
 with replicate at SYDNEY_DS.pubs2
```

- **示例 3** – 激活对发布 *pubs2_pub* 的预订 *pubs2_sub*，其中主数据库是 TOKYO_DS.*pubs2*，复制数据库是 SYDNEY_DS.*pubs2*。

```
activate subscription pubs2_sub
 for publication pubs2_pub
 with primary at TOKYO_DS.pubs2
 with replicate at SYDNEY_DS.pubs2
```

用法

- 使用 **activate subscription** 激活主 Replication Server 和复制 Replication Server 上的预订。预订的对象可能是表复制定义、函数复制定义、数据库复制定义或发布。
- 此命令是批量实现进程的第二步的开始。第一步是使用 **define subscription** 创建预订。
- 要完成批量实现，需要从介质装载数据，恢复与复制数据库的连接（如果此连接已被挂起），然后执行 **validate subscription**。
- 在创建预订的 Replication Server 上执行 **activate subscription**。
- **activate subscription** 将预订的状态从“DEFINED”更改为“ACTIVE”。主数据服务器上的后续更新通过主 Replication Server 分发。
- 如果通过现有预订已经向发布添加了任何新项目，必须通过实现新数据刷新发布预订，才能为新项目创建预订。

  使用 **define subscription** 开始此进程后，请使用 **activate subscription** 激活新项目预订。然后为新项目预订手动装载预订数据，并使用 **validate subscription** 来验证发布预订。
- 激活发布预订时，将会同时激活其所有的项目预订，而不是一次激活一个。
- 此命令修改多个节点上的 RSSD 表。在主 Replication Server 和复制 Replication Server 上使用 **check subscription** 可查看对每个预订的影响。
- 有关预订实现的详细信息，请参见《Replication Server 管理指南第一卷》。

with suspension 子句

- 使用 **with suspension** 子句时，**activate subscription** 将在更改预订状态后挂起 DSI。这样可以防止复制 Replication Server 在装载预订数据前发送被复制的表的更新。在复制节点装载数据后，请执行 **resume connection** 以应用更新。如果不使用 **with suspension**，则应当在实现预订之前禁止对主版本进行更新。
- 如果数据库是热备份应用程序的一部分，**with suspension** 子句将在更改预订状态后挂起活动数据库的 DSI 和备用 DSI。这样使您可以将数据装载到这两个数据库中，然后继续更新活动数据库。

  如果通过记录将数据装载到活动数据库中（例如，通过使用记录的 **bcp** 或在活动数据库中执行事务），请使用 **with suspension at active replicate only** 子句，这样就不会挂起备用 DSI。在这种情况下，您不必将预订数据装载到备用数据库中，因为可从活动数据库复制预订数据。

权限

activate subscription 在复制 Replication Server 上可以由具有“创建对象”权限的用户执行，在主 Replication Server 上则可以由具有“主预订”权限的用户执行。

另请参见

- check subscription （第 177 页）
- create subscription （第 280 页）
- define subscription （第 290 页）

- drop subscription （第 312 页）
- resume connection （第 321 页）
- validate subscription （第 390 页）

add partition

使 Replication Server 可以使用某一分区。分区既可以是一个磁盘分区，也可以是一个操作系统文件。

注意： **add partition** 和 **create partition** 相同，只是命令名不同。为了保持向后兼容，仍然支持 **add partition** 并将其作为 **create partition** 的另一表示形式，但以后不再支持。

语法

有关语法信息，请参见 **create partition**。

用法

有关用法信息，请参见 **create partition**。

另请参见

- create partition （第 253 页）

admin config

显示所有 Replication Server 配置参数。

语法

```
admin config [,[[[{ "connection" | logical_connection}
   , data_server, database] | [ "route", repserver]]
   [, configuration_name] | [ "table", data_server, database,
   [, table_name [[, table_owner], [, configuration_name]]]]
```

注意： 如果配置值长于 255 个字节，则 **admin config** 命令只显示前 251 个字节和一个省略号 (...).

参数

- **"connection"** – 显示连接配置参数。
- **logical_connection** – 显示逻辑连接配置参数。
- **"table"** – 指定要查询的表的名称。与 *table_name*（最多 200 个字符的字符串）一起使用。*table_owner* 是表名的可选限定符，表示表的所有者。

如果不指定表名，**admin config** 将显示所有表的配置参数。

- **data_server, database** – 所查询的数据服务器和数据库。

  如果要显示的配置参数与连接有关，则 server 必须是数据服务器，并且必须提供 *database*。如果要显示的参数与路由有关，则 server 必须是 Replication Server，并且不能提供 *database*。
- **”route”** – 显示路由配置参数。
- **repserver** – 指定路由的目标 Replication Server。
- **configuration_name** – 要显示其值和状态的配置参数。

示例

- **示例 1** – 显示所有 Replication Server 全局配置参数：

```
admin config
go
```

```
Configuration          Config   Run      Default
                       Value    Value    Value
--------------------   ------   -----    -------
cm_max_connections     65       65       64
dsi_cmd_batch_size     8193     8193     8192

Legal Values          Datatype   Status
--------------------  --------   ------------------
range: 1,2147483647   integer    Restart required
range: 1,2147483647   integer    Restart required
(2 rows affected)
```

- **示例 2** – 显示到 Replication Server TOKYO_RS 的路由的所有配置参数：

```
admin config, "route", TOKYO_RS
```

- **示例 3** – 显示到 pdb1 的连接的所有配置参数：

```
admin config,"connection",ost_wasatch_04,pdb1
go
```

```
Configuration          Config   Run      Default
                       Value    Value    Value
--------------------   ------   -----    -------
dsi_cmd_batch_size     NULL     NULL     8192

Legal Values          Datatype   Status
--------------------  --------   ------------------
range: 1,2147483647   integer    Connection/Route
                                 restart required
(1 row affected)
```

- **示例 4** –

  在使用 **dsi_command_convert** 在 SYDNEY_DS 数据服务器的 pubs2 数据库中的 *tb1* 表上设置 **d2none** 后显示所有配置参数：

```
admin config, “table”, SYDNEY_DS, pubs2
```

admin config 显示：

```
Configuration         Config             Run                Default
                       Value              Value               Value
-------------------   ------             -----              -------
dsi_compile_enable    <server default>    <server default>    on
dsi_command_convert   d2none             d2none              none

Legal Values                                                Datatype
-----------------------------------------------------------  -------
-
list: on,of                                                  string
list: none, i2none, d2none, u2none, i2di, u2di, t2none       string

Status                 Table
--------------------   -----
Restart not required   dbo.tb1
Restart not required   dbo.tb1

(2 rows affected)
```

- **示例 5** – 在 SYDNEY_DS 数据服务器的 pubs2 数据库中的 *tb1* 表上使用 **dsi_command_convert** 后只显示 **dsi_command_convert** 的配置参数：

```
admin config, "table", SYDNEY_DS, pubs2, tb1,
dsi_command_convert
```

admin config 显示：

```
Configuration         Config   Run      Default
                      Value    Value    Value
-------------------   ------   -----    -------
dsi_command_convert   d2none   d2none   none

Legal Values                                                Datatype
-----------------------------------------------------------  -------
-
list: none, i2none, d2none, u2none, i2di, u2di, t2none       string

Status                 Table
--------------------   -----
Restart not required   dbo.tb1

(1 row affected)
```

用法

使用 **admin config** 检索用于自定义和调优 Replication Server 的各种配置参数（服务器、连接、逻辑连接和路由等）。

有关配置和调优 Replication Server 参数的详细信息，请参见《Replication Server 管理指南第一卷和第二卷》。

admin disk_space

显示 Replication Server 所访问的每个磁盘分区的使用情况。

语法

```
admin disk_space
```

示例

- **示例 1** – 显示磁盘分区的相关信息：

```
admin disk_space
```

```
Partition     Logical           Part.Id
----------    -----------       -------
/dev/hdb2     partition_1       101

Total Segs    Used Segs         State
-----------   ---------         -------
         20            3         ON-LINE
```

用法

表 5. admin disk_space 的输出列说明

列	说明
Partition 区	Replication Server 使用的设备名
Logical	指派给分区的逻辑名
Part.Id	分区 ID
Total Segs	一个分区内 1MB 段的总数
Used Segs	Replication Server 当前使用的总段数
State	此设备的状态。可以是： • ON-LINE – 此设备正常 • OFF-LINE – 无法找到此设备 • DROPPED – 此设备已被删除，但尚未消失（有些队列正在使用它）

权限

任何用户都可以执行此命令。

另请参见

- admin who （第 88 页）
- alter partition （第 149 页）
- create partition （第 253 页）
- drop partition （第 306 页）

admin echo

返回用户输入的字符串。

语法

```
admin echo, character_string [, with_log]
```

参数

- **character_string** – 用户输入的字符串。
- **with_log** – 将用户输入的字符串写入到 Replication Server 日志中。

示例

- **示例 1** – Replication Server 返回用户输入的字符串“hello”。

  ```
  admin echo, hello
  ```

  ```
  echo
  -----
  hello
  ```

- **示例 2** – Replication Server 返回“Hello world!”，并将 “Hello world!”写入 Replication Server 日志。

  ```
  admin echo, 'Hello world!', with_log
  ```

  ```
  echo
  ------------
  Hello world!
  ```

用法

- 使用 **admin echo** 确定本地 Replication Server 是否正在运行。
- 此命令的作用不是网络回应。如果不输入参数，将不返回任何内容。

权限

任何用户都可以执行此命令。

admin get_generation

检索主数据库的生成号。

语法

```
admin get_generation, data_server, database
```

参数

- **data_server** – 包含主数据库的数据服务器。
- **database** – 要检索其生成号的数据库。

示例

- **示例 1** –

```
admin get_generation, TOKYO_DS, pubs2
```

```
Current generation number for TOKYO_DS.pubs2 is 0
```

用法

- 数据库生成号是 RepAgent 为日志记录生成的原始队列 ID 的前 2 个字节。生成号是日志传送语言 (LTL) **distribute** 命令的一个参数。
- 每次为主数据库装载数据后，生成号都会递增。递增生成号可防止 Replication Server 忽略在装载后应用的任何事务（将其视为重复事务）。
- 通过在 Adaptive Server 数据库中执行 Adaptive Server **dbcc settrunc**，可以递增生成号。

权限

任何用户都可以执行此命令。

另请参见

- dbcc settrunc （第 447 页）

admin health

显示 Replication Server 的状态。

语法

```
admin health
```

示例

- **示例 1** – 显示 Replication Server 的状态。

```
admin health
```

```
Mode         Quiesce    Status
-------      -------    -------
NORMAL       TRUE       HEALTHY
```

用法

表 6. admin health 的输出列说明

列	说明
Mode	与恢复有关的 Replication Server 状态。状态为以下值之一: • NORMAL – Replication Server 正常运行。 • REBUILDING – 这是 Replication Server 执行 **rebuild queues** 命令时的瞬时状态。 • RECOVERY – Replication Server 处于独立模式，并且已经执行 **rebuild queues** 命令。 • STANDALONE – Replication Server 当前没有接受或启动任何连接。只能使用 **-M** 标志启动 Replication Server 以进入此状态。通过关闭 Replication Server，然后不使用 **-M** 标志重新启动它，即可退出独立模式。
Quiesce	指示 Replication Server 是否已停顿。值可以是: • TRUE – Replication Server 已停顿，即，已刷新所有消息。 • FALSE – Replication Server 未停顿。
Status	Replication Server 的总体状态。值可以是: HEALTHY – 所有线程均正常执行。 SUSPECT – 某个线程停止，Replication Server 原以为它正在运行。或者线程处于“正在连接”状态。“正在连接”状态表示 Replication Server 所连接的服务器不可用、存在问题，或 Replication Server 即将成功连接，suspect 状态是暂时的。 执行 **admin who_is_down** 可以查看未运行的线程。

权限

任何用户都可以执行此命令。

另请参见

- admin quiesce_check （第 57 页）
- admin quiesce_force_rsi （第 58 页）
- admin who （第 88 页）
- admin who_is_down （第 104 页）

- admin who_is_up （第 105 页）
- rebuild queues （第 320 页）

admin log_name

显示当前日志文件的路径。

语法

```
admin log_name
```

示例

- **示例 1** – 显示当前 Replication Server 的日志文件的路径。

```
admin log_name
```

```
Log File Name
-----------------------------------
/work/log/TOKYO_RS.log
```

用法

如果带 **-e** 标志启动 Replication Server，并给出错误日志的完整路径，**admin log_name** 将返回完整路径。如果给出相对路径名，**admin log_name** 将返回 Replication Server 当前工作目录中的相对路径名。

权限

任何用户都可以执行此命令。

另请参见

- admin set_log_name （第 62 页）

admin logical_status

显示逻辑连接的状态信息。

语法

```
admin logical_status [, logical_ds, logical_db]
```

参数

- **logical_ds** – 逻辑连接的数据服务器名称。

- **logical_db** – 逻辑连接的数据库名称。

示例

- **示例 1** – 此输出显示 LDS.*pubs2* 逻辑连接处于正常的活动状态。当前活动数据库是 TOKYO_DS 数据服务器中的 *pubs2* 数据库。备用数据库是 SYDNEY_DS 数据服务器中的 *pubs2* 数据库。TOKYO_RS Replication Server 管理逻辑连接。两个物理连接都是活动的。当前未进行特别的操作。

```
admin logical_status, LDS, pubs2
```

Logical Connection Name	Active Connection Name	Active Conn State	Standby Connection Name	Standby Conn State
[109] LDS.pubs2	[115] TOKYO_DS.pubs2	Active/	[116] SYDNEY_DS. pubs2	Active/

Controller RS	Operation in Progress	State of Operation in Progress	Spid
[16777317] TOKYO_RS	None	None	

用法

- 使用 **admin logical_status** 了解热备份应用程序中活动数据库和备用数据库的逻辑连接的状态。
- 如果不指定 *logical_ds* 和 *logical_db*，**admin logical_status** 将显示此 Replication Server 控制的所有逻辑连接的相关信息。
- admin logical_status 输出表的列说明对各个输出列进行说明。

表 7. admin logical_status 的输出列说明

列	说明
Logical Connection Name	逻辑连接和逻辑数据服务器的 DBID（数据库 ID）和数据库名。
Active Connection Name	DBID、数据服务器和当前活动数据库的数据库名。
Active Connection State	活动连接的状态说明。可以是 active、suspended 或 suspended by error。
Standby Connection Name	DBID、数据服务器和当前备用数据库的数据库名称。
Standby Connection State	备用连接的状态说明。可以是 active、suspended、suspended by error 或 waiting for marker。

列	说明
Controller RS	管理逻辑数据库、活动数据库和备用数据库的 Replication Server 的 RSID (Replication Server ID) 和名称。
Operation in Progress	正在进行的操作的说明。可以是 None、Switch Active 或 Create Standby。
State of Operation in Progress	操作的当前步骤。
Spid	正在执行操作的服务器线程的进程 ID。

权限

任何用户都可以执行此命令。

另请参见

- abort switch （第 43 页）
- admin sqm_readers （第 71 页）
- admin who （第 88 页）
- create connection （第 214 页）
- create logical connection （第 252 页）
- switch active （第 338 页）
- wait for create standby （第 392 页）
- wait for switch （第 393 页）

admin pid

显示 Replication Server 的进程 ID。

语法

```
admin pid
```

示例

- **示例 1** – 当前 Replication Server 的进程 ID 是 12032。

```
admin pid
```

```
pid
--------------
12032
```

用法

显示 Replication Server 的进程 ID。

权限

任何用户都可以执行此命令。

admin quiesce_check

确定 Replication Server 中的队列是否已停顿。

语法

```
admin quiesce_check
```

示例

- **示例 1** – TOKYO_RS Replication Server 已停顿。

  ```
  admin quiesce_check
  ```

  ```
  Replication Server TOKYO_RS is quiesced
  ```

- **示例 2** – 此消息指示系统未处于停顿状态，因为队列 103:1 中存在未读消息。报告的读位置 (30.2) 和写位置 (32.1) 表明，队列中写入的块数多于读取的块数。假设系统停顿之前没有写入其它块，读位置必须前进到段 32、块 2。

  ```
  admin quiesce_check
  ```

  ```
  Can't Quiesce. Queue 103:1 has not been read out.
   Write=32.1 Read=30.2
  ```

用法

- **admin quiesce_check** 确定 Replication Server 是否停顿。
- 如果符合以下条件，Replication Server 将会停顿：
  - 没有预订实现队列。
  - Replication Server 已读取并处理所有队列中的所有消息。
  - 没有任何入站 (RepAgent) 队列包含未传递的已提交事务。
  - RSI 队列中的所有消息已发送到其目标 Replication Server，并已收到确认。
  - DSI 队列中的所有消息已应用，并且已从数据服务器收到确认。

权限

任何用户都可以执行此命令。

另请参见

- admin quiesce_force_rsi （第 58 页）
- suspend connection （第 334 页）
- suspend log transfer （第 336 页）

admin quiesce_force_rsi

确定 Replication Server 是否已停顿，并强制 Replication Server 传递 RSI 队列中的消息并获取确认。

语法

```
admin quiesce_force_rsi
```

示例

- **示例 1** – TOKYO_RS Replication Server 已停顿。

```
admin quiesce_force_rsi
```

```
Replication Server TOKYO_RS is quiesced
```

- **示例 2** – 此消息指示系统未处于停顿状态，因为队列 103:1 中存在未读消息。报告的写位置 (32.1) 和读位置 (30.2) 表明，队列中写入的块数多于读取的块数。

```
admin quiesce_force_rsi
```

```
Can't Quiesce. Queue 103:1 has not been read out.
Write=32.1 Read=30.2
```

用法

- 在执行 **admin quiesce_force_rsi** 之前，执行 **suspend log transfer from all**。这可以防止 RepAgent 与 Replication Server 进行连接。
- 在所有入站队列都停顿后执行此命令。
- 如果符合以下条件，Replication Server 将会停顿：
  - 没有预订实现队列
  - Replication Server 已读取所有队列中的所有消息
  - 没有任何入站 (RepAgent) 队列包含未传递的已提交事务
  - RSI 队列中的所有消息已发送到其目标 Replication Server，并已收到确认
  - DSI 队列中的所有消息已应用，并且已从数据服务器收到确认
- RSI 通常每 30 秒将队列清空一次。

权限

任何用户都可以执行此命令。

另请参见

- admin quiesce_check （第 57 页）
- suspend connection （第 334 页）
- suspend log transfer （第 336 页）

admin rssd_name

显示 RSSD 的数据服务器和数据库的名称。

语法

```
admin rssd_name
```

示例

- **示例 1** – 在此示例中，TOKYO_DS 是数据服务器的名称，TOKYO_RSSD 是 RSSD 的名称。

```
admin rssd_name
```

```
RSSD Dataserver    RSSD Database
---------------    ------------
TOKYO_DS           TOKYO_RSSD
```

用法

显示 RSSD 的数据服务器和数据库的名称。

权限

任何用户都可以执行此命令。

admin schedule

显示有关 Replication Server 中的任务日程表的信息。

语法

```
admin “schedule”[, ‘sched_name’]
```

参数

- **‘sched_name’** – 要显示的日程表的名称。

示例

- **示例 1** – 要显示名为 **schedule1** 的日程表，请输入：

```
admin “schedule”, ‘schedule1’
```

输出为：

```
Schedule  Schedule    Status  Type  Owner  Sequence   Command
Name      Time
--------  ----------  ------  ----  -----  --------   -------
s1        27 * * * *    1       0     sa       1      conn_suspend.sh
```

用法

必须用双引号括起 **“schedule”** 子句，因为 **schedule** 是 Replication Server 的一个关键字。

如果不指定任何日程表名，只执行 **admin “schedule”** 会显示有关 Replication Server 中所有现有日程表的信息。

权限

admin “schedule” 需要“sa”权限。

另请参见

- alter schedule （第 168 页）
- drop schedule （第 311 页）
- create schedule （第 277 页）

admin security_property

显示有关所支持的基于网络的安全性机制和安全服务的信息。

语法

```
admin security_property [, mechanism_name]
```

参数

- **mechanism_name** – 支持的基于网络的安全性机制。

示例

- **示例 1** –

```
admin security_property
```

```
Mechanism        Feature             Supported
---------------  ----------------    ------------
```

```
DCE              Unified Login      yes
DCE              Confidentiality    yes
DCE              Integrity          no
...
```

用法

- 不带选项执行此命令时，显示缺省安全性机制的名称、此机制可用的安全性服务以及所在节点是否支持可用的服务。
- 要执行 **admin security_property**，必须在当前 Replication Server 上启用基于网络的安全性，方法是使用 **configure replication server** 将 **use_security_services** 参数设置为 on。

权限

任何用户都可以执行此命令。

另请参见

- admin security_setting （第 61 页）
- alter connection （第 109 页）
- alter route （第 161 页）
- configure replication server （第 181 页）
- create connection （第 214 页）
- create route （第 273 页）
- set proxy （第 331 页）

admin security_setting

显示 Replication Server 的基于网络的安全性参数和值。

语法

```
admin security_setting [, rs_idserver |, rs_server |,
data_server.database]
```

参数

- **rs_idserver** – 当前 Replication Server 所连接的 ID Server。
- **rs_server** – 当前 Replication Server 所连接的 Replication Server。
- **data_server** – 当前 Replication Server 所连接的目标数据库所在的数据服务器。
- **database** – 当前 Replication Server 所连接的目标数据库。

示例

- **示例 1** –

```
admin security_setting
```

```
Server         Feature            Status
-------------  ---------------    -----------
Global         Unified Login      required
Global         Confidentiality    not_required
Global         Integrity          not_required
...
```

用法

- 要执行 **admin security_setting**，必须在当前 Replication Server 上启用基于网络的安全性，方法是使用 **configure replication server** 将 **use_security_services** 参数设置为“on”。
- 如果不带选项执行 **admin security_setting**，Replication Server 将显示用 **configure replication server** 配置的缺省值。

权限

任何用户都可以执行此命令。

另请参见

- admin security_property （第 60 页）
- alter connection （第 109 页）
- alter route （第 161 页）
- configure replication server （第 181 页）
- create connection （第 214 页）
- create route （第 273 页）
- set proxy （第 331 页）

admin set_log_name

关闭现有的 Replication Server 日志文件并打开新的日志文件。

语法

```
admin set_log_name, log_file
```

参数

- **log_file** – 新日志文件的名称。

示例

- **示例 1** – 打开名为 SYDNEY_RS.log 的新日志文件。可以使用 **admin log_name** 命令来检验路径和日志文件名。

```
admin set_log_name,
  '/work/log/SYDNEY_RS.log'
```

用法

- 如果此命令失败，则原来的日志文件仍保持打开状态。
- 如果重新启动了 Replication Server，将使用命令行中指定的日志文件名。如果命令行中未指定任何名称，则使用缺省日志文件名。
- 如果输入的日志文件名包含字母和数字以外的字符，要用引号将此文件名引起来。例如，如果日志文件名中包含句点 (.)，就应使用引号，如上例所示。
- **admin set_log_name** 显示您输入的名称。请输入绝对路径名，以使输出最便于使用。

权限

任何用户都可以执行此命令。

另请参见

- admin log_name （第 54 页）

admin show_connection_profiles

列出 Replication Server 中定义的每个配置文件的配置文件名称、版本和注释。

语法

```
admin show_connection_profiles[, "match_string"]
```

参数

- **match_string** – 过滤显示的连接配置文件。仅显示那些名称中包含提供的字符串的连接配置文件。

示例

- **示例 1** – 列出 Replication Server 当前定义的所有连接配置文件的名称：

```
admin show_connection_profiles
go
```

```
Profile Name                 Version  Comments
------------               -------- ------------------------------
rs_ase_to_db2                standard Standard ASE to DB2
```

```
replication
                                        connection profile.
rs_ase_to_udb               standard Standard ASE to UDB
replication
                                        connection profile.
rs_ase_to_oracle            standard Standard ASE to Oracle
replication
                                        connection profile.
rs_ase_to_msss              standard Standard ASE to Microsoft
SQLServer
                                     replication connection profile.
rs_ase_to_iq                standard Standard ASE to Sybase IQ
                                     replication connection profile.
rs_ase_to_ase               standard Standard ASE to ASE
replication
                                        connection profile.
rs_db2_to_msss              standard Standard DB2 to Microsoft
SQLServer
                                     replication connection profile.
rs_db2_to_oracle            standard Standard DB2 to Oracle
replication
                                        connection profile.
rs_db2_to_udb               standard Standard DB2 to UDB
replication
                                        connection profile.
rs_db2_to_ase               standard Standard DB2 to ASE
replication
                                        connection profile.
rs_oracle_to_db2            standard Standard Oracle to DB2
replication
                                        connection profile.
rs_oracle_to_udb            standard Standard Oracle to UDB
replication
                                         connection profile.
rs_oracle_to_msss          standard Standard Oracle to Microsoft
                                   SQLServer replication connection
                                        profile.
rs_oracle_to_ase            standard Standard Oracle to ASE
replication
                                        connection profile.
rs_msss_to_db2             standard Standard Microsoft SQLServer
to DB2
                                     replication connection profile.
rs_msss_to_oracle          standard Standard Microsoft SQLServer
to
                                      Oracle replication connection
profile.
rs_msss_to_udb             standard Standard Microsoft SQL Server
to
                                        UDB replication connection
profile.
rs_msss_to_sqlany          standard Standard Microsoft SQLServer
to SQL
                                     Anywhere replication connection
                                        profile.
rs_msss_to_ase             standard Standard MicrosoftSQL Server
```

```
to ASE
                                        replication connection profile.
rs_udb_to_db2                  standard Standard udb to db2
replication
                                             connection profile.
rs_udb_to_msss                 standard Standard UDB to Microsoft
SQLServer
                                        replication connection profile.
rs_udb_to_oracle               standard Standard UDB to Oracle
replication
                                             connection profile.
rs_udb_to_ase                  standard Standard UDB to ASE
replication
                                             connection profile.
rs_db2_to_db2                  standard Standard DB2 to DB2
replication
                                             connection profile.
rs_oracle_to_oracle            standard Standard Oracle to Oracle
                                        replication connection profile.
rs_udb_to_udb                  standard Standard UDB to UDB
replication
                                             connection profile.
rs_msss_to_msss               standard Standard Microsoft SQLServer
to
                                       Microsoft SQLServer replication
                                             connection profile.

(36 rows affected)
```

- **示例 2** – 列出连接配置文件名称中包含字符串“oracle”的 Replication Server 当前定义的所有连接配置文件的名称:

```
admin show_connection_profiles, “oracle”
go
```

```
Profile Name                    Version  Comments
----------------                -------- ----------------------------
---
rs_ase_to_oracle                standard Standard ASE to Oracle
replication
                                             connection profile.
rs_db2_to_oracle                standard Standard DB2 to Oracle
replication
                                             connection profile.
rs_oracle_to_db2                standard Standard Oracle to DB2
replication
                                             connection profile.
rs_oracle_to_udb                standard Standard Oracle to UDB
replication
                                             connection profile.
rs_oracle_to_msss              standard Standard Oracle to Microsoft
                                    SQLServer replication connection
                                             profile.
rs_oracle_to_ase                standard Standard Oracle to ASE
replication
                                              connection profile.
rs_msss_to_oracle              standard Standard Microsoft SQLServer
```

```
to
                                      Oracle replication connection
                                        profile.
rs_udb_to_oracle              standard Standard UDB to Oracle
replication
                                        connection profile.
rs_oracle_to_oracle           standard Standard Oracle to Oracle
                                   replication connection profile.
```

用法

在使用 **using profile** 选项创建连接时，请使用 **admin show_connection_profiles** 确定可用配置文件的名称和版本。

另请参见

- create connection using profile （第 219 页）

admin show_connections

显示有关 Replication Server 与数据服务器和其它 Replication Server 的所有连接的信息。

语法

```
admin show_connections[, 'primary' | 'replicate' | 'logical']
```

参数

- **primary** – 显示有关所有主连接的信息。
- **replicate** – 显示有关所有复制连接的信息。
- **logical** – 显示有关所有逻辑连接的信息。

示例

- **示例 1** – 显示此 Replication Server 的连接数据。

```
admin show_connections
```

```
       Server             User              Database
       ------             ----              --------
       SYDNEY_DS          pubs2_maint       pubs2sb
       SYDNEY_RS          SYDNEY_RS_rsi     NULL

       State                   Owner               Spid
       -----                   -----               ----
       already_faded_out        DSI                  89
       active                   RSI                  53

connection state   number  comments
----------------   -----   --------
```

```
 connecting         0      in the process of connecting to a server
 active             2      established connections owned and used by
                             threads
 idle               0       established connections owned but not
 being
                             used
 being_faded_out    0       idle connections that are being closed
 already_faded_out  0       idle connections that have been closed
 free               1      established connections not owned by any
                             threads
 closed             61      closed connections not owned by any t
hreads
 limbo              0      connection handles in state transition
 total              64      total number of connection handlers a
vailable
```

- **示例 2** – 显示到主数据库的所有连接。例如，在控制 SALES_DS 数据服务器中的主数据库的 Replication Server 中，输入：

  ```
  admin show_connections, 'primary'
  ```

  您会看到：

  ```
  Connection Name     Server         Database     User
  ----------------    ----------     ---------    ----------
  SALES_DS.pdb        SALES_DS       pdb          pdb_maint
  SALES_DS.pdb_conn2  SALES_DS       pdb          pdb_maint
  ```

  SALES_DS.pdb 是 Replication Server 和 SALES_DS 数据服务器的 pdb 数据库之间的缺省连接，因为连接名与数据服务器和数据库名的组合相匹配。

  SALES_DS.pdb_conn2 是 Replication Server 和 SALES_DS 数据服务器的 pdb 数据库之间的替代连接，因为连接名与数据服务器和数据库名的组合不匹配。

- **示例 3** – 显示到复制数据库的所有连接。例如，在控制 FINANCE_DS 和 NY_DS 数据服务器中的复制数据库的 Replication Server 中，输入：

  ```
  admin show_connections, 'replicate'
  ```

  您会看到：

  ```
  Connection Name         Server        Database   User
  -------------------    ----------   --------   ----------
  FINANCE_DS.fin_rdb     FINANCE_DS   fin_rdb    rdb_maint
  NY_DS.ny_rdb_conn2     NY_DS        ny_rdb     rdb_maint
  ```

  FINANCE_DS.fin_rdb 是 Replication Server 和 FINANCE_DS 数据服务器的 fin_rdb 数据库之间的缺省连接，因为连接名与数据服务器和数据库名的组合相匹配。

  NY_DS.ny_db_conn2 是 Replication Server 和 NY_DS 数据服务器的 ny_rdb 数据库之间的替代连接，因为连接名与数据服务器和数据库名的组合不匹配。

- **示例 4** – 显示到逻辑数据库的所有连接。

  ```
  admin show_connections, 'logical'
  ```

  您会看到：

```
Connection Name         Server       Database
-------------------     ----------   --------
WS_DS.ws_db             WS_DS        ws_db
WS_DS.ws_db1            WS_DS        ws_db
```

其中 WS_DS.ws_db 是缺省逻辑连接，而 WS_DS.ws_db1 是替代逻辑连接。

用法

- 此命令显示有关来自于当前 Replication Server 的缺省和替代数据库连接以及路由的信息。
- **admin show_connections** 输出表的列说明对此命令的输出进行说明。

表 8. admin show_connections 的输出列说明

列	说明
Connection Name	源自此 Replication Server 的缺省和替代主连接或复制连接的名称
Server	与此 Replication Server 连接的数据服务器或 Replication Server 的名称
User	此客户端的登录名
Database	此 Replication Server 连接的数据库的名称（对于路由，此值为 null）
State	此连接的状态
Owner	指示线程的所有者。值为以下一项： DSI – 数据服务器接口（对于数据库） RSI – Replication Server 接口（对于 Replication Server）
Spid	此线程的唯一标识符
connection state	值为以下一项： • *active* – 正在使用此连接 • *already_faded_out* – 此连接已经属于某所有者并已关闭 • *being_faded_out* – 此连接已经属于某所有者并正在关闭 • *closed* – 连接已关闭，不属于任何线程 • *connecting* – 正在与服务器连接 • *free* – 此连接已打开，不属于任何所有者 • *idle* – 此连接已经属于某所有者，但未使用 • *limbo* – 连接处理处于转换状态 • *total* – 连接的总数
number	此类型连接的数量
comments	*connection state* 字段的说明

权限

任何用户都可以执行此命令。

另请参见

- alter connection （第 109 页）
- alter logical connection （第 146 页）
- alter route （第 161 页）
- create connection （第 214 页）
- create logical connection （第 252 页）
- create route （第 273 页）
- drop connection （第 297 页）
- drop logical connection （第 305 页）
- drop route （第 309 页）
- resume connection （第 321 页）
- suspend connection （第 334 页）

admin show_function_classes

显示现有的函数字符串类及其父类的名称，并指出继承的层数。

语法

```
admin show_function_classes
```

示例

- **示例 1** –

```
admin show_function_classes
```

```
Class                          ParentClass                Level
--------                       ------------               -----
sql_derived_class              rs_default_function_class      1
DB2_derived_class              rs_db2_function_class          2
rs_db2_function_class          rs_default_function_class      1
rs_default_function_class      BASE_CLASS                     0
(and so on)
```

用法

级别 0 是基类，如 *rs_default_function_class*；级别 1 是从基类继承的派生类，依此类推。

权限

任何用户都可以执行此命令。

另请参见

- alter connection （第 109 页）
- alter function string class （第 145 页）
- create connection （第 214 页）
- create function （第 231 页）
- create function string （第 236 页）
- create function string class （第 249 页）
- drop function string class （第 304 页）
- move primary （第 318 页）

admin show_route_versions

显示起始/终止于 Replication Server 的路由的版本号。

语法

```
admin show_route_versions
```

示例

- **示例 1** – 在此示例中，repserver_1510.repserver_1500 的路由版本是 15.0.0。

```
admin show_route_versions
```

```
Source RepServer  Dest. RepServer  Route Version
------------------------------------------------
repserver_1510    repserver_1510   1500
```

用法

- 路由版本是源 Replication Server 和目标 Replication Server 节点版本中最低的版本。如果路由版本低于最低的节点版本，则需要执行路由升级。
- 版本号确定在混合版本环境中可以对路由使用哪些功能集。
- 对于每个路由，**admin show_route_versions** 显示源 Replication Server 的名称、目标 Replication Server 的名称及路由的版本。

权限

任何用户都可以执行此命令。

另请参见

- admin show_site_version （第 71 页）
- sysadmin fast_route_upgrade （第 359 页）

admin show_site_version

显示 Replication Server 的节点版本。

语法

```
admin show_site_version
```

示例

- **示例 1** – 在此示例中，Replication 节点版本是 15.1.0。

```
admin show_site_version
```

```
Site Version
-------------
1510
```

用法

显示 Replication Server 的节点版本。该节点版本确定您可以使用哪些 Replication Server 功能。设置该节点版本后，不能将其降级为更低的版本。

权限

任何用户都可以执行此命令。

另请参见

- sysadmin site_version （第 372 页）

admin sqm_readers

显示正在读取稳定队列的线程的读取点和删除点。

语法

```
admin sqm_readers, q_number, q_type
```

参数

- **q_number** – Replication Server 指派给队列的 ID 号。可在 **admin who, sqm** 命令的输出中找到此 ID 号。
- **q_type** – 队列的类型。入站队列的类型为 1。出站队列的类型为 0。

示例

- **示例 1** –

```
admin sqm_readers, 103, 1
```

```
RdrSpid     RdrType     Reader                              Index
-------     -------     ------                              -----
46          SQT         103:1 DIST LDS.pubs                     0
57          SQT         103:1 DSI 107 SYDNEY_DS.pubs2           1

First Seg.Block  Next Read  Last Seg.Block  Delete  WriteWait
---------------  ---------  --------------  ------  ---------
14.43            14.44      14.43                1          1
14.43            14.44      14.43                1          0
```

用法

- **admin sqm_readers** 报告正在读取入站队列的每个 Replication Server 线程的读取点和删除点。利用此信息，可以帮助您确定 Replication Server 无法从队列中删除消息的原因。
- Replication Server 无法删除正在读取队列的所有线程的最低删除点之外的点。删除点是第一个段块。
- 使用 **admin who, sqm** 命令可查找 *q_number*。
- admin sqm_readers 输出表的列说明对 **admin sqm_readers** 命令的输出列进行说明。

表 9. admin sqm_readers 的输出列说明

列	说明
RdrSpid	此读取器的唯一标识符。
RdrType	正在读取队列的线程的类型。
Reader	有关读取器的信息。有关此信息的完整说明，请参见 admin who 输出表的“Name”和“Info”列。
Index	此读取器的索引。
First Seg.Block	队列中第一个取消删除的段号和块号。此信息在转储队列时有用。
Next Read	下一个要从队列中读取的段、块和行。
Last Seg.Block	写入队列的最后一个段和块。此信息在转储队列时有用。
Delete	读取器是否允许删除。如果值为“1”，表示读取器允许删除。

列	说明
WriteWait	读取器是否正在等待写入。如果值为“1”，表示读取器正在等待写入。

权限

任何用户都可以执行此命令。

另请参见

- admin who （第 88 页）
- admin stats （第 73 页）

admin stats

显示有关 Replication Server 操作的信息和统计信息。

语法

```
admin {stats | statistics} [, sysmon | "all"
    | module_name [, inbound | outbound] [, display_name ]]
    [, server[, database]] | [instance_id]
    [, {display |, save} [, obs_interval] [, sample_period]]
```

参数

- **sysmon** – 仅显示已确定对性能和调优特别重要的那些计数器的统计信息。计数器几乎是从所有模块中挑选的。这是缺省值。
- **"all"** – 显示所有计数器的统计信息。
- **module_name** – 显示命名模块的计数器中的统计信息，其中 *module_name* 为 *cm*、*dsi*、*dist*、*dsiexec*、*repagent*、*rsi*、*rsiuser*、*serv*、*sqm*、*sqt*、*sts*、*rsh*、*sync* 等。可以使用 **rs_helpcounter** 来获取有效的模块名。
- **inbound | outbound** – *sqt* 或 *sqm* 的类型。如果既没有为 *sqt* 或 *sqm* 模块提供 **inbound**，也没有提供 **outbound**，Replication Server 将报告这两种队列的统计信息。
- **display_name** – 是计数器的名称。使用 **rs_helpcounter** 可获取有效的显示名。*display_name* 只与 *module_name* 一起使用。
- **server[, database]** – 如果要收集的统计信息与连接有关，则 *server* 必须是数据服务器，并且必须提供 *database*。如果要收集的统计信息与路由有关，则 server 必须是 Replication Server，并且不能指定 *database*。
- **instance_id** – 标识模块（如 SQT 或 SQM）的特定实例。若要查看实例 ID，请执行 **admin who** 并查看 *Info* 列。

  注意： 对于 *rsh* 模块，必须使用 *SPID*。若要查看 *SPID*，请执行 **admin who** 并查看 *Spid* 列。

实例 ID 0 表示 Replication Server 范围的统计信息。

- **display** – 在计算机屏幕上显示统计信息。这是缺省值。
- **save** – 在 RSSD 中保存统计信息。截断或保留旧采样数据，具体取决于当前的 **stats_reset_rssd** 设置。
- **obs_interval** – 指定采样周期内每个观测间隔的长度。如果不指定间隔，将只有一个间隔，长度等于采样周期。每个观测间隔必须至少为 15 秒。格式可以是以秒为单位的数值，也可以是“hh:mm[:ss]”。
- **sample_period** – 指示总的采样持续时间。缺省值为 0，它将报告当前计数器值。如果设置为非 0 值，将重置当前计数器值，然后在指定的采样周期内进行收集。格式可以是以秒为单位的数值，也可以是“hh:mm[:ss]”。

示例

- **示例 1** – 收集两分钟连接 108 的出站 SQT 统计信息，并将数据发送到 RSSD。

```
admin stats, sqt, outbound, 108, save, 120
```

- **示例 2** – 收集两小时连接 108 的出站 SQT 统计信息，并将数据发送到 RSSD。此外，还会将采样周期划分为长度为 30 秒的观测间隔。

```
admin stats, sqt, outbound, 108, save, 30, "02:00:00"
```

- **示例 3** – 为连接 102 的入站队列显示 SQM 和 SQMR 模块的统计信息。

```
admin stats, sqm, inbound, 102
```

```
Report Time:            10/31/05 02:14:17 PM
Instance                        Instance ID  ModType/InstVal
--------------------------      -----------  ----------------
SQM, 102:1 pds01.tpcc                   102                1

Monitor                 Obs     Last    Max     Avg ttl/obs
---------------------  ------  ------  ------  ------------
#*SegsActive                1       1       1             1

================================================================
=========
Instance                        Instance ID  ModType/InstVal
------------------------------  -----------  ----------------
SQMR, 102:1 pds01.tpcc, 0 SQT           102                11

Observer                                  Obs    Rate x/sec
----------------------                    ------  ----------
SleepsWriteQ                                   4           0
```

注意：在输出中，计数器名称前面的前缀提供该计数器的相关信息。例如，前面带有 # 表明未重置计数器，即使执行了 **admin stats, reset**；而前面带有 * 表明必须对计数器采样，而无论使用哪种 **stats_sampling** 设置。在本示例中，始终对 SegsActive 计数器进行采样，并且从不进行重置。

- **示例 4** – 收集 SQM 模块中 SleepsWriteQ 计数器的所有实例的统计信息。

```
admin stats, sqm, SleepsWriteQ
```

```
Report Time               10/31/05 02:17:03 PM

Instance                              Observer      Obs    Rate x/sec
------------------------------------  -------------  ----   ---
-------
SQMR, 101:0 edsprs01.edbprs01,
0, DSI   SleepsWriteQ      0            0
SQMR, 102:0 pds01.tpcc,
0, DSI          SleepsWriteQ      0            0
SQMR, 102:1 pds01.tpcc, 0, DSI          SleepsWriteQ     20
     0
SQMR, 103:0 rds01.tpcc, 0, DSI           SleepsWriteQ      0
      0
```

- **示例 5** – 开始采样并将统计信息保存到 RSSD 中，持续时间为 1 小时 30 分钟，间隔为 20 秒：

```
admin stats, "all", save, 20, "01:30:00"
```

用法

- 有三种类型的统计信息收集器：
  - Observer – 对事件发生次数进行计数。例如，Replication Server 使用观测器对观测来自 RepAgent 的命令的次数进行计数。
  - Monitor – 定期对值进行采样。例如，Replication Server 使用监控器对已发送命令的大小进行采样。
  - Counter – 收集监控器和观测器未观测的统计信息。计数器通常累计特定值的累积总值，包括完成特定任务所需的总毫秒数。例如，Replication Server 使用计数器累计从 RepAgent 接收两个命令之间所经过的时间。

  观测器、监控器和计数器观察四种类型的统计信息：观测次数、观测值总和、最后观测值和最大观测值。
- **admin stats** 输出包含以下信息的报告：
  - Instance – 模块的具体实现。
  - Instance ID – 给定模块实例的数字标识符。例如，两个不同 SQM 实例的实例 ID 可以是 102 和 103。
  - ModType/InstVal – 有某些情况下，一个实例可能具有多种版本或模块类型。例如，给定的 SQM 实例可能具有入站类型和出站类型。对于 SQM 实例，入站版本的模块类型为 1，出站版本的模块类型为 0。
  - Monitor、Observer 或 Counter – 显示正在观测的统计信息收集器的名称。例如，SleepsWriteQ。
  - Obs – 统计信息收集器在观测周期内的观测次数。
  - Last – 在观测周期内观测到的最后一个值。
  - Max – 在观测周期内观测到的最大值。
  - Total – 在观测周期内观测到的值的总和。
  - Avg ttl/obs – 在观测周期内观测到的值的平均值。该值是通过计算 Total/Obs 得出的。

  - Rate x/sec – 在给定观测周期内观测到的每秒变化量。观测器通过计算观测周期内的 Obs/秒数 得出该值。观测器和计数器通过计算观测周期内的 Total/秒数得出该值。
- 缺省情况下，**admin stats** 报告 sysmon 计数器的值。
- 缺省情况下，**admin stats** 不报告显示 0（零）次观测的计数器。要更改此行为，请将 **stats_show_zero_counters** 配置参数设置为 on。
- 如果在计算机屏幕上显示统计信息，则不会将其存储在 RSSD 中。同样，如果将统计信息存储在 RSSD 中，则不会在屏幕上显示它们。
- 如果使用 **admin stats...display_name** 显示特定计数器的统计信息，Replication Server 将始终显示该计数器的统计信息，即使 **stats_sampling** 为 off 及观测次数为零也是如此。
- 可以使用 **admin stats** 和独立模块名称来收集相关模块的统计信息。无法在 **admin stats** 命令中使用相关模块名称来收集统计信息。

独立模块	相关模块
数据服务器接口 (DSI)	DSI 执行程序 (DSIEXEC)
数据服务器接口 (DSI)	活动对象统计信息 (AOBJ)
稳定队列管理器 (SQM)	SQM 读取器 (SQMR)
线程同步 (SYNC)	SYNC 元素 (SYNCELE)

有关 Replication Server 模块的详细信息，请参见《Replication Server 管理指南第二卷》。

权限

任何用户都可以执行此命令。

另请参见

- configure replication server （第 181 页）

admin stats, backlog

按照段和块来报告入站和出站队列中等待分配的复制事务的数量。

语法

```
admin {stats | statistics}, backlog
```

示例

- **示例 1** – 报告积压在入站和出站队列中的事务。

```
admin stats, backlog
```

```
Report Time:                 10/31/05 02:17:01 PM

Instance                      Monitor          Obs  Last  Max  Avg
ttl/obs
----------------------------  ---------------  ---  ----  ---  --
---------
SQMR 101:0
edsprs01.edbprs01,  *SQMRBacklogSeg   0     0    0            0
      0, DSI
SQMR 102:0
pds01.tpcc,         *SQMRBacklogSeg   0     0    0            0
      0, DSI
SQMR 102:1
pds01.tpcc,         *SQMRBacklogSeg 695     3    3            1
      0, SQT
SQMR 103:0
rds01.tpcc,         *SQMRBacklogSeg   0     0    0            0
      0, DSI

================================================================
=========
Report Time:                 10/31/05 02:56:11 PM

Instance                       Monitor            Obs Last Max Avg
ttl/obs
----------------------------- ----------------- --- ---- --- ---
--------
SQMR 101:0 edsprs01.edbprs01, *SQMRBacklogBlock    0    0   0
       0
      0, DSI
SQMR 102:0
pds01.tpcc,         *SQMRBacklogBlock    0    0   0           0
      0, DSI
SQMR 102:1
pds01.tpcc,         *SQMRBacklogBlock  692   50  64          29
      0, SQT
SQMR 103:0
rds01.tpcc,         *SQMRBacklogBlock  251    0   2           0
      0, DSI

================================================================
=========
```

用法

- **admin stats, backlog** 输出以下信息:
  - Instance – 模块的具体实现。
  - Monitor – 监控器或计数器的名称。
  - Obs – 在观测周期内观测到的段数或块数。
  - Last – 在上一观测周期内观测到的段数或块数。
  - Max – 在观测周期内执行的任何观测中观测到的最大段数或块数。
  - Total – 在观测周期内观测到的所有段数或块数的总和。

- **admin stats, backlog** 收集 SQMRBacklogSeg 和 SQMRBacklogBlock 计数器的数据。
- 段为 1MB，块为 16K。

权限

任何用户都可以执行此命令。

admin stats, cancel

取消当前运行的异步命令。对于多个观测间隔，不会删除取消时已保存的数据。

语法

```
admin {stats | statistics}, cancel
```

用法

可以使用 **admin stats, cancel** 显式地终止当前运行的异步命令。当采样已在后台运行时，Replication Server 不允许运行其它采样命令。

权限

任何用户都可以执行 **admin stats, cancel**。

admin stats, {md | mem | mem_in_use}

报告与内存使用情况有关的信息。

语法

```
admin {stats | statistics}, {md | mem | mem_in_use}
```

参数

- **md** – 报告与 DIST 和 RSI 用户关联的消息传递统计信息。
- **mem** – 按照段大小报告当前使用的内存段数。
- **mem_in_use** – 报告当前使用的内存量（以字节为单位）。

示例

- **示例 1** – 报告使用的内存总量（以字节为单位）。

```
admin stats, mem_in_use
```

```
Memory_in_Use
```

```
-------------
14215074
```

用法

消息传递统计信息与 DIST 线程和 RSI 用户相关联。

权限

任何用户都可以执行此命令。

admin stats, reset

重置所有可重置的计数器。

语法

```
admin {stats | statistics}, reset
```

示例

- **示例 1** – 将所有计数器重置为 0。此命令不产生输出。

  ```
  admin stats, reset
  ```

用法

rs_statcounter.counter_status 列的位 0x10 指明能否重置计数器。为计数器设置该位后，不能使用 **admin stats, reset** 或任何其它命令重置该计数器。

权限

任何用户都可以执行此命令。

另请参见

- admin stats （第 73 页）
- admin stats, status （第 79 页）

admin stats, status

显示监控器和计数器的配置设置。

语法

```
admin {stats | statistics}, status
```

示例

- **示例 1** –

```
1> admin stats, status
2> go
```

```
Command in progress, sampling period 00:30:00, time elapsed
00:02:32
```

```
Sybase Replication Server Statistics Configuration
==================================================
Configuration              Default            Current
--------------------       ----------         --------
stats_sampling             off                on
stats_show_zero_counters   off                off
stats_reset_rssd            on                 on
```

用法

- 显示以下这些配置参数的缺省值和当前值:
  - **stats_sampling** – 指示是打开还是关闭采样。
  - **stats_show_zero_counters** – 指定是否显示自上次重置后观测次数为零的计数器。

权限

任何用户都可以执行 **admin stats, status**。

admin stats, {tps | cps | bps}

按照每秒的事务数、命令数或字节数报告当前吞吐量。

语法

```
admin {stats | statistics}, {tps | cps | bps}
```

参数

- **tps** – 指定 Replication Server 按照每秒的事务数报告当前吞吐量。
- **cps** – 指定 Replication Server 按照每秒的命令数报告当前吞吐量。
- **bps** – 指定 Replication Server 按照每秒字节数报告当前吞吐量。

示例

- **示例 1** – 显示按照每秒的命令数计算吞吐量的计数器。由于输出长度的原因，此处仅显示其中的一部分:

```
admin stats, cps
```

```
Report Time:            10/31/05 02:58:54 PM
Instance
Observer                                  Obs    Rate x/sec
----------------------------------   -----  ----------
REP AGENT, pds01.tpcc *CmdsRecv      69876           0

(1 row affected)
=============================================================================

Report Time:              10/31/05 02:58:54 PM
Instance
Observer                                       Obs    Rate x/sec
---------------------------------------   -----  -----------
SQM, 101:0 edsprs01.edbprs01 *CmdsWritten       0            0
SQM, 102:0 pds01.tpcc *CmdsWritten              0            0
SQM, 102:1 pds01.tpcc *CmdsWritten          69886           25
SQM, 103:0 rds01.tpcc *CmdsWritten          48174           17

(4 rows affected)
=============================================================================
=========

Report Time:            10/31/05 02:58:54 PM
Instance
Observer                                       Obs        Rate x/sec
---------------------------------------------- -----------
----------
SQMR, 101:0 edsprs01.edbprs01, 0, DSI *CmdsRead            0
         0
SQMR, 102:0 pds01.tpcc, 0, DSI *CmdsRead
0          0
SQMR, 102:1 pds01.tpcc, 0, SQT *CmdsRead
              50499         18
SQMR, 103:0 rds01.tpcc, 0, DSI
*CmdsRead               48144         17

(4 rows affected)
=============================================================================
=======

...
```

用法

- 在计算每秒的吞吐量时，Replication Server 根据上次使用 **admin stats, reset** 重置计数器后处理的事务数和经历的秒数来进行计算。
- 对于每种类型的计算，将由不同的模块来报告吞吐量：
  - 每秒事务数 – 由 SQT、DIST、DSI 和其它模块报告。
  - 每秒命令数 – 由 RepAgent、RSIUSER、SQM、DIST、DSI 和 RSI 模块报告。
  - 每秒字节数 – 由 RepAgent、RSIUSER、SQM、DSI 和 RSI 模块报告。SQM 按照每秒的字节数和块数来报告事务。

权限

任何用户都可以执行此命令。

admin time

显示 Replication Server 的当前时间。

语法

```
admin time
```

参数

- **None** –

示例

- **示例 1** –

```
admin time
```

```
Time
-------------------------
    Feb 15 2001 9:28PM
```

用法

- 在调试或检查延迟问题时，**admin time** 可用于确定计算机时间或时区差异。
- 此命名也可在脚本中用来确定 Replication Server 启动或完成任务的时间。

权限

任何用户都可以执行此命令。

admin translate

对某个值执行数据类型转换，以带有分隔符的文本格式显示结果。

语法

```
admin translate, value, source_datatype, target_datatype
```

参数

- **value** – 要转换的值的文字表示。

- **source_datatype** – 数据类型的名称（Replication Server 本机数据类型或用于说明 *value* 的内容和格式的数据类型定义）。
- **target_datatype** – 数据类型的名称（Replication Server 基本数据类型或作为转换所请求的输出的数据类型定义）。

示例

- **示例 1** – 此示例将 DB2 *TIMESTAMP* 值“1999-06-22-14.35.23.123456”转换为 Oracle *DATE* 值“22-Jan-99”。

```
admin translate, '1999-06-22-14.35.23.123456',
rs_db2_timestamp, rs_oracle_date
```

- **示例 2** – 此示例将 Adaptive Server 二进制值 0x1122aabb 转换为 Oracle 二进制值“1122aabb”。

```
admin translate, 0x1122aabb, 'binary(4)',
        'rs_oracle_binary(4)'
```

用法

- 根据源数据类型的基本数据类型的分隔要求来分隔 *value*。
- 如果 *source_datatype* 或 *target_datatype* 要求指定长度（例如，*char(255)*），则用单引号将数据类型名括起。
- 根据要测试的是类级别转换还是列级转换，源数据类型和目标数据类型可能有所不同。因此：
  - 对于类级别转换 – 使用 *source_datatype* 的已发布数据类型。
  - 对于列级别数据类型 – 使用 *source_datatype* 的已声明数据类型和 *target_datatype* 的已发布数据类型。
- 将 **admin translate** 与 Replication Server 的诊断版本一起使用，可跟踪转换过程中的错误。
- 有关支持的数据类型转换的信息，请参见《Replication Server 异构复制指南》。有关使用异构数据类型支持 (HDS) 转换数据类型的信息，请参见《Replication Server 管理指南第一卷》。

权限

任何用户都可以执行此命令。

另请参见

- alter replication definition （第 152 页）
- create replication definition （第 258 页）
- alter connection （第 109 页）
- create connection （第 214 页）

admin verify_repserver_cmd

检验 Replication Server 能否成功执行复制定义请求。

语法

```
admin verify_repserver_cmd, 'rs_api'
```

参数

- **rs_api** – 包含复制命令语言 (RCL) 命令和想要验证的所有相应参数的字符串。

  用单引号括起 *rs_api* 并将字符串中的每个单引号替换为两个单引号。

示例

- **示例 1** – 在此示例中，**admin verify_repserver_cmd** 测试在将旧的复制定义版本复制到目标（例如备用数据库或复制数据库）后是否可以使用 **alter replication definition** 删除复制定义中的列并成功挂起目标 DSI：

  ```
  admin verify_repserver_cmd, 'alter replication
  definition authors drop address, city, state, zip
  with DSI_suspended'
  ```

  如果 Replication Server 可以执行 **alter replication definition** 命令，则 Replication Server 将返回下面的消息：

  ```
  The replication definition command can be executed
  successfully.
  ```

- **示例 2** – 下面的示例显示使用 **admin verify_repserver_cmd** 查看是否可以从复制定义中删除不存在的列时所发生的情况：

  ```
  admin verify_repserver_cmd, 'alter replication
  definition authors_does_not_exist
  drop address, city, state, zip'
  ```

  Replication Server 将返回一条消息，说明名为“authors_does_not_exist”的复制定义不存在。

- **示例 3** – 下面的示例显示 **admin verify_repserver_cmd** 可以检测语法错误，例如在命令行中使用“columns”关键字：

  ```
  admin verify_repserver_cmd, 'alter replication
  definition authors drop columns address, city, state, zip
  with DSI_suspended'
  ```

  Replication Server 将返回一条消息，例如：

  ```
  Line 1, character 71: Incorrect syntax with the keyword
  'columns'.
  ```

- **示例 4** – 下面的示例显示 **admin verify_repserver_cmd** 可以检测引号的使用是否正确，例如使用双引号括起 'off'：

```
admin verify_repserver_cmd, ‘alter replication
definition authors replicate sqldml “off”’
```

Replication Server 将返回一条消息，例如：

```
Line 1, Incorrect syntax with the keyword 'off'.
```

正确的语法为：

```
admin verify_repserver_cmd, ‘alter replication
definition authors replicate sqldml ‘‘off’’’
```

用法

- 当 Replication Agent 将复制定义 RCL 发送到 Replication Server 进行执行而复制定义 RCL 无法执行时，Replication Agent 将会关闭。为避免出现这种情况，请在直接从主数据库执行该 RCL 之前使用 **admin verify_repserver_cmd** 验证 Replication Server 是否可以成功执行复制定义请求。如果 Replication Server 无法成功执行请求，它将返回错误。
- Replication Server 支持将 **admin verify_repserver_cmd** 用于与 **rs_send_repserver_cmd** 相同的复制定义命令：
  - **alter replication definition**
  - **create replication definition**
  - **drop replication definition**
  - **alter applied function replication definition**
  - **create applied function replication definition**
  - **alter request function replication definition**
  - **create request function replication definition**

权限

任何用户都可以执行此命令。

另请参见

- admin verify_repserver_cmd （第 84 页）
- alter replication definition （第 152 页）
- rs_send_repserver_cmd （第 546 页）
- sysadmin skip_bad_repserver_cmd （第 374 页）

admin version

显示 Replication Server 软件的版本号。

语法

```
admin version
```

示例

- **示例 1** –

```
admin version
```

```
Version
-----------------------------------------------------
Replication Server/15.0/P/Sun_svr4/OS 5.8/1/OPT/Wed
Jan  4 17:47:58 2006 Copyright 1992, 2006
```

用法

- Replication Server 的软件版本号就是该软件产品的版本级别。
- 软件版本号自身不能确定您可以使用 Replication Server 中的哪些功能。复制系统的系统版本号和 Replication Server 的节点版本号也决定了您可以使用哪些功能。
- Replication Server 的节点版本号可能等于或低于软件版本号。有关详细信息，请参见 **sysadmin site_version**。
- 复制系统的系统版本号可能等于或低于软件版本号。有关详细信息，请参见 **sysadmin site_version**。

权限

任何用户都可以执行此命令。

另请参见

- sysadmin site_version （第 372 页）
- sysadmin system_version （第 385 页）

admin version, "connection"

列出用户数据库的升级状态并确定在升级 Replication Server 后必须升级的用户数据库。

语法

```
admin version, "connection"
```

示例

- **示例 1** – 在升级的 Replication Server 上输入：

```
admin version, "connection"
```

将会看到用户数据库和数据服务器、数据库 ID、相应的 Replication Server 以及数据库状态的列表。例如：

```
dbid    Name         Controller RS   Status
------  -----------  --------------  ------------------
101     pds.pdb01    rs_12           Database needs upgrade
102     pds.pdb02    rs_12           Database is not accessible
103     rds.rdb01    rs_12           Database has been upgraded
```

用法

- 在升级的 Replication Server 上输入 **admin version, "connection"**。
- “数据库无法访问”状态表示 Replication Server 无法连接到此用户数据库，因为该数据库不可用，或者因为 Replication Server 用来连接到该数据库的维护用户 ID 没有足够的权限进行连接。

请参见《配置指南》中的“Fixing a Failed or Missed User Database Upgrade with sysadmin upgrade, "database"”（使用 sysadmin upgrade, "database" 修复失败或忽略的用户数据库升级）。

权限

任何用户都可以执行 **admin version, "connection"**。

另请参见

- sysadmin upgrade, "database" （第 387 页）

admin version, route

报告从当前 Replication Server 升级到目标 Replication Server 或从源 Replication Server 升级到当前 Replication Server 的路由，并检查路由升级的状态。

语法

```
admin version, "route"
```

示例

- **示例 1** – 报告从当前 NY_RS Replication Server 到目标 LON_RS Replication Server 的路由升级状态：

```
admin version, "route"
```

如果：

- 路由升级失败，您需要从升级中恢复路由，则会看到以下内容：

```
Source    Desitnation     Route       Proposed  Status
                          Version     Version
-------   -------------   ----------  --------  -----------
NY_RS     LON_RS          1500        1570      need route
                                                 upgrade recovery
```

- 路由升级未继续，并且仍存在需要升级的路由，则会看到以下内容：

```
Source    Desitnation     Route       Proposed  Status
                          Version     Version
-------   -------------   ----------  --------  -----------
NY_RS     LON_RS          1500        1570      need route
                                                 upgrade
```

- 路由不需要升级或者路由升级成功，则不会在输出中列出该路由。

用法

- 在使用 **admin version, route** 时，可以看到以下内容：
  - 路由的源 Replication Server。
  - 路由的目标 Replication Server。
  - 路由的当前版本。
  - 预期升级到的建议路由版本。
  - 路由升级的状态。

请参见《配置指南》中的“升级路由”。

权限

任何用户都可以执行此命令。

admin who

显示有关 Replication Server 中正在运行的线程的信息。

语法

```
admin who [, {dist | dsi | rsi | sqm | sqt}[, no_trunc | ,connection
identifier1
[, connection identifier2] ...]]
```

参数

- **dist** – 返回有关分配器线程的信息。这些线程将入站队列中的事务分发到复制数据库和 Replication Server。
- **dsi** – 返回有关 DSI 线程的信息。这些线程将复制事务应用于数据库。

- **rsi** – 返回有关 RSI 线程的信息。这些线程将消息发送到其它 Replication Server。
- **sqm** – 返回有关 SQM 线程的信息。这些线程管理 Replication Server 稳定队列。
- **sqt** – 返回有关 SQT 线程的信息。这些线程将队列和组函数读入事务。
- **no_trunc** – 将 Info 列的大小从 40 个字符增加到 80 个字符。这在显示长数据服务器或数据库名称时十分有用。

  注意: 如果您指定了连接标识符，则无法使用 **no_trunc**。
- **连接标识符** – 过滤线程模块的 **admin who** 输出。根据线程模块，您可以将连接标识符与下面一项或多项组合
  - *db_id* – 数据库标识符，它是一个数字
  - *db_name* – 数据库名
  - *ds_name* – 数据服务器名
  - *q_number* – 稳定队列号
  - *q_type* – 稳定队列类型，其中 0 表示出站队列，1 表示入站队列
  - *rs_id* – Replication Server 标识符，它是一个数字
  - *rs_name* – Replication Server 名

表 10. admin who 线程和对应的连接标识符

线程	连接标识符	示例
DIST	通过提供以下任意一项来显示一个或多个特定连接的分配器 (DIST) 线程信息: • *db_id* • *ds_name* • *ds_name* 和 *db_name* 如果您仅指定数据服务器名作为连接标识符，**admin who** 将会列出属于该数据服务器的所有数据库的分配器信息。	显示有关数据服务器“ASE_01”和数据库“DB01”的 DIST 信息: `admin who, dist, ASE_01, DB01`
DSI	通过提供以下任意一项来显示一个或多个特定连接的数据服务器接口 (DSI) 线程信息: • *db_id* • *ds_name* • *ds_name* 和 *db_name* 如果您仅指定数据服务器名作为连接标识符，**admin who** 将会列出属于该数据服务器的所有数据库的 DSI 连接信息。	显示 *db_id* = 101 的 DSI 信息: `admin who, dsi, 101`

线程	连接标识符	示例
RSI	通过提供以下一项来显示特定连接的 Replication Server 接口 (RSI) 线程信息： • *rs_id* • *rs_name* 您可以通过 Replication Server 名或 Replication Server 标识符来指定连接。	显示 *rs_id* = 16777318 的 RSI 信息： `admin who, rsi, 16777318`
SQM	通过提供以下任意一项来显示一个或多个特定连接的稳定队列管理器 (SQM) 线程信息： • *db_name* 和 *db_name*，带或不带 *q_type* • *ds_name*，带或不带 *q_type* • *q_number*，带或不带 *q_type* • *rs_id*，带或不带 *q_type* • *rs_name*，带或不带 *q_type* 如果您不指定 *q_type*，则该命令会列出指定数据服务器的所有 **admin who, sqm** 条目的入站和出站队列类型。	显示 *q_type*=1（表示入站队列）的数据服务器“ASE_01”的 SQM 信息： `admin who, sqm, ASE_01, 1`
SQT	通过提供以下任意一项来显示一个或多个特定连接的稳定队列事务 (SQT) 线程信息： • *db_name* 和 *db_name*，带或不带 *q_type* • *ds_name*，带或不带 *q_type* • *q_number*，带或不带 *q_type* • *rs_id*，带或不带 *q_type* • *rs_name*，带或不带 *q_type* 如果您不指定 *q_type*，则该命令会列出指定数据服务器的所有 **admin who, sqt** 条目的入站和出站队列类型。	显示入站队列的数据服务器“ASE_01”、数据库“DB01”的 SQT 信息： `admin who, sqt, ASE01, DB01, 1`

示例

- **示例 1** – 在以下示例中，**admin who** 显示 Replication Server 中所有线程的状态。DSI 调度程序线程在输出中显示为“DSI”。DSI 执行程序线程显示为“DSI EXEC”。如果 Replication Server 启动时 DSI 挂起，即使配置了多个 DSI 执行程序线程，输出也仅显示一个 DSI 执行程序线程。

```
admin who
```

```
 Spid Name         State              Info
 ---- ----------   ------------------ --------------------------
  97  DIST         Active             103 LDS.pubs2
  98  SQT          Awaiting Wakeup    103:1  DIST LDS.pubs2
  68  SQM          Awaiting Message   103:0 LDS.pubs2
  89  DSI EXEC     Awaiting Message   106(1) SYDNEY_DS.pubs2sb
  91  DSI          Awaiting Command   106 SYDNEY_DS.pubs2sb
  21  DSI EXEC     Awaiting Message   101(1) TOKYO_DS.TOKYO_RSSD
```

```
  10  DSI          Awaiting Command    101 TOKYO_DS.TOKYO_RSSD
  16  DIST         Active              101 TOKYO_DS.TOKYO_RSSD
  17  SQT          Awaiting Wakeup     101:1  DIST TOKYO_DS.TOKYO_
RSSD
  15  SQM          Awaiting Message    101:1 TOKYO_DS.TOKYO_RSSD
  14  SQM          Awaiting Message    101:0 TOKYO_DS.TOKYO_RSSD
  30  REP AGENT    Awaiting Command    TOKYO_DS.TOKYO_RSS
      USER
   4  DSI EXEC     Awaiting Message    104(1) TOKYO_DS.pubs2
   0  DSI          Awaiting Command    104 TOKYO_DS.pubs2
   8  REP AGENT    Awaiting Command    TOKYO_DS.pubs2
      USER
  53  RSI          Awaiting Wakeup     SYDNEY_RS
  52  SQM          Awaiting Message    16777318:0 SYDNEY_RS
      RSI USER     Inactive            TOKYO_RS
  11  dSUB         Active
   6  dCM          Awaiting Message
   9  dAIO         Awaiting Message
  12  dREC         Active              dREC
  71  USER         Active              sa
  47  GATEWAY      Awaiting Command    SYDNEY_RS
   5  dALARM       Awaiting Wakeup
  13  dSYSAM        Sleeping
```

- **示例 2** – 在以下示例中，**admin who, dist** 命令显示 Replication Server 中的每个 DIST 线程的相关信息。

```
admin who, dist
```

```
Spid      State                     Info
-----     ----------------          ---------------------
21         Active                   102 SYDNEY_DS.SYDNEY_RSSD
22         Active                   106 SYDNEY_DS.pubs2

PrimarySite  Type   Status   PendingCmds   SqtBlocked
-----------  ----   ------   -----------   ----------
102          P      Normal             0            1
106          P      Normal             0            1

Duplicates   TransProcessed   CmdsProcessed   MaintUserCmds
----------   --------------   -------------   -------------
         0              715            1430               0
       290                1             293               0

NoRepdefCmds   CmdsIgnored   CmdMarkers   RSTicket   SqtMaxCache
------------   -----------   ----------   --------   -----------
           0             0            0          0             0
           0             0            1          0             0
```

- **示例 3** – 在此示例中，**admin who, dsi** 显示在 Replication Server 中运行的每个 DSI 调度程序线程的相关信息。

```
admin who, dsi
```

```
 Spid                 State                          Info
 -----       ----------------          ----------------------
     8       Awaiting Message          101 TOKYO_DS.TOKYO_RSSD
    79       Awaiting Message          104 TOKYO_DS.pubs2
```

```
   145       Awaiting Message          105 SYDNEY_DS.pubs2sb

Maintenance User  Xact_retry_times   Batch    Cmd_batch_size
 ----------------  ----------------   -----    --------------
 TOKYO_RSSD_maint                3      on              8192
      pubs2_maint                3      on              8192
      pubs2_maint                3      on              8192

Xact_group_size        Dump_load     Max_cmds_to_log
 ---------------        ---------     ---------------
           65536              off                  -1
           65536              off                  -1
           65536              off                  -1

Xacts_read       Xacts_ignored    Xacts_skipped
----------       -------------    -------------
        39                   0                0
         0                   0                0
      1294                   2                0
```

```
Xacts_succeeded       Xacts_failed    Xacts_retried  Current Orig
in DB
---------------       ------------    -------------  ------------
-----
             0                28               0               102
             0                 0               0                 0
             0                93               0               104

Current Origin QID     Subscription Name       Sub Command
-------------------    -----------------       -----------
    0x000000000...                  NULL              NULL
    0x000000000...                  NULL              NULL
    0x000000000...                  NULL              NULL

Current Secondary QID      Cmds_read      Cmds_parsed_by_sqt
---------------------      ---------      ------------------
                 NULL            129                       0
                 NULL              0                       0
                 NULL           6740                       0

IgnoringStatus   Xacts_Sec_Ignored   GroupingStatus   TriggerStat
us
--------------   -----------------   --------------   -----------
--
      Applying                   0              on             on
      Applying                   0              on             on
      Applying                   0             off            off

ReplStatus     NumThreads     NumLargeThreads     LargeThreshold
----------     ----------     ---------------     --------------
        on              1                   0                100
        on              1                   0                100
       off              3                   1                 20

CacheSize  Serialization    Max_Xacts_in_group  Xacts_retried_blk
---------  ---------------  ------------------  -----------------
```

```
0          wait_for_commit                  20                    0
0          wait_for_commit                 200                    0
0          wait_for_start                   20                    0

CommitControl                    CommitMaxChecks CommitLogChecks
-------------------------------- --------------- ---------------
on                                           400             200
on                                           400             200
on                                           400             200

CommitCheckIntvl IsolationLevel   dsi_rs_ticket_report   RSTicket
---------------- --------------   --------------------   --------
1000             default          on                     0
1000             default          on                     0
1000             default          on                     0
```

- **示例 4** – 在此示例中，**admin who, rsi** 显示有关 RSI 线程的信息。

```
admin who, rsi
```

```
 Spid     State              Info
 ----     ---------------    ----------------
   53     Awaiting Wakeup    SYDNEY_RS

 Packets Sent      Bytes Sent       Blocking Reads
 ------------      -------------    --------------
  3008.000000      624678.000000               269

 Locater Sent      Locater Deleted
 ------------      ---------------
 0x000000...        0x000000...
```

- **示例 5** – 在此示例中，**admin who, sqm** 显示有关 SQM 线程的信息。

```
admin who, sqm
```

```
    Spid      State                Info
    ----      ----------------     -------------------------
      14      Awaiting Message     101:0 TOKYO_DS.TOKO_RSSD
      15      Awaiting Message     101:1 TOKYO_DS.TOKYO_RSSD
      52      Awaiting Message     16777318:0 SYDNEY_RS
      68      Awaiting Message     103:0 LDS.pubs2

    Duplicates     Writes     Reads     Bytes
    ----------     ------     -----     -----
    0                             0         0
    0                          8867      9058
    0               0.1        2037      2037
    0               0.1.0         0         0



    B Writes   B Filled  B Reads   B Cache   Save_Int:Seg
    --------   -------   -------   -------   ------------
    0           0                        0           0:0
    0          34         44          2132          0:33
    0           3         54           268           0:4
    0           0         23             0       strict:O
```

```
    First Seg.Block     Last Seg.Block      Next Read
    ---------------     --------------      ---------
            0.1                 0.0             0.1.0
           33.10               33.10           33.11.0
            4.12                4.12            4.13.0
            0.1                 0.0             0.1.0

    Readers      Truncs
    -------      ------
          1           1
          1           1
          1           1
          1           1
```

- **示例 6** – 在此示例中，**admin who, sqt** 显示有关 SQT 线程的信息。

```
admin who, sqt
```

```
Spid    State              Info
----    ---------------    -------------------------
  17    Awaiting Wakeup    101:1 TOKYO_DS.TOKYO_RSSD
  98    Awaiting Wakeup    103:1 DIST LDS.pubs2
  10    Awaiting Wakeup    101 TOKYO_DS.TOKYO_RSSD
   0    Awaiting Wakeup    106 SYDNEY_DSpubs2sb

Closed       Read       Open       Trunc
------       ----       ----       -----
     0          0          0           0
     0          0          0           0
     0          0          0           0
     0          0          0           0

Removed   Full   SQM Blocked    First Trans   Parsed
-------   ----   -----------    -----------   -------
      0      0             1              0         0
      0      0             1              0         0
      0      0             0              0         0
      0      0             0              0         0

SQM Reader   Change Oqids   Detect Orphans
----------   ------------   --------------
         0              0                0
         0              0                0
         0              0                1
         0              0                1
```

- **示例 7** – 在此示例中，有一个从 RS_NY 主 Replication Server 到 RS_LON 复制 Replication Server 的专用路由，用于 NY_DS.pdb1 主连接。在两个 Replication Server 中输入 **admin who**，您会看到：
  - 在 RS_LON 上：

```
admin who
```

```
Spid Name       State              Info
45 SQT          Awaiting Wakeup    103:1  DIST NY_DS.pdb1
13 SQM          Awaiting Message   103:1 NY_DS.pdb1
```

```
32 REP AGENT   Awaiting Command   NY_DS.pdb1
16 RSI         Awaiting Wakeup    RS_LON
11 SQM         Awaiting Message   16777318:0 RS_LON
55 RSI        Awaiting Wakeup    RS_LON(103) /* Dedicated RSI
thread */
53 SQM         Awaiting Message   16777318:103 RS_LON(103) /
*Dedicated RSI outbound queue */
```

- 在 RS_NY 上:

```
admin who
```

```
Spid Name     State               Info
37 RSI USER   Awaiting Command     RS_NY(103) /*Dedicated RSI
user */
32 RSI USER   Awaiting Command    RS_NY
```

用法

- 如果使用带有选项的 **admin who**，必须在选项之前使用逗号。
- 如果您指定了连接标识符，但 Replication Server 找不到满足条件的信息，则输出不会显示任何记录。
- 要显示有关 Replication Server 中所有线程的信息，请执行不带任何选项的 **admin who**。

admin who 的输出列说明

如果执行不带选项的 **admin who**，将显示 *spid*、*Name*、*State* 和 *Info* 列。如果执行该命令时带有任一选项，也显示 *spid*、*State* 和 *Info* 列。

spid 列

这是 Replication Server 中正在运行的线程的唯一标识符。如果某线程被挂起或关闭，此字段将为空。

Name 和 *Info* 列

Name 是 Replication Server 线程的类型。*Info* 内容因线程类型而异。

表 11. admin who 输出的 Name 和 Info 列

Name	说明	*info* 列的内容
dAlarm	报警守护程序。此线程跟踪由其它线程设置的报警。例如，连接的淡出时间和预订守护程序重试间隔。	空
dAIO	异步 I/O 守护程序。它管理到 Replication Server 的稳定队列的异步 I/O。	空
dCM	连接管理器的守护程序。它管理与数据服务器和其它 Replication Server 的连接。	空

Name	说明	*info* 列的内容
dREC	恢复守护程序。此线程将在休眠一段时间（该时间可使用 **rec_daemon_sleep_time** 配置参数进行配置）后，启动 *rs_recovery* 表中指定的所有恢复操作。	空
dSUB	预订重试守护程序。此线程在一定超时周期（可通过 **sub_daemon_sleep_time** 配置参数配置该周期）后被唤醒，并尝试重新启动所有失败的预订。	空
dSYSAM	SySAM 守护程序。此线程跟踪检出的许可证。	空
dVERSION	版本守护程序。在 Replication Server 升级后首次启动时，将短暂激活此线程。它将 Replication Server 的新软件版本号传递到 ID Server。	此 Replication Server 的版本。
DIST	分配器线程。每个主数据库都有一个分配器线程从入站队列读取事务，确定需要哪些预订，然后转发事务。	线程正在分发其更新的数据服务器和数据库的名称。
DSI	DSI 调度程序线程。此线程通过 SQT 读取稳定队列并通过 DSI 执行程序线程应用事务。	线程写入的数据服务器的名称。
DSI EXEC	DSI 执行程序线程。此线程对复制数据库执行事务，并对复制数据服务器返回的错误作出响应操作。	DSI 执行程序线程的 ID 和与之连接的数据服务器的名称。
GATEWAY	网关服务器线程。此线程将命令从客户端传送到服务器，并将服务器的应答返回给客户端。	充当网关服务器的 Replication Server 的名称。
REP AGENT USER	作为 RepAgent 线程的客户端连接。此线程检验 RepAgent 提交是否有效，并将它们写进入站队列。	主数据服务器的名称及 RepAgent 正在转发其日志的数据库的名称。
RSI	RSI 发送器。此线程将消息从一个 Replication Server 发送到另一个 Replication Server。	从中发送消息的 Replication Server 的名称。
RSI User	连接到此服务器的 Replication Server 的客户端连接线程。它将要发送到其它 Replication Server 或数据库的消息写入出站队列。	作为客户端连接到此服务器的 Replication Server 的名称。
RS User	用于在主 Replication Server 上创建或删除预订的 Replication Server 连接。	预订所有者的名称。

Name	说明	*info* 列的内容
SQM	稳定队列管理器。此线程管理一个 Replication Server 稳定队列。	队列号：Replication Server 或数据库的 ID。 队列类型：1 表示入站队列，0 表示出站队列。 所有其它数字是队列所针对的预订的 ID。 队列标识符：针对以下这些队列： • 对于用于将消息假脱机到另一个 Replication Server 的队列，队列标识符是另外一个 Replication Server 的名称。 • 对于用于将消息假脱机到数据库的队列，队列标识符是数据服务器和数据库的名称。 • 对于用于假脱机与正在创建或删除的预订相关的消息的队列，队列标识符是复制定义的名称和预订的名称。
SQT	稳定队列事务接口。此线程从稳定队列读取消息流，并按照提交顺序重新组合事务。分配器和 DSI 使用此线程。	与相应的 SQM 线程相同。
USER	执行 RCL 命令的客户端线程。	此客户端的登录名。

State 列

State 列包含线程执行状态。下表说明了 Replication Server 线程的有效状态。根据 DSI 线程是调度程序线程还是执行程序线程，对其状态进行了不同的定义。有关定义，请参见《Replication Server 故障排除指南》。

表 12. admin who 输出的状态列说明

State	说明
Active	当前正在处理某一命令。
Active, DSI timer	当前正在处理某一命令。**dsi_timer** 为 on。
	DSI 线程正在等待将批处理命令提交给复制数据服务器。
Awaiting Command	线程正在等待客户端发送命令。
Awaiting Command, DSI timer	线程正在等待客户端发送命令。**dsi_timer** 为 on。

State	说明
Awaiting Commit Order	线程正在等待它提交完成的事务的时机。
Awaiting I/O	线程正在等待 I/O 操作完成。
Awaiting Message	线程正在等待来自 Open Server™ 消息队列的消息。
Awaiting Message, DSI timer	线程正在等待来自 Open Server™ 消息队列的消息。**dsi_timer** 为 on。
Awaiting Wakeup	线程已经过休眠，正在等待唤醒。
Checking Condition	线程正在等待事件发生。
Connecting	线程正在连接。
Controlling Mem	线程正在执行内存控制。
Disconnecting	线程正在断开连接。
Down	线程还未启动或已终止。
Getting Lock	线程正在等待互斥锁。
Inactive	源 Replication Server 与目标 Replication Server 未连接时，RSI User 线程在路由的目标端的状态。
Initializing	正在初始化线程。
Invalid	线程处于未知状态。
Locking Resource	线程正在尝试锁定共享资源。
Not Running	线程正在执行清理以准备关闭。
Reading Disk	线程正在准备磁盘读取。
SkipUntil Dump	线程已收到 resync database 标记，并在 DSI 处理后续的 dump database 标记之前一直保持此状态。
Setting Condition	线程正在设置另一个线程唤醒的条件。
SkipUntil Resync	在执行 **skip to resync** 后线程将会恢复，并在线程收到 resync database 标记之前一直保持此状态。
Sleeping	线程正在限定的期限内放弃处理器时间。
Sleeping For Mem	线程正在休眠，直到内存可用。
Suspended	用户已将线程挂起。
Unlocking Resource	线程正在释放共享资源。

admin who, dist 的输出列说明

此命令返回的表包含 Replication Server 中每个 DIST 线程的行。

表 13. admin who, dist 的输出列说明

列	说明
PrimarySite	SQT 线程的主数据库的 ID。
Type	此线程是物理连接或逻辑连接。
Status	线程的状态是“normal”还是“ignoring”。
PendingCmds	此线程的待执行命令的数量。
SqtBlocked	线程是否正在等待 SQT。
Duplicates	线程已找到并删除的重复命令的数量。
TransProcessed	已由线程处理的事务的数量。
CmdsProcessed	已由线程处理的命令的数量。
MaintUserCmds	属于维护用户的命令的数量。
NoRepdefCmds	由于没有定义相应的表复制定义而删除的命令数。 对于热备份，可以让 Rep Server 创建复制定义。在多节点可用性 (MSA) 中，用户可以定义数据库复制定义。在上述任一情况中，如果复制数据源自于没有表复制定义的源，则计数器将增加并且复制数据将转到目标。
CmdsIgnored	在其状态变为“normal”之前被删除的命令的数量。
CmdMarkers	已处理的特殊标记的数量。
RSTicket	Replication Server **stats_sampling** 参数为 on 时由 DIST 线程处理的 **rs_ticket** 子命令数量。 最小值：0 最大值：2^{63}-1 缺省值：0
SqtMaxCache	数据库连接的 SQT（稳定队列事务接口）高速缓存的最大值（以字节为单位）。 缺省值为 0，表示将 **sqt_max_cache_size** 的当前设置作为连接的最大高速缓存大小。

admin who, dsi 的输出列说明

此命令返回的表包含 Replication Server 中运行的每个 DSI 调度程序线程的行。如果某个数据库存在 DSI 调度程序线程，但 **admin who, dsi** 输出中未显示该线程，请使用 **resume connection** 重新启动该数据库的数据服务器接口。

表 14. admin who, dsi 的输出列说明

列	说明
Maintenance User	应用事务的维护用户的登录名。
Xact_retry_times	错误操作是 RETRY_LOG 或 RETRY_STOP 时重试失败事务的次数。

列	说明
Batch	指示批处理选项是否打开。如果已打开，可将多个命令作为批处理提交到数据服务器。
Cmd_batch_size	可发送到数据服务器的一批输出命令的最大大小（以字节为单位）。
Xact_group_size	由源命令组成的事务组的最大大小（以字节为单位）。
Dump_load	指示转储/装载选项是否打开。此配置选项协调主数据库和复制数据库之间的转储。
Max_cmds_to_log	可记录在事务的例外日志中的命令的最大数量。如果值为 –1，表明可以记录无限多个命令。
Xacts_read	DSI 从出站稳定队列读取的事务的数量。此值应当随 DSI 应用事务而不断增加。可以使用此信息来监控活动频率。
Xacts_ignored	确定为重复事务的事务数。通常，由于某些事务以前已经应用，因此这些事务在启动时会被忽略。从 DSI 队列进行的删除操作被延迟，所以在启动时，就会检测重复事务并将其忽略。如果发现大量被忽略的事务，有可能是 *rs_lastcommit* 表已损坏。有关详细信息，请参见《Replication Server 故障排除指南》。
Xacts_skipped	通过用 **skip first transaction** 恢复连接而跳过的事务的数量。
Xacts_succeeded	成功对数据库应用的事务的数量。
Xacts_failed	失败的事务的数量。根据错误映射，有些事务可能会被写入例外日志。您应当检查例外日志。
Xacts_retried	已重试的事务的数量。
Current Origin DB	当前事务的原始数据库 ID。
Current Origin QID	如果状态为“Active”，此值为正在处理的事务的开始日志记录的原始队列 ID。否则，此值为已处理的上一个事务的开始日志记录的原始队列 ID。
Subscription Name	如果线程正在处理预订，此值为预订的名称。
Sub Command	如果线程正在处理预订，此值为预订命令：activate、validate、drop 或 unknown。
Current Secondary QID	如果线程正在处理以递增方式应用的基本预订，此列包含当前事务的队列 ID。
Cmds_read	从 DSI 队列中读取的命令的数量。
Cmds_parsed_by_sqt	在 DSI 队列读取之前，由 SQT 进行语法分析的命令的数量。
IgnoringStatus	如果 DSI 在等待标记时忽略事务，则包含“Ignoring”。如果 DSI 在执行数据库中的事务，则包含“Applying”。
Xacts_Sec_ignored	在热备份应用程序中，切换后被忽略的事务的数量。

列	说明
GroupingStatus	如果 DSI 在执行组中的事务，则包含“on”。如果 DSI 一次一个地执行事务，则包含“off”。
TriggerStatus	如果 **set triggers** 为 **on**，则包含“on”。如果 **set triggers** 为 **off**，则包含“off”。
ReplStatus	指示 Replication Server 是否复制数据库中的事务。对于备用数据库，缺省值为“off”。对于所有其他数据库，缺省值为“on”。
NumThreads	正在使用的并行 DSI 线程的数量。
NumLargeThreads	为用于大事务而保留的并行 DSI 线程的数量。
LargeThreshold	在并行 DSI 配置中，一个事务中允许包含的命令的数量，一旦超过此数量，此事务将被视为大事务。
CacheSize	数据库连接的 SQT 高速缓存的最大值（以字节为单位）。缺省值为 0，表示将 **sqt_max_cache_size** 参数的当前设置用作 SQT 高速缓存的最大值。
Serialization	使用并行 DSI 线程时用于保持串行一致性的方法。
Max_Xacts_in_group	组中包含的最大事务数。缺省值为 20。可以使用 **alter connection** 命令配置此数字。
Xacts_retried_blk	由于超过锁争用的最大检查次数 DSI 回退事务的次数。
CommitControl	指示提交控制是内部控制还是外部控制。如果是内部控制，则设置为 true。
CommitMaxChecks	指示在回退事务并重试之前的最大锁争用尝试次数。
CommitLogChecks	指示在记录消息之前的最大锁争用尝试次数。
CommitCheckIntvl	事务在发出锁争用检查之前等待的时间（以毫秒为单位）。
IsolationLevel	DSI 连接的数据库隔离级别。
RSTicket	Replication Server **stats_sampling** 参数为“on”时由 DSI 队列管理器处理的 **rs_ticket** 子命令数量。 缺省值为 0，表示将 **sqt_max_cache_size** 的当前设置作为连接的最大高速缓存大小。
dsi_rs_ticket_report	确定是否要调用函数字符串 **rs_ticket_report**。如果 **dsi_rs_ticket_report** 设置为 on，将调用 **rs_ticket_report** 函数字符串。 缺省值：off

admin who, rsi 的输出列说明

此命令显示将消息发送到其它 Replication Server 的 RSI 线程的相关信息。

表 15. admin who, rsi 的输出列说明

列	说明
Packets Sent	发送的网络包数。
Bytes Sent	发送的字节总数。
Blocking Reads	通过阻塞读取来读取稳定队列的次数。
Locater Sent	发送的上一条消息的定位器（包含队列段、块和行）。
Locater Deleted	接收者已确认并已被 Replication Server 删除的上一个定位器。

admin who, sqm 的输出列说明

此命令显示管理 Replication Server 稳定队列的 SQM 线程的相关信息。

表 16. admin who, sqm 的输出列说明

列	说明
Duplicates	已检测到并已忽略的重复消息的数量。启动时通常会有一些重复消息。
Writes	写入队列的消息的数量。
Read	从队列读取的消息的数量。此数量通常超过写入的数量，因为在启动时读取最后一个段，以确定从何处开始写入。而且，长事务可能会导致重新读取消息。
Bytes	已写入的字节数。
B Writes	已写入的 16K 块的数量。此值可能大于 *Bytes*/16K，因为有些 16K 块未被写满。您可以通过用 *Bytes* 除以 *B Writes* 来确定块的密度。
B Filled	由于已满而写入磁盘的 16K 块的数量。
B Reads	已读取的 16K 块的数量。
B Cache	高速缓存中已读取的 16K 块的数量。
Save_Int:Seg	*Save_Int* 间隔和 *Save_Int* 列表中最早的段。*Save_Int* 间隔是 SQM 段中的所有消息都已得到目标的确认后，Replication Server 维护此段的分钟数。 例如，如果值为 5:88，表示 *Save_Int* 间隔为 5 分钟，其中段 88 是 *Save_Int* 列表中最早的段。 此功能在复制系统失败的情况下提供冗余。例如，Replication Server 从其它 Replication Server 接收数据时可能丢失其磁盘分区。通过使用 *Save_Int* 功能，进行发送的 Replication Server 可以重新创建在 *Save_Int* 间隔期间保存的全部消息。 如果一个队列由多个读取器线程读取，则可能使用 *Save_Int* 值“strict”。Replication Server 维护 SQM 段，直至读取此队列的所有线程均已读取此段上的消息并将它们应用于各自的目标。

列	说明
First Seg.Block	队列中第一个取消删除的段号和块号。如果 *First Seg.Block* 和 *Last Seg.Block* 的数字不匹配，数据仍将保留在队列中等待处理。 此信息在转储队列时有用。有关详细信息，请参见《Replication Server 故障排除指南》。
Last Seg.Block	写入队列的最后一个段和块。如果 *First Seg.Block* 和 *Last Seg.Block* 的数字不匹配，数据仍将保留在队列中等待处理。 此信息在转储队列时有用。有关详细信息，请参见《Replication Server 故障排除指南》。
Next Read	下一个要从队列中读取的段、块和行。
Readers	正在读取队列的线程的数量。
Truncs	队列的截断点的数量。

admin who, sqt 的输出列说明

SQT 线程从稳定队列读取事务，然后按提交顺序将它们传递到 SQT 读取器。读取器既可以是 DIST 线程，也可以是 DSI 线程。

SQT 将它正在处理的事务存储在内存高速缓存中。此表中显示的 *Closed*、*Read*、*Open*、*Trunc* 和 *Removed* 列适用于 SQT 高速缓存中的事务。

表 17. admin who, sqt 的输出列说明

列	说明
Closed	SQT 高速缓存中已提交的事务的数量。这些事务已从稳定队列中读取，正在等待处理。
Read	已处理但尚未从队列中删除的事务的数量。
Open	SQT 高速缓存中未提交或未中止的事务的数量。
Trunc	事务高速缓存中的事务的数量。*Trunc* 是 *Closed*、*Read* 和 *Open* 列的总数。
Removed	其组成消息已从内存中删除的事务的数量。SQT 处理大事务时会发生这种情况。这些消息将从稳定队列中重新读取。
Full	指示 SQT 已耗尽其高速缓存中的内存。只要有仍在等待处理的已关闭或读取事务，此情况就不是问题。如果 SQT 高速缓存经常是满的，请考虑增加其配置大小。要执行此操作，请参见“alter connection”。
SQM Blocked	如果 SQT 正在等待 SQM 读取消息，则值为 1。除非没有已关闭的事务，否则此状态应是暂时的。

列	说明
First Trans	此列包含有关队列中第一个事务的消息，并可以用于确定其是否为未终止的事务。此列由三部分信息组成： • ST：后面是 O（已打开）、C（已关闭）、R（已读取）或 D（已删除） • Cmds：后面是第一个事务中的命令的数量 • qid：后面是第一个事务的段、块和行
Parsed	已进行语法分析的事务的数量。
SQM Reader	SQM 读取器句柄的索引。
Change Oqids	指示原始队列 ID 已被更改。
Detect Orphans	指示正在执行孤立检测。

权限

任何用户都可以执行此命令。

admin who_is_down

显示有关未运行的 Replication Server 线程的信息。

语法

```
admin who_is_down [, no_trunc]
```

参数

- **no_trunc** – 将 Info 列的大小从 40 个字符增加到 80 个字符。这在显示长数据服务器或数据库名称时十分有用。

示例

- **示例 1** –

```
admin who_is_down
```

```
Spid   Name            State          Info
----   --------------  -----------    ------------
       RSI              Suspended      SYDNEY_RS
```

用法

- **admin who_is_down** 输出中的 *Spid* 列始终为空。未运行的线程没有进程。

- 在 **admin health** 显示 Replication Server 可疑的情况下，执行 **admin who_is_down**。此命令的输出不列出状态为“Connecting”的线程，而这可能是线程运行情况可疑情况的原因。
- 有关此命令的输出的说明，请参见 **admin who**。

权限

任何用户都可以执行此命令。

另请参见

- admin health （第 52 页）
- admin who （第 88 页）
- admin who_is_up （第 105 页）

admin who_is_up

显示有关正在运行的 Replication Server 线程的信息。

语法

```
admin who_is_up [, no_trunc]
```

参数

- **no_trunc** – 将 Info 列的大小从 40 个字符增加到 80 个字符。这在显示长数据服务器或数据库名称时十分有用。

示例

- **示例 1** –

```
admin who_is_up
```

```
 Spid  Name        State              Info
  ----  --------    ---------------    --------------------
    97  DIST        Active             103 LDS.pubs2
    98  SQT         Awaiting Wakeup    103:1 DIST LDS.pubs2
    96  SQM         Awaiting Message   103:1 LDS.pubs2
    68  SQM         Awaiting Message   103:0 LDS.pubs2
    89  DSI EXEC    Awaiting Message   106(1) SYDNEY_DS.pubs2sb
    91  DSI         Awaiting Command   106 SYDNEY_DS.pubs2sb
    21  DSI EXEC    Awaiting Message   101(1) TOKYO_DS.TOKYO_RSSD
    10  DSI         Awaiting Command   101 TOKYO_DS.TOKYO_RSSD
    16  DIST        Active             101 TOKYO_DS.TOKYO_RSSD
    17  SQT         Active             101:1 DIST TOKYO_DS.TOKYO
    15  SQM         Awaiting Message   101:1 TOKYO_DS.TOKYO_RSSD
    14  SQM         Awaiting Message   103:1 TOKYO_DS.TOKYO_RSSD
    30  REP AGENT   Awaiting Command   TOKYO_DS.TOKYO_RSSD
        USER
```

```
     4  DSI EXEC   Awaiting Message  104(1) TOKYO_DS.pubs2
     9  dAIO       Awaiting Message
    12  dREC       Active            dREC
    61  USER       Active            sa
     5  dALARM      Awaiting Wakeup
```

用法

有关输出的说明，请参见 **admin who**。

权限

任何用户都可以执行此命令。

另请参见

- admin who （第 88 页）
- admin who_is_down （第 104 页）

allow connections

为指定数据库将 Replication Server 置于恢复模式。

语法

```
allow connections
```

用法

- 执行 **allow connections** 即可开始重放已重新装载的转储中的日志记录。
- 以独立模式启动 Replication Server，并对正在重放其日志的每个数据库执行 **set log recovery**。
- 执行 **allow connections** 后，Replication Server 仅接受从以恢复模式启动的 RepAgent 对指定数据库的连接请求。这可确保 Replication Server 在接收当前事务之前接收重放的日志记录。
- 如果以独立模式重新启动 Replication Server，并在没有先执行 **set log recovery** 命令的情况下执行 **allow connections**，Replication Server 将从独立模式转变为正常模式。
- 有关详细的恢复过程，请参见《Replication Server 管理指南第二卷》。

权限

allow connections 需要“sa”权限。

另请参见

- ignore loss （第 317 页）

- rebuild queues （第 320 页）
- set log recovery （第 330 页）

alter applied function replication definition

更改 **create applied function replication definition** 命令所创建的函数复制定义。

语法

```
alter applied function replication definition repdef_name
        {with replicate function named 'func_name' |
        add @param_name datatype[, @param_name datatype]… |
        add searchable parameters @param_name[, @param_name]… |
        send standby {all | replication definition} parameters}
         [with DSI_suspended]
```

参数

- **repdef_name** – 要更改的应用函数复制定义的名称。
- **with replicate function named 'func_name'** – 指定要在复制数据库上执行的存储过程的名称。*func_name* 是一个字符串，最大长度为 255 个字符。
- **add** – 将参数及其数据类型添加到应用函数复制定义。
- **@param_name** – 要添加到复制参数或可搜索参数列表中的参数名。每个参数名必须以 @ 字符开头。
- **数据类型** – 要添加到参数列表的参数的数据类型。有关数据类型及其语法的列表，请参见“数据类型”。Adaptive Server 存储过程和函数复制定义不能包含数据类型为 *text*、*unitext*、*rawobject* 和 *image* 的参数。
- **add searchable parameters** – 指定可在 **create subscription** 或 **define subscription** 命令的 **where** 子句中使用的其它参数。
- **send standby** – 在热备份应用程序中，指定是将此函数中的所有参数都发送到备用数据库 (**send standby all parameters**)，还是只将复制定义中指定的参数发送到备用数据库 (**send standby replication definition parameters**)。缺省值是 **send standby all parameters**。
- **with DSI_suspended** – 允许您挂起备份 DSI（如果有）和每个预订复制 DSI 线程。在 Replication Server 将旧复制定义版本的所有数据应用于备用数据库或复制数据库后，Replication Server 将会在备用数据库或复制数据库中挂起 DSI 线程。

  在 Replication Server 挂起 DSI 线程后，您可以对目标存储过程以及任何自定义函数字符串进行更改。当您恢复该 DSI 线程时，Replication Server 将使用变更的复制定义复制主更新。

  在以下情况下，您不需要使用 **with DSI_suspended**：

  - 没有对复制定义的预订。
  - 不需要更改自定义函数字符串。

- 不需要更改复制存储过程或备用存储过程。

注意： 如果有来自节点版本早于 1550 的复制 Replication Server 的预订，则不会挂起该 Replication Server 的复制 DSI 线程。

示例

- **示例 1** – 将 *@notes*、*@pubdate* 和 *@contract* 参数添加到 **titles_frep** 函数复制定义中：

```
alter applied function replication definition titles_frep
add @notes varchar(200), @pubdate datetime, @contract bit
```

- **示例 2** – 将 *@type* 和 *@pubdate* 参数添加到 **titles_frep** 函数复制定义的可搜索参数列表中：

```
alter applied function replication definition titles_frep
add searchable parameters @type, @pubdate
```

- **示例 3** – 在复制数据库中将要复制的 **titles_frep** 函数复制定义更改为 **newtitles** 存储过程并指示 Replication Server 在将执行 **alter applied replication definition** 之前已存在的主数据复制到复制数据库之后挂起目标 DSI：

```
alter applied function replication definition titles_frep
with replicate function named 'newtitles'
with DSI suspended
```

用法

- 使用 **alter applied function replication definition** 可以更改现有的应用函数复制定义。您可以添加复制参数和可搜索参数、选择要将哪些参数发送到热备份以及为要在复制数据库中执行的存储过程指定另一个名称。
- **alter applied function replication definition** 只能变更用 **create applied function replication definition** 命令创建的复制定义。
- 变更函数复制定义时，为函数复制定义指定的名称、参数和数据类型必须与要复制的存储过程相匹配。只有函数复制定义中指定的参数才会被复制。
- 同一存储过程的多个函数复制定义必须具有相同的参数列表。如果添加新参数，则会自动将新参数添加到为该存储过程创建的所有函数复制定义。
- 必须在主 Replication Server 上执行 **alter applied function replication definition**。
- 在任何子句中，一个参数名只能出现一次。
- 在添加参数时，必须指示 Replication Server 将 **alter applied function replication definition** 与函数复制定义的分发进行协调。此外，您必须指示 Replication Server 协调对存储过程和复制定义进行的更改。

  要变更复制定义，请参见《Replication Server 管理指南第一卷》的“管理复制的表”中的“复制定义更改请求过程”。
- 使用 **with replicate function named** 子句可以为要在复制数据库中执行的存储过程指定名称。请参见 **create applied function replication definition**。

有关 **alter applied function replication definition** 的详细信息，请参见《Replication Server 管理指南第一卷》。

权限

alter applied function replication definition 需要“create object”权限。

另请参见

- alter function string （第 143 页）
- alter replication definition （第 152 页）
- alter function replication definition （第 141 页）
- alter request function replication definition （第 159 页）
- create applied function replication definition （第 206 页）
- create request function replication definition （第 269 页）
- rs_send_repserver_cmd （第 546 页）
- rs_helprepversion （第 540 页）

alter connection

更改数据库连接的属性。

语法

```
alter connection to data_server.database {
     [for replicate table named [table_owner.]table_name
     [set table_param [to] 'value' ]] |
     set function string class [to] function_class |
     set error class [to] error_class |
     set replication server error class [to] rs_error_class |
     set password [to] passwd |
     set log transfer [to] {on | off} |
     set database_param [to] 'value' |
     set security_param [to] 'value' |
     set security_services [to] 'default']
     set dataserver and database name [to] new_ds.new_db |
     set trace [to] 'value'}
```

参数

- **data_server** – 数据服务器，其中包含要更改其连接的数据库。
- **database** – 要更改其连接的数据库。
- **for replicate table named** – 指定复制数据库的表的名称。*table_name* 是一个字符串，最多可包含 200 个字符。*table_owner* 是表名的可选限定符，表示表的所有者。如果实际表的所有者与复制定义中指定的所有者不对应，数据服务器操作可能会失败。

- **table_param** – 影响您使用 **for replicate table name** 指定的表的表级参数。

  有效值：**dsi_compile_enable** 和 **dsi_command_convert**。有关说明，请参见表 18. 影响数据库连接的参数。
- **function_class** – 用于数据服务器的函数字符串类。有关 Replication Server 为数据库连接提供的函数类列表，请参见“函数字符串变量修饰符”。
- **error_class** – 用于处理数据库错误的错误类。有关 Replication Server 为数据库连接提供的错误类列表，请参见“错误类和函数类”。
- **rs_error_class** – 用于处理数据库的 Replication Server 错误的错误类。有关 Replication Server 错误类列表，请参见“错误类和函数类”。
- **passwd** – 与用于数据库连接的登录名一起使用的新口令。如果未启用基于网络的安全性，则必须指定口令。
- **log transfer on** – 允许连接将事务从 RepAgent 发送到 Replication Server。
- **log transfer off** – 阻止连接从主数据库 RepAgent 发送事务。
- **database_param** – 影响与 Replication Server 的数据库连接的参数。
- **value** – 包含选项的新值的字符串。

  如果您使用 **trace** 选项，*value* 的语法将采用“*module, condition,[on|off]*”，其中：

  - *module* – 指定模块类型。有效值为 *econn*。
  - *condition* – 指定要设置的跟踪条件。
  - *on* 或 *off* – 指定所需条件的状态。

  注意： **alter connection** 命令中的 **trace** 参数允许空字符串。例如：

  ```
  alter connection to data_server.database
  set trace to ''
  ```

  空字符串会在连接后或在 Replication Server 重新启动时禁用 ExpressConnect 跟踪值。

表 18. 影响数据库连接的参数

database_param	说明和*值*
async_parser	使 Replication Server 可以异步分析来自 RepAgent 的命令。 将 **async_parser** 设置为 on 会将： • **exec_prs_num_threads** 设置为 2 • **ascii_pack_ibq** 设置为 on • **cmd_direct_replicate** 设置为 on • **dist_cmd_direct_replicate** 设置为 on 缺省值：off **注意：** 在配置异步语法分析程序之前，确保 **smp_enable** 设置为 on，并且 Replication Server 主机可以支持用于语法分析的附加线程。您必须将 Replication Server 节点版本设置为 1571 或更高版本，之后才能将 **ascii_pack_ibq** 设置为 on。如果节点版本早于 1571，则将 **async_parser** 设置为 on 只会设置 **exec_prs_num_threads**、**cmd_direct_replicate** 和 **dist_cmd_direct_replicate**。
ascii_pack_ibq	通过使用 ASCII 包，减少入站队列中的打包命令所消耗的稳定队列存储空间。 缺省值：off **注意：** 您必须为 Replication Server 启用异步语法分析程序才能在入站队列中从 ASCII 包受益。您必须将 Replication Server 节点版本设置为 1571 或更高版本，之后才能将 **ascii_pack_ibq** 设置为 on。
batch	指定 Replication Server 向数据服务器发送命令的方式。如果 **batch** 为“on”，Replication Server 可以将多个命令作为单个批处理命令发送到数据服务器。如果 **batch** 为“off”，Replication Server 一次一个地将命令发送到数据服务器。 缺省值：on
batch_begin	指示 **begin transaction** 是否可以与其它命令（如 **insert**、**delete** 等等）同批发送。 缺省值：on
cmd_direct_replicate	针对执行程序线程将 **cmd_direct_replicate** 设置为 on，以便将分析的数据连同二进制或 ASCII 数据一起直接发送到执行程序线程。当需要时，分配器线程可以直接从分析的数据中检索数据并进行处理，通过节省重新分析数据所花费的时间来提高复制性能。 缺省值：off

database_param	说明和*值*
dist_cmd_direct_replicate	将 **dist_cmd_direct_replicate** 设置为 on 可允许 DIST 模块通过内存高速缓存将内部分析的数据发送到 DSI。 缺省值：on 如果将 **dist_cmd_direct_replicate** 设置为 off，DIST 模块将通过出站队列将数据发送到 DSI。
command_retry	重试失败的事务的次数。此值必须大于或等于 0。 缺省值：3
db_packet_size	网络包的最大大小。数据库通信期间，网络包的值必须在数据库所接受的范围内。 缺省值：对于所有 Adaptive Server 数据库，网络包为 512 字节最大值：16,384 个字节
deferred_name_resolution	在 Replication Server 中启用延迟名称解析以便在 Adaptive Server 中支持延迟名称解析。只有 Adaptive Server 15.5 和更高版本中支持延迟名称解析。 在 Replication Server 中启用延迟名称解析支持之前，您必须确保复制 Adaptive Server 中支持延迟名称解析。 在使用 **alter connection** 或 **alter logical connection** 执行 **deferred_name_resolution** 后，挂起或恢复连接。 缺省值：off
disk_affinity	指定用于分配下一个分区的分配提示。输入一个分区的逻辑名称，在当前分区已满时应该将下一个段分配给此分区。 缺省值：off
dist_sqt_max_cache_size	入站队列的最大稳定队列事务 (SQT) 高速缓存大小。缺省值为 0，表示 **sqt_max_cache_size** 参数的当前设置用作连接的高速缓存最大大小。 缺省值：0 对于 32 位 Replication Server： • 最小值 – 0 • 最大值 – 2147483647 对于 64 位 Replication Server： • 最小值 – 0 • 最大值 – 2251799813685247

database_param	说明和*值*
dist_stop_unsupported_cmd	如果 **dist_stop_unsupported_cmd** 为 on，并且下游 Replication Server 不支持某个命令，DIST 将会自我挂起。如果为 off，DIST 将忽略不支持的命令。 无论 **dist_stop_unsupported_cmd** 参数的设置如何，如果 Replication Server 发现不能发送给较低版本 Replication Server 的命令的第一个实例，都会始终记录一个错误消息。 缺省值：off
dsi_alt_writetext	控制大对象更新发送到复制数据库的方式。值为： • dcany – 生成包含主键列的 **writetext** 命令。在使用 DirectConnect Anywhere™ 作为接口填充非 ASE 复制数据库时，此设置可防止全表扫描。 • off – 生成包含文本指针的 Adaptive Server **writetext** 命令。 缺省值：off **注意：** 如果您要使用 ExpressConnect 来连接非 ASE 复制数据库，则不需要配置 **dsi_alt_writetext** 数据库参数。
dsi_bulk_copy	为连接打开或关闭批量拷入功能。如果 **dynamic_sql** 和 **dsi_bulk_copy** 均设置为 on，Replication Server 会在适当时应用批量拷入，如果 Replication Server 无法使用批量拷入，则使用动态 SQL。 缺省值：off
dsi_bulk_threshold	事务中的连续 **insert** 命令数，在达到此数字时，将会触发 Replication Server 使用批量拷入。当稳定队列事务 (SQT) 遇到大批量 **insert** 命令时，它将在内存中保留指定数量的 **insert** 命令以确定是否应用批量拷入。由于这些命令保留在内存中，Sybase 建议您不要将该值配置为比 **dsi_large_xact_size** 配置值高太多。 Replication Server 对到 Sybase IQ 的实时装载 (RTL) 复制和到 Adaptive Server 的高容量自适应复制 (HVAR) 使用 **dsi_bulk_threshold**。如果编译后对一个表执行 **insert**、**delete** 或 **update** 操作的命令数少于指定数量，RTL 和 HVAR 将使用语言代替批量接口。 最小值：1 **注意：** 启用 RTL 或 HVAR 时不要设置为“1”，因为这会影响性能。 缺省值：20 配置级别：服务器、数据库 设置时，对服务器级使用 **configure replication server**，对数据库级使用 **alter connection**。 **注意：** 必须将 **dsi_compile_enable** 设置为“on”才能对 RTL 或 HVAR 使用 **dsi_bulk_threshold**。

database_param	说明和*值*
dsi_cdb_max_size	Replication Server 可为 HVAR 或 RTL 生成的最大净更改数据库大小（以兆字节为单位）。 • 缺省值 – 1024 • 最小值 – 0 • 最大值 – 2,147,483,647 在 HVAR 中，Replication Server 使用 **dsi_cdb_max_size** 作为阈值以便： • 使用连续复制模式复制的大型事务。 • 停止将小型可编译事务分组到需要大于 **dsi_cdb_max_size** 的净更改数据库的组中。 在 RTL 中，Replication Server 使用 **dsi_cdb_max_size** 通过完全增量编译以增量方式刷新大型事务组。
dsi_charset_convert	用于处理主 Replication Server 和复制 Replication Server 之间的数据和标识符字符集转换的规范。此参数适用于要在所讨论的 DSI 中应用的所有数据和标识符。值为： • on – 从主 Replication Server 字符集转换为复制 Replication Server 字符集；如果这两个字符集不兼容，将出错并关闭 DSI。 • allow – 在字符集兼容时转换；还会将任何未转换的更新应用于数据库。 • off – 不尝试转换。如果有不相同但兼容的字符集，并且不希望进行任何转换，在这种情况下就需要使用此选项。在预订实现过程中，“off”设置的行为与“allow”相同。 缺省值：on
dsi_cmd_batch_size	Replication Server 放在批处理命令中的字节数的最大值。 缺省值：8192 个字节
dsi_cmd_prefetch	允许 DSI 在等待数据服务器的响应时预先构建下一批命令，因此可提高 DSI 效率。如果您还调优数据服务器以增强性能，则在使用此功能时，您可能会获得额外的性能提高。 缺省值：off 在将 **dsi_compile_enable** 设置为“on”时，Replication Server 会忽略您为 **dsi_cmd_prefetch** 设置的内容。 许可证：在高级服务选件下单独许可。请参见《Replication Server 管理指南第二卷》的“性能调优”中的“Replication Server – 高级服务选件”。
dsi_cmd_separator	在批处理命令中分隔各个命令的字符。 缺省值：换行符 (\n) **注意：** 您必须在交互模式中更新此参数，而不是通过执行 DDL 生成的脚本或其它任何脚本。运行脚本不能重置 **dsi_cmd_separator**。

database_param	说明和*值*
dsi_command_convert	指定如何转换复制命令。以下操作的组合指定转换类型： • **d** – delete • **i** – insert • **u** – update • **t** – truncate • **none** – 无操作 **dsi_command_convert** 的操作组合包括 **i2none**、**u2none**、**d2none**、**i2di**、**t2none** 和 **u2di**。 必须键入数字 2。转换前的操作在 2 之前，转换后的操作在 2 之后。例如： • **d2none** – 不复制 **delete** 命令。 • **i2di**,**u2di** – 将 **insert** 和 **update** 都转换为 **delete** 后跟 **insert**，相当于自动更正。 • **t2none** – 不复制 **truncate table** 命令。 缺省值：**none** 也可以在表级别配置此参数。 设置时，对数据库级配置使用 **alter connection**，对表级配置使用带 **for replicate table named** 子句的 **alter connection**。 将 **dsi_command_convert** 设置为 **none** 可删除连接或表的当前 **dsi_command_convert** 设置。
dsi_commit_check_locks_intrvl	在执行下一个 **rs_dsi_check_thread_lock** 函数字符串之前 DSI 执行线程要等待的毫秒 (ms) 数。用于并行 DSI。 缺省值：1000 毫秒（1 秒） 最小值：0 最大值：86,400,000 毫秒（24 小时）
dsi_commit_check_locks_log	在记录警告消息前，DSI 执行程序线程执行 **rs_dsi_check_thread_lock** 函数字符串的次数。用于并行 DSI。 缺省值：200 最小值：1 最大值：1,000,000

database_param	说明和*值*
dsi_commit_check_locks_max	DSI 执行程序线程在回退其事务并重试之前，检查其是否阻塞复制数据库中的其它事务的次数的最大值。用于并行 DSI。 缺省值：400 最小值：1 最大值：1,000,000
dsi_commit_control	指定提交控制处理是由 Replication Server 使用内部表在内部进行处理 (on)，还是使用 *rs_threads* 系统表在外部进行处理 (off)。 缺省值：on
dsi_compile_enable	设置为“on”可在服务器级、数据库级或表级启用 RTL 或 HVAR。 缺省值： • off – 服务器级和数据库级。Replication Server 将使用连续日志顺序的逐行更改复制。 • on – 表级 设置时，对服务器级配置使用 **configure replication server**，对数据库级配置使用 **alter connection**，对表级配置使用带 **for replicate table named** 子句的 **alter connection**。 如果复制新行更改引发问题，则为受影响的表将 **dsi_compile_enable** 设置为“off”，例如表上的触发器要求对表执行的所有操作按日志顺序复制，因而不允许编译。 **注意：** 在表级将 **dsi_compile_enable** 设置为“off”之前，请在服务器级或数据库级将 **dsi_compile _enable** 设置为“on”。 在将 **dsi_compile_enable** 设置为“on”时，Replication Server 会忽略您为 **replicate_minimal_columns** 和 **dsi_cmd_prefetch** 设置的内容。 在服务器级、数据库级或表级执行 **dsi_compile_enable** 后，挂起并恢复连接。 许可证：在高级服务选件下单独许可。请参见《Replication Server 管理指南第二卷》的“性能调优”中的“高级服务选件”。

database_param	说明和*值*
dsi_compile_max_cmds	指定事务组的最大大小，以命令数为单位。当 HVAR 或 RTL 达到正在编译的当前组的最大组大小时，HVAR 或 RTL 启动新组。 但如果没有其它要读取的数据，那么即使组没有达到最多命令数，HVAR 或 RTL 也会立即将当前状态的组应用于复制数据库。HVAR 或 RTL 不会等待更多数据到达以便使组大小达到所设置的限制。 在 RTL 中，Replication Server 使用带有 **dsi_cdb_max_size** 的 **dsi_compile_max_commands** 在完全增量编译中触发组的增量刷新。 在 HVAR 中，Replication Server 使用带有 **dsi_cdb_max_size** 的 **dsi_compile_max_commands** 来检测大型事务，随后使用连续复制模式对这些事务进行复制。 缺省值：10,000 最小值：100 您可以在服务器级或数据库级配置此参数。 设置时，对服务器级使用 **configure replication server**，对数据库级使用 **alter connection**。 **注意：** 必须将 **dsi_compile_enable** 设置为“on”才能使用 **dsi_compile_max_cmds**。
dsi_compile_retry_threshold	为要在重试阶段为 HVAR 或 RTL 编译的事务组中的命令数指定阈值。 如果包含失败事务的组中的命令数为： • 小于 **dsi_compile_retry_threshold** 的值，Replication Server 将重新尝试在连续复制模式下处理该组。 • 大于 **dsi_compile_retry_threshold** 的值，Replication Server 将重新尝试使用 HVAR 处理该组，随后可能需要更多次重试来标识失败的事务。 缺省值：100 最小值：0 最大值：2,147,483,647
dsi_connector_type	指定用于实现连接器的数据库驱动程序技术。此参数与 **dsi_dataserver_make** 一起用于标识与连接相关联的连接器。如果您要复制到 ASE 或 IQ，请将此参数值设置为 *ctlib*，如果要复制到 Oracle，请将值设置为 *oci*。 缺省值：ctlib。 有效值：ctlib、oci。

database_param	说明和*值*
dsi_dataserver_make	指定包含您要连接到的复制数据库的数据服务器的类型。 可能的值有：ASE、IQ 和 ORA。 使用 **dsi_dataserver_make** 和 **dsi_connector_type** 可标识与连接相关联的连接器。 设置为 IQ 将复制到 Sybase IQ。设置为 ASE 将复制到 Adaptive Server，设置为 ORA 将复制到 Oracle。 您可以在数据库级配置 **dsi_dataserver_make**。 如果不指定此参数，Replication Server 将从数据库连接的函数字符串类名推导数据服务器类型。 如果函数字符串类是自定义的，Replication Server 将无法推导数据服务器类型，因此缺省为“ASE”。
dsi_exec_request_sproc	在主 Replication Server 的 DSI 上打开或关闭请求存储过程。 缺省值：on
dsi_fadeout_time	DSI 连接关闭之前的空闲时间（以秒为单位）。如果值为“-1”，说明连接将不会断开。 缺省值：600 秒
dsi_ignore_underscore_name	当事务分区规则设置为“name”时，请指定 Replication Server 是否忽略以下划线开头的事务名。值为“on”和“off”。 缺省值：on
dsi_isolation_level	指定事务的隔离级别。ANSI 标准和 Adaptive Server 支持的值包括： • 0 – 确保一个事务所写入的数据表示实际数据。 • 1 – 防止脏读，并确保一个事务所写入的数据表示实际数据。 • 2 – 防止非重复读取和脏读，并确保一个事务所写入的数据表示实际数据。 • 3 – 防止幻像行、非重复读取和脏读，并确保一个事务所写入的数据表示实际数据。 **注意：** 通过使用 **rs_set_isolation_level** 函数字符串，也为支持其它隔离级别的数据服务器提供支持。Replication Server 支持所有的复制数据服务器值。 缺省值是目标数据服务器的当前事务隔离级别。
dsi_keep_triggers	指定是否应为数据库中的复制事务引发触发器。 如果设置为“off”，将使 Replication Server 在 Adaptive Server 数据库中将触发器设置为关闭，这样对连接执行事务时就不会引发触发器。 对于除备用数据库外的所有数据库，设置为“on”。 缺省值：on（备用数据库除外）

database_param	说明和*值*
dsi_large_xact_size	事务中允许包含的命令的数量，一旦超过此数值，此事务将被视为大事务。 缺省值：100 最小值：4 最大值：2,147,483,647 当打开 **dsi_compile_enable** 时，将忽略此参数。
dsi_max_cmds_in_batch	定义可对其输出命令进行批处理的源命令的最大数。 必须挂起并恢复连接，参数中的任何更改才能生效。 范围：1 – 1000 缺省值：100
dsi_max_cmds_to_log	写入事务的例外日志的命令的数量。 缺省值：-1（所有命令） 有效值：0 到 2147483647
dsi_max_xacts_in_group	指定组中事务的最大数量。较大的数字可以改善复制数据库的数据延迟问题。值的范围：1 – 1000. 缺省值：20 当打开 **dsi_compile_enable** 时，将忽略此参数。
dsi_max_text_to_log	为失败事务中每个 **rs_writetext** 函数写入例外日志的字节数。请更改此参数以防止包含较大 *text*、*unitext*、*image* 或 *rawobject* 列的事务填满 RSSD 或其日志。 缺省值：-1（所有 *text*、*unitext*、*image* 或 *rawobject* 列）
dsi_non_blocking_commit	Replication Server 在提交后保存消息的分钟数。值 0 表示禁用非阻塞提交。 **注意：** 不能将此参数与 **alter connection** 一起使用以在备份环境中配置活动数据库连接。 缺省值：0 最大值：60
dsi_num_large_xact_threads	为用于大事务而保留的并行 DSI 线程的数量。最大值是 **dsi_num_threads** 的值减去 1。 缺省值：0
dsi_num_threads	要使用的并行 DSI 线程的数量。最大值为 255。 缺省值：1

database_param	说明和*值*
dsi_partitioning_rule	指定 DSI 用来将可用的并行 DSI 线程中的事务分区的分区规则（一个或多个）。值为 **origin**、**origin_sessid**、**time**、**user**、**name** 和 **none**。 有关详细信息，请参见《Replication Server 管理指南第二卷》的“性能调优”的“使用并行 DSI 线程”中的“分区规则：减少争用和增大并行度”。 缺省值：none 当打开 **dsi_compile_enable** 时，将忽略此参数。
dsi_proc_as_rpc	指定 Replication Server 如何应用存储过程复制。 • 设置为 on 可使用远程过程调用 (RPC) 调用。 • 设置为 off 可使用语言调用。 缺省值：off 当复制数据库是 Adaptive Server 时，**dsi_proc_as_rpc** 可以是 on 或 off。 当复制数据库是 Oracle 时： • 如果您使用 ExpressConnect for Oracle (ECO)，则设置为 on。ECO 仅支持使用 RPC 的存储过程复制。缺省情况下，如果您在创建从 Replication Server 到 Oracle 数据库的连接时使用一个 Oracle ECO 连接配置文件，Replication Server 会将 设置为 on。请参见“Replication Server Options 15.5”的“ExpressConnect for Oracle 15.5 Installation and Configuration Guide 15.5”（ExpressConnect for Oracle 15.5 安装和配置指南 15.5）中的“Configuring ExpressConnect for Oracle”（配置 ExpressConnect for Oracle）。 • 如果您使用 ECDA Option for Oracle，则设置为 off。ECDA 不支持将 RPC 用于存储过程复制。
dsi_quoted_identifier	为数据服务器接口 (DSI) 启用或禁用带引号的标识符支持。 缺省值：off
dsi_replication	指定由 DSI 应用的事务是否在事务日志中标记为正在复制。 当 **dsi_replication** 设置为“off”，DSI 在 Adaptive Server 数据库中执行 **set replication off**，以防止 Adaptive Server 将复制信息添加到 DSI 执行的事务的日志记录中。由于这些事务由维护用户执行，因此，通常不需要进一步复制（除非具有备用数据库），将此参数设置为“off”，可以避免将不必要的信息写入事务日志中。 对于复制数据库的热备份应用系统中的活动数据库，以及使用复制的合并复制应用模型的应用系统，**dsi_replication** 必须设置为“on”。 缺省值：on（对于热备份应用中的备用数据库，缺省值为“off”）

database_param	说明和*值*
dsi_replication_ddl	通过指定是否将事务复制回原始数据库来支持双向复制。 当 **dsi_replication_ddl** 设置为 on 时，DSI 向复制数据库发送 **set replication off**，这会指示复制数据库将随后的 DDL 事务标记为可用于系统日志而不进行复制。因此，这些 DDL 事务不会复制回原始数据库，这会在双向 MSA 复制环境中启用 DDL 事务复制。 缺省值：off
dsi_row_count_validation	如果您的表行不同步，并希望绕过缺省错误操作和消息，可以将 **dsi_row_count_validation** 设置为 **off** 来禁用行计数验证。 缺省值：**on**，启用行计数验证。 在为连接设置 **dsi_row_count_validation** 时，无需挂起并重新开始数据库连接；该参数会立即生效。不过，新设置将影响 Replication Server 在您执行命令后处理的一批复制对象。更改此设置不会影响 Replication Server 当前正在处理的一批复制对象。
dsi_rs_ticket_report	确定是否要调用函数字符串 **rs_ticket_report**。如果 **dsi_rs_ticket_report** 设置为 on，将调用 **rs_ticket_report** 函数字符串。 缺省值：on

database_param	说明和*值*
dsi_serialization_method	指定用于确定何时可以启动事务而又仍保持一致性的方法。在任何情况下，都会保留提交顺序。 这些方法按并行度的大小从高到低进行排序。并行度越大，将会导致在将并行事务应用于复制数据库时，这些并行事务之间的争用越多。若要减少争用，请使用 **dsi_partition_rule** 选项。 • **no_wait** – 指定事务一旦就绪即可启动，而不考虑其他事务的状态。 • **wait_for_start** – 指定某个事务可以在预定在它紧前面提交的事务启动后立即启动。 • **wait_for_commit** – 指定一个事务要等到其前一个预定提交的事务做好提交准备后才启动。 • **wait_after_commit** – 指定一个事务要等到其前一个预定提交的事务已完全提交后才启动。 **注意：** 如果 **dsi_commit_control** 设置为“on”，则只能将 **dsi_serialization_method** 设置为 **no_wait**。 保留下面这些选项，仅仅是为了向后兼容早期版本的 Replication Server： • **none** – 与 **wait_for_start** 相同。 • **single_transaction_per_origin** – 与 **dsi_partitioning_rule** 设置为 **origin** 的 **wait_for_start** 相同。 **注意：** 不再支持将 **isolation_level_3** 值作为序列化方法，但它与将 **dsi_serialization_method** 设置为 **wait_for_start** 并将 **dsi_isolation_level** 设置为 3 的操作相同。 缺省值：**wait_for_commit**
dsi_sqt_max_cache_size	出站队列的 SQT（稳定队列事务接口）高速缓存大小的最大值（以字节为单位）。 缺省值为“0”，表示将 **sqt_max_cache_size** 的当前设置作为连接的最大高速缓存大小。 缺省值：0 对于 32 位 Replication Server： • 最小值 – 0 • 最大值 – 2GB（2,147,483,648 字节） 对于 64 位 Replication Server： • 最小值 – 0 • 最大值 – 2 千万亿字节（2,251,799,813,685,247 字节）

database_param	说明和*值*
dsi_stage_all_ops	在配置 Replication Server 和 Sybase IQ InfoPrimer 集成时阻止指定表的编译。 .如果必须保留表历史记录，例如缓慢更改的维度 (SCD) 表，请将 **dsi_stage_all_ops** 设置为 on。 请参见《Replication Server 异构复制指南》的“Sybase IQ as Replicate Data Server”（Sybase IQ 作为复制数据服务器）的“Replication Server 和 Sybase IQ InfoPrimer 集成”的“参数”中的**“dsi_stage_all_ops”**。
dsi_text_convert_multiplier	更改复制节点的 *text* 或 *unitext* 数据类型列的长度。当 *text* 或 *unitext* 数据类型列由于字符集转换而必须扩展或收缩时，可使用 **dsi_text_convert_multiplier**。Replication Server 将主 *text* 或 *unitext* 数据的长度与 **dsi_text_convert_multiplier** 的值相乘，以确定复制节点处的 *text* 或 *unitext* 数据的长度。该参数的数据类型为 *float*。 • 如果字符集转换涉及扩大 *text* 或 *unitext* 数据类型列，请将 **dsi_text_convert_multiplier** 设置为大于或等于 1.0。 • 如果字符集转换涉及缩小 *text* 或 *unitext* 数据类型列，请将 **dsi_text_convert_multiplier** 设置为小于或等于 1.0。 缺省值：1
dsi_timer	使用 **dsi_timer** 配置参数可指定事务在主数据库提交的时间和事务在备用数据库或复制数据库提交的时间之间的延迟。在延迟期结束后，Replication Server 将按照提交顺序处理出站队列中的事务。 在使用 **alter connection** 或 **alter logical connection** 执行 **dsi_timer** 后，挂起或恢复连接。 以 hh:mm 格式指定延迟。 • 最大值：24 小时。 • 缺省值：00:00，表示没有延迟。 **注意：** Replication Server 不支持主数据库上的 RepAgent 或 Replication Agent 与具有要在其中执行 **dsi_timer** 的 DSI 连接的 Replication Server 之间存在时区差异。

database_param	说明和*值*
dsi_xact_group_size	置入一个分组事务的最大字节数，包括稳定队列开销。一个分组事务由多个事务组成，DSI 将这些事务作为单个事务加以应用。值为 –1 意味着无分组。 Sybase 建议将 **dsi_xact_group_size** 设置为最大值，并使用 **dsi_max_xacts_in_group** 控制组中的事务数。 **注意：** 此内容不适用于 Replication Server 15.0 版和更高版本。这是为了与旧版本的 Replication Server 保持兼容而保留的。 最大值：2,147,483,647 缺省值：65,536 个字节 当打开 **dsi_compile_enable** 时，将忽略此参数。
dump_load	仅在复制节点上设置为“on”，以便启用协调的转储。有关详细信息，请参见《Replication Server 管理指南第二卷》。 缺省值：off
dynamic_sql	为连接打开或关闭动态 SQL 功能。只有在将此参数设置为 on 时，与动态 SQL 有关的其它配置参数才会生效。 **注意：** 如果将 **dynamic_sql** 和 **dsi_bulk_copy** 均设置为 on，DSI 将应用批量拷入。如果未使用批量拷入，则会使用动态 SQL。 缺省值：off
dynamic_sql_cache_management	管理连接的动态 SQL 高速缓存。 值： • mru – 指定在达到 **dynamic_sql_cache_size** 后释放旧动态 SQL 准备语句，以便为新语句腾出空间。 • fixed – 指定在达到 **dynamic_sql_cache_size** 后停止分配新的动态 SQL 语句。 缺省值：fixed
dynamic_sql_cache_size	使 Replication Server 能够估算可使用连接的动态 SQL 的数据库对象数。可以使用 **dynamic_sql_cache_size** 限制数据服务器上的资源需求。 缺省值：100 最小值：1 最大值：65,535

database_param	说明和*值*
exec_cmds_per_time-slice	指定 LTI 或 RepAgent 执行程序线程在放弃 CPU 之前可以处理的 LTL 命令数。通过增大此值，您可以使 RepAgent 执行程序线程更长时间地控制 CPU 资源，这样可以提高从 RepAgent 到 Replication Server 的吞吐量。 使用 **alter connection** 在连接级别设置此参数。 请参见《Replication Server 管理指南第二卷》的“性能调优”中的“控制 RepAgent 执行程序可处理的命令数”。 缺省值：2,147,483,647 最小值：1 最大值：2,147,483,647
exec_max_cache_size	指定要为执行程序命令高速缓存分配的的内存量。 缺省值：对于 32 位和 64 位 Replication Server 为 1,048,576 字节。 对于 32 位 Replication Server： • 最小值 – 0 • 最大值 – 2,147,483,647 字节 对于 64 位 Replication Server： • 最小值 – 0 • 最大值 – 2,251,799,813,685,247 字节 设置时，对到 Replication Server 的所有数据库连接使用 **configure replication server**，或对特定的数据库连接使用 **alter connection**。 请参见《Replication Server 管理指南第二卷》的“性能调优”的“使用调优参数的建议”中的“Executor Command Cache”（执行程序命令高速缓存）。
exec_nrm_request_limit	指定可用于主数据库中等待规范化的消息的内存量。 在使用 **exec_nrm_request_limit** 之前使用 **configure replication server** 将 **nrm_thread** 设置为“on”。 最小值：16,384 个字节 最大值：2,147,483,647 字节 缺省值： • 32 位 – 1,048,576 字节 (1MB) • 64 位 – 8,388,608 字节 (8MB) 在更改 **exec_nrm_request_limit** 的配置后，挂起并恢复 Replication Agent。 许可证：在高级服务选件下单独许可。请参见《Replication Server 管理指南第二卷》的“性能调优”中的“Replication Server – 高级服务选件”。

database_param	说明和*值*
exec_prs_num_threads	通过为来自主数据库的特定连接启动多个分析程序线程来启用异步分析功能，并指定连接的异步分析程序线程数。 缺省值：0（禁用异步分析程序） 最小值：0 最大值：20 **注意：** 在配置异步语法分析程序之前，确保 **smp_enable** 设置为 on，并且 Replication Server 主机可以支持用于语法分析的附加线程。
exec_sqm_write_request_limit	指定可用于存放等待写入到入站队列的消息的内存量。 缺省值：1MB 最小值：16KB 最大值：2GB
md_sqm_write_request_limit	指定分配器可用于存放等待写入出站队列的消息的内存量。 **注意：** 在 Replication Server 12.1 中， **md_sqm_write_request_limit** 取代了 **md_source_memory_pool**。保留 **md_source_memory_pool** 是为了与旧版本 Replication Server 兼容。 缺省值：1MB 最小值：16KB 最大值：2GB
parallel_dsi	提供配置并行 DSI 线程的速记方法。 设置为“on”将配置以下值： • 将 **dsi_num_threads** 设置为 5 • 将 **dsi_num_large_xact_threads** 设置为 2 • 将 **dsi_serialization_method** 设置为“wait_for_commit” • 将 **dsi_sqt_max_cache_size** 设置为 1 MB（在 32 位平台上）和 20 MB（在 64 位平台上）。 如果设置为“off”，则将这些并行 DSI 值配置为它们的缺省值。 您可以将此参数设置为“on”，然后分别设置各个并行 DSI 配置参数，以对配置进行微调。 缺省值：off

database_param	说明和*值*
rep_as_standby	如果用 **sp_reptostandby** 标记了数据库并且 **rep_as_standby** 设置为 on，则会复制表复制定义未包含在数据库复制定义中的表。若要复制表，请将： • **rep_as_standby** 设置为 on • **send maint xacts to replicate** 设置为 false • **send warm standby xacts** 设置为 true 缺省值：off
replicate_minimal_columns	指定 Replication Server 是发送所有事务的所有复制定义列，还是仅发送在复制数据库中执行更新或删除操作所需要的复制定义列。 值为 On 和 Off。 当复制定义不包含 **replicate minimal columns** 子句或根本没有复制定义时，Replication Server 将使用此连接级参数。 **注意：** 如果复制定义具有 **replicate all columns** 并且 **replicate_minimal_columns** 连接属性设置为“on”，连接将复制最少列。 如果行的列值没有变化也要将所有列复制到目标数据库，请将 DSI 连接的 **replicate_minimal_ columns** 设置为“off”。 可以使用 **admin config** 显示 **replicate_minimal_columns** 配置信息。 在将 **dsi_compile_enable** 设置为“on”时，Replication Server 会忽略您为 **replicate_minimal_columns** 设置的内容。 请参见《Replication Server 管理指南第二卷》的“性能调优”中的“使用 replicate_minimal_columns 和动态 SQL”。
save_interval	消息成功传递到目标数据服务器之后，Replication Server 对其进行保存的分钟数。有关详细信息，请参见《Replication Server 管理指南第二卷》。 缺省值：0 分钟

database_param	说明和*值*
sqm_cmd_cache_size	Replication Server 可以在 SQM 命令高速缓存中存储的已分析数据的最大大小（以字节为单位）。 32 位 Replication Server： • 缺省值 – 1,048,576 • 最小值 – 0，禁用 SQM 命令高速缓存 • 最大值 – 2,147,483,647 64 位 Replication Server： • 缺省值 – 20,971,520 • 最小值 – 0 • 最大值 – 2,251,799,813,685,247 如果 **cmd_direct_replicate** 或 **sqm_cache_enable** 为 off，Replication Server 将忽略为 **sqm_cmd_cache_size** 设置的任何值。
sqm_max_cmd_in_block	指定每个 SQM 块中已分析数据可以与之关联的最大条目数。 缺省值：320 最小值：0 最大值：4096 将 **sqm_max_cmd_in_block** 的值设置为 SQM 块中的条目数。根据数据配置文件，由于块大小是固定的，而消息大小是无法预料的，因此每个块有不同的条目数。如果设置的值过大，则会浪费内存。如果设置的值过小，则会影响复制性能。 如果 **cmd_direct_replicate** 或 **sqm_cache_enable** 为 off，Replication Server 将忽略为 **sqm_max_cmd_in_block** 设置的任何值。

database_param	说明和*值*
sqt_max_prs_size	由 HVAR 和 RTL 的事务分析进程解包的命令消耗的最大内存（以字节为单位）。 对于 32 位 Replication Server： • 缺省值 – 2,147,483,647 (2GB) • 最小值 – 0 • 最大值 – 2,147,483,647 对于 64 位 Replication Server： • 缺省值 – 2,147,483,647 (2GB) • 最小值 – 0 • 最大值 – 2,251,799,813,685,247 在服务器级别使用 **configure replication server** 为所有连接设置该参数，或者在数据库级别使用 **alter connection** 为单独的连接设置该参数。数据库级别的缺省值为 0。如果您保留数据库级缺省值或重置为缺省值，Replication Server 将使用在服务器级别设置的值。 在升级到 Replication Server 15.7.1 和更高版本后，必须对 32 位和 64 位 Replication Server 将缺省值设置为 2GB。
stage_operations	在配置 Replication Server 和 Sybase IQ InfoPrimer 集成时，针对 Relication Server 设置为 on 可以将操作写入到指定连接的 staging 表。 请参见《Replication Server 异构复制指南》的“Sybase IQ as Replicate Data Server”（Sybase IQ 作为复制数据服务器）的“Replication Server 和 Sybase IQ InfoPrimer 集成”的“参数”中的**“stage_operations”**。
sub_sqm_write_request_limit	指定预订实现线程或取消实现线程用于存放等待写入出站队列的消息的内存。 缺省值：1MB 最小值：16KB 最大值：2GB
unicode_format	支持以 `U&"` 格式发送 Unicode 数据。 将 **unicode_format** 设置为下列值之一： • `string` – Unicode 字符转换为字符串格式。例如，`string`“hello”将作为“hello”发送出去。 • `ase` – Unicode 字符以 `U&’’` 格式发送出去。例如，`string`“hello”将作为“U&'\0068\0065\006c\006c\006f’’”发送出去。双字节 Unicode 值按 Adaptive Server Enterprise 所要求的网络顺序发送。 缺省值：`string`

database_param	说明和*值*
use_batch_markers	控制对 **rs_batch_start** 和 **rs_batch_end** 函数字符串的处理。如果将 **use_batch_markers** 设置为 on，将在每个批处理命令前面加上 **rs_batch_start** 函数字符串，而在每个批处理命令后面附加 **rs_batch_end** 函数字符串。 如果复制数据服务器需要在批处理命令的开头或结尾处发送其它 SQL，而该 SQL 语句未包含在 **rs_begin** 函数字符串中，此时才需要将 **use_batch_markers** 设置为 on。 缺省值：off

- **security_param** – 对连接的基于网络的安全性具有影响的参数。有关参数的列表和值的说明，请参见 **create route** 中的“影响基于网络的安全性的参数”表。
- **set security_services to 'default'** – 重置所有基于网络的连接安全性功能，使其与 Replication Server 的全局设置相匹配。
- **new_ds 和 new_db** – 连接的新数据服务器和数据库的名称。

  注意： *new_ds* 和 *new_db* 参数的值可以与您为 *data_server* 和 *database* 参数定义的值相同。

- **trace** – 允许在 DSI 级别进行 ExpressConnect 跟踪。
- **value** – 包含选项的新值的字符串。

  如果您使用 trace 选项，value 的语法将采用“module, condition,[on|off]”格式，其中：

  - *module* – 指定模块类型。有效值为 *econn*。
  - *condition* – 指定是将 trace 选项设置为 *on* 还是 *off*。
  - *on* 或 *off* – 指定所需条件的状态。

  注意： **alter connection** 命令中的 **trace** 参数允许空字符串。例如：

  ```
  alter connection to data_server.database
  set trace to ''
  ```

  空字符串会在连接后或在 Replication Server 重新启动时禁用 ExpressConnect 跟踪值。

示例

- **示例 1** – 将 TOKYO_DS 数据服务器中的 *pubs2* 数据库的函数字符串类更改为 *sql_derived_class*：

  ```
  suspend connection to TOKYO_DS.pubs2

  alter connection to TOKYO_DS.pubs2b
  set function string class to sql_derived_class

  resume connection to TOKYO_DS.pubs2
  ```

- **示例 2** – 更改 LTI 或 RepAgent 执行程序线程在必须将 CPU 让给其它线程之前可以处理的 LTL 命令数：

```
suspend connection to TOKYO_DS.pubs2
alter connection to TOKYO_DS.pubs2b
set exec_cmds_per_timeslice to ' 10'
resume connection to TOKYO_DS.pubs2
```

用法

- 使用 **suspend connection** 在更改连接之前挂起连接上的活动。
- 在创建连接的 Replication Server 上执行 **alter connection**。
- 使用 **log transfer off** 停止从主数据库传递数据之前，应确保没有为此数据库中的数据定义的复制定义。
- 要更改到 Replication Server 的路由，请使用 **alter route**。
- 使用 **set function string class [to]** *function_class* 可激活非 Sybase 数据服务器的类级别转换。
- 您可以使用 **alter connection** 参数设置缺省连接或替代连接的连接参数。
  为替代连接设置的任何值都将覆盖从缺省连接继承的值或缺省值。
- 在创建连接的 Replication Server 上执行 **alter connection**。

数据库连接参数

- 使用 **alter connection** 更改 DSI 或数据库连接的配置参数。要更改 DSI 配置参数值，请挂起与 DSI 的连接，更改此值，然后恢复与 DSI 的连接。此过程将使新值生效。
- Replication Server 配置参数存储在 *rs_config* 系统表中。可通过更新表中的行来修改某些参数。有关详细信息，请参见《Replication Server 管理指南第一卷》。
- 有关配置并行 DSI 线程的详细信息，请参见《Replication Server 管理指南第二卷》。
- 使用 **assign action** 允许对由于特定数据服务器错误而导致失败的事务进行重试。
- 更改函数字符串类之前，应确保原来的类及新类所需的所有函数字符串都已存在。
- 更改错误类之前，应确保存在新类。
- 可以为需要使用命令分隔符来识别命令结尾的数据服务器更改字符。
  如果已指定另一个不同的分隔符，并希望将其更改为原来的换行符，请输入 **alter connection** 命令，如下所示：

  ```
  alter connection to data_server.database
    set  to '<Return>'
  ```

  其中，在两个单引号之间按一下 Return 键，而不输入其他任何字符。

dsi_bulk_copy 参数

当 **dsi_bulk_copy** 为 on 时，SQT 会计算事务所包含的同一表中连续 **insert** 语句的数量。如果该数量达到 **dsi_bulk_threshold**，DSI 会：

1. 将数据批量拷入到 Adaptive Server，直至 DSI 达到不是 **insert** 的命令或属于不同复制表的命令。
2. 继续执行事务中的其余命令。

Adaptive Server 将在批量拷入操作结束时（如果操作成功）或在故障点处发送批量操作结果。

注意： 批量拷入的 DSI 实现支持多语句事务，从而允许 DSI 执行批量拷入，即使事务中包含不属于批量复制的命令。

dsi_partitioning_rule 参数

一次可指定多个分区规则。用逗号将这些值隔开，不包含空格。例如：

```
alter connection to data_server.database
  set dsi_partitioning_rule to 'origin,time'
```

dataserver and database name 参数

使用 **dataserver and database name** 参数，您可以将连接从使用一个连接器切换为使用另一个连接器。例如，如果您正在使用 ASE/CT-Lib 连接器和 DirectConnect™ for Oracle 复制到 Oracle，并且您想要将连接切换为使用 Oracle/OCI 连接器，您可能需要使用新的数据服务器和数据库名称。因为 Sybase interfaces 文件中为 DirectConnect/Oracle 指定的名称可能与 Oracle TNS 名称文件中的 Oracle 数据服务器名称不同。要进行更改，请执行以下操作：

1. 挂起连接。
2. 将连接设置 **dsi_dataserver_make** 更改为 *ora*，并将 **dsi_connector_type** 更改为 *oci*。
3. 将连接设置 **dataserver and database name** 更改为 **new_ds** 和 **new_db**
   其中：
   - *new_ds* – Oracle `tnsnames.ora` 文件中数据服务器的名称
   - *new_ds* – 数据库的名称

   注意： *new_ds* 和 *new_db* 参数的值可以与您为 *data_server* 和 *database* 参数定义的值相同。
4. 重新开始连接。

dump_load 参数

将 **dump_load** 设置为“on”之前，先创建 **rs_dumpdb** 和 **rs_dumptran** 函数的函数字符串。Replication Server 不会在系统提供的类或从这些类继承的派生类中生成这些函数的函数字符串。

save_interval 配置参数

设置 **save_interval** 以将事务保存在 DSI 队列中，以便在从备份中恢复数据库后进行重新同步时使用。如果设置存放复制数据或接收复制函数的数据库的热备份，设置保存间隔也非常有用。可以使用 **sysadmin restore_dsi_saved_segments** 来恢复积压的事务。

基于网络的安全性参数

- 连接两端必须使用具有相同安全性机制和安全性功能的兼容安全性控制层 (SCL) 驱动程序。数据服务器必须支持 **set proxy** 或等效的命令。

复制系统管理员负责为每个服务器选择和设置安全性功能。在尝试建立连接之前，Replication Server 不会查询远程服务器的安全性功能。如果连接两端的安全性功能不兼容，连接将失败。

- **alter connection** 为从 Replication Server 到目标数据服务器的出站连接修改基于网络的安全性设置。它用 **configure replication server** 覆盖缺省安全性参数集。
- 如果将 **unified_login** 设置为“required”，只有具有“sa”权限的复制系统管理员才可以在没有凭据的情况下登录到 Replication Server。如果安全性机制失败，复制系统管理员可以使用口令登录到 Replication Server 并禁用 **unified_login**。
- Replication Server 可具有多种安全性机制；所支持的每种机制都在 `libtcl.cfg` 文件中的 SECURITY 下列出。
- 消息加密进程占用的资源较多，会造成严重的性能下降。大多数情况下，明智的做法是只将某些连接的 **msg_confidentiality** 设置为“required”。或者，选择占用资源较少的安全性功能，如 **msg_integrity**。

使用 alter connection 更改维护口令

- 可以使用 **alter connection** 命令更改任何 DSI 连接的维护用户口令：

```
alter connection to data_server.database
set password to password
```

- 如果 Replication Server 使用 ERSSD 并且 *data_server.database* 与 ERSSD 名称匹配，请使用 **alter connection** 和 **set password** 更新 *rs_maintusers* 表，在 ERSSD 上发出 **sp_password**，并更新配置文件行 *RSSD_maint_pw_enc*。

权限

alter connection 需要“sa”权限。

另请参见

- admin show_connections （第 66 页）
- admin who （第 88 页）
- create alternate connection （第 202 页）
- create connection （第 214 页）
- configure replication server （第 181 页）
- create error class （第 228 页）
- create function string class （第 249 页）
- drop connection （第 297 页）
- resume connection （第 321 页）
- set proxy （第 331 页）
- suspend connection （第 334 页）

alter connector

更改数据库连接器的属性。

语法

```
alter connector dataserver_make.connector_type
set option [to] value
```

参数

- **dataserver_make** – 指示数据库服务器。
- **connector_type** – 指示用于实现连接器的连接器技术。
- **option** – 为连接器的各种跟踪选项提供选择。

  支持的选项有：

  - **trace**
  - **trace_logpath**
- **value** – 包含选项的新值的字符串。

  如果您使用 **trace** 选项，*value* 的语法将采用“*module, condition,[on|off]*”，其中：

  - *module* – 指定模块类型。有效值为 *econn*。
  - *condition* – 指定要设置的跟踪条件。
  - *on* 或 *off* – 指定所需条件的状态。

示例

- **示例 1** – 将所有 DSI 实例配置为使用 ASE/CT-Lib 连接器并启用 general_1 跟踪条件：

  ```
  alter connector "ase"."ctlib"
  set trace to "econn,general_1,on"
  ```
- **示例 2** – 在此示例中，将 *option* 参数设置为 **trace_logpath**，并且除了将 ASE/CT-Lib 连接器生成的所有跟踪消息写入到 Replication Server 日志文件外，还写入到特定于连接器的跟踪文件：

  ```
  alter connector "ase"."ctlib"
  set trace_logpath to "/sybase/sybase_rep/log/"
  ```

  通常，日志文件名由以下部分构成：

  - `ec`
  - *dataserver_make*
  - *connector_type*

- `.log`

dataserver_make 和 *connector_type* 是变量。它们的值取决于要使用的数据库类型以及关联的连接器技术。例如，为 ASE/CT-Lib 创建的特定于连接器的日志文件为 `ecasectlib.log`。

- **示例 3**

  要关闭将跟踪消息写入到特定于连接器的跟踪文件，请更改 **trace_logpath** 配置设置：

```
alter connector "iq"."ctlib"
set trace_logpath to "fully-qualified path name"
```

用法

- 在创建连接的 Replication Server 上执行 **alter connector**。
- 执行 **alter connector** 为使用指定连接器的所有连接打开跟踪。

另请参见

- alter connection （第 109 页）

alter database replication definition

更改现有的数据库复制定义。

语法

```
alter database replication definition db_repdef
         with primary at srv.db
         {[not] replicate DDL | [not] replicate setname setcont |
         [not] replicate [{SQLDML | DML_options} [in table_list]}
          [with dsi_suspended]
```

```
setname ::= {tables | functions | transactions | system procedures}
```

```
setcont ::= [in ([owner1.] name1 [, [owner2.] name2 [, ...]])
```

注意： *setname* 中的 functions 一词指用户定义的存储过程或用户定义的函数。

参数

- **db_repdef** – 数据库复制定义的名称。
- **server_name.db** – 主服务器/数据库组合的名称。例如：*TOKYO.dbase*。
- **[not] replicate DDL** – 指示 Replication Server 是否将 DDL 发送到预订数据库。如果未包含“replicate DDL”，或者此子句包含“not”，则不会将 DDL 发送到复制数据库。

- **[not] replicate setname setcont** – 指定是否将 *setname* 类别中指定的对象发送到复制数据库。对于表、函数、事务和系统过程，*setname* 类别最多分别包含一个子句。

  如果省略系统过程 *setname* 或包含 **not** 选项，则 Replication Server 不会复制系统过程。

  如果省略表、函数或事务 *setname* 或包含 **not** 选项，Replication Server 将复制 *setname* 类别的所有对象。

  由 *setname* 指定的过滤器类别取代当前的过滤器类别，如果是新类别，则将该过滤器类别添加到数据库复制过滤器中。
- **[not] replicate {SQLDML | DML_options} [in table_list]** – 通知 Replication Server 是否将 SQL 语句复制到 *table_list* 中定义的表。
- **SQLDML** – 指定以下数据操纵语言 (DML) 操作：
  - U – **update**
  - D – **delete**
  - I – **insert select**
  - S – **select into**
- **DML_options** – 以下 DML 操作的任意组合：
  - U – **update**
  - D – **delete**
  - I – **insert select**
  - S – **select into**

  当数据库复制模式设置为 **UDIS** 的任意组合时，RepAgent 发送 Replication Server 构建 SQL 语句所需的各个日志记录和信息。
- **owner** – 表的所有者或执行事务的用户。Replication Server 不会为函数或系统过程处理所有者信息。

  可以用由单引号括起的空格或星号替换 *owner*。
  - 空格 (‘ ‘) – 表示没有所有者。
  - 星号 (*) – 表示全部所有者。因此，例如，**.publisher* 表示所有名为 *publisher* 的表，而与其所有者无关。
- **name** – 表、函数、事务或系统过程的名称。

  可以用由单引号括起的空格或星号替换 *name*。
  - 空格 (‘ ‘) – 表示没有名称。例如，*maintuser.*’ ‘表示所有未命名的维护用户事务。
  - 星号 (*) – 表示所有名称。例如，*robert.** 表示 *robert* 拥有的所有表（或事务）。

- **with dsi_suspended** – 通知复制 Replication Server 挂起复制 DSI。可以用于重新同步数据库的信号需求。

示例

- **示例 1** – 更改数据库复制定义 *rep_1C* 以滤除 table2。复制 DSI 将被挂起：

```
alter database replication definition rep_1C
    with primary at PDS.pdb
    not replicate tables in (table2)
    with dsi_suspended
```

- **示例 2** – 为 *tb1* 和 *tb2* 表应用 **update** 和 **delete** 语句：

```
alter database replication definition dbrepdef
      with primary at ds1.pdb1
      replicate 'UD' in (tb1,tb2)
go
```

用法

- 在执行 **alter database replication definition** 时，Replication Server 将 *rs_marker* 写进入站队列中。直至该标记到达 DIST 后，**alter database replication definition** 才生效。这使 DIST 有时间合并数据库预订解析引擎 (DSRE) 中的更改。
- 更改数据库复制定义可能会使主数据库和复制数据库不同步。有关重新同步数据库的说明，请参见《Replication Server 管理指南第一卷》。

  SQL 语句复制
  - 如果在复制定义中未指定过滤器，则缺省过滤器为 **not replicate** 子句。若要更改 SQLDML 过滤器，请应用 **alter database replication definition**。您可以在 **replicate** 子句中指定一个或多个 SQLDML 过滤器。
  - 有关 SQL 语句复制的详细信息，请参见 **create database replication definition**。

另请参见

- create database replication definition （第 225 页）
- drop database replication definition （第 298 页）

alter encryption key

重新生成加密密钥。

语法

```
alter encryption key key_name regenerate
```

参数

- ***key_name*** – 要生成的加密密钥的名称。

  有效值:

  `rs_password_key`: 口令加密密钥

示例

- **示例 1** – 重新生成口令加密密钥:

```
alter encryption key rs_password_key regenerate
```

用法

- Replication Server 使用 `rs_encryptionkeys` RSSD 系统表中的 `rs_password_key` 行和 Replication Server 配置文件中的 **RS_random** 属性进行口令加密。
  使用 **alter encryption key** 命令可为 `rs_password_key` 行和 **RS_random** 属性重新生成随机值。RSSD 中的口令将自动使用新的加密密钥重新加密。

权限

alter encryption key 需要 sa 权限。

alter error class

通过从其它错误类中复制错误操作来更改现有错误类。

语法

```
alter [replication server] error class error_class
   set template to template_error_class
```

参数

- **replication server** – 指示错误类是 Replication Server 错误类，而不是数据服务器错误类。
- **error_class** – 要修改的错误类。
- **set template to template_error_class** – 可以使用此子句根据其它错误类来更新某个错误类。**alter error class** 将模板错误类中的错误操作复制到现有错误类中。

示例

- **示例 1** – 根据 **rs_sqlserver_error_class** 更改 **my_error_class**:

```
alter error class my_error_class
    set template to rs_sqlserver_error_class
```

- **示例 2** – 根据缺省 Replication Server 错误类 **rs_repserver_error_class** 更改 Replication Server 错误类 **my_rs_err_class**：

```
alter replication server error class my_rs_err_class
set template to rs_repserver_error_class
```

用法

- 可以使用 **alter error class** 命令和另一个错误类（作为模板）来更改错误类。**alter error class** 可将模板错误类中的错误操作复制到要更改的错误类中，并覆盖具有相同错误代码的错误操作。
- *rs_sqlserver_error_class* 是为 Adaptive Server 数据库提供的缺省错误类；而 *rs_repserver_error_class* 是为 Replication Server 提供的缺省错误类。最初，这两个错误类没有主节点。必须先在主节点上创建这些错误类，然后才能更改缺省错误操作。
- 可以使用 **create connection** 和 **alter connection** 命令，将非 Adaptive Server 错误类指派给非 Adaptive Server 复制数据库上的特定连接。
- 当 Replication Server 建立到非 ASE 复制服务器的连接时，Replication Server 会验证是否对连接启用从非 ASE 复制服务器返回本机错误代码的选项。如果未启用此选项，Replication Server 会记录警告消息，表明连接有效但错误操作映射可能不正确。

  请参见 Replication Server Options 文档中的“**ReturnNativeError**”以在 Enterprise Connect™ Data Access (ECDA) Option for ODBC 中为复制服务器设置此选项。
- 有关非 Adaptive Server 错误类列表，请参见表 29. 错误类和函数类。有关非 Adaptive Server 复制错误类的详细信息，请参见《Replication Server 管理指南第二卷》。

另请参见

- assign action （第 172 页）
- create error class （第 228 页）
- drop error class （第 299 页）

alter function

将参数添加到用户定义的函数中。

语法

```
alter function table_rep_def.function_name
    add parameters @param_name datatype
    [, @param_name datatype]...
```

参数

- **table_rep_def** – 使用用户定义的函数的复制定义的名称。
- **function_name** – 要更改的用户定义的函数的名称。
- **@param_name** – 要添加到用户定义的函数的参数列表中的参数名。参数名必须符合标识符规则，且前面必须带有 @ 符号。
- **数据类型** – 参数的数据类型。有关数据类型及其语法的列表，请参见“数据类型”。此参数不能是 *text*、*unitext*、*raw object* 或 *image*。

示例

- **示例 1** –

```
alter function publishers_rep.upd_publishers
 add parameters @state char(2)
```

  将名为 *state* 的整数参数添加到 *publishers_rep* 复制定义的 *upd_publishers* 函数中。

用法

- 执行 **alter function** 之前，停顿复制系统。可以使用 Replication Server Manager 或《Replication Server 故障排除指南》中介绍的过程来停顿系统。
- 用户定义的函数最多可以包含 255 个参数。
- 在更新期间更改函数可能会导致不可预料的结果。受影响的数据在更改函数之前应该处于停顿状态。
- 更改用户定义的函数后，可能还必须更改使用新参数的函数字符串。
- 为一个复制定义更改用户定义的函数的同时，也为主表中的所有复制定义更改了此函数。
- 不要将 **alter function** 用于复制函数。而应使用 **alter function rep def**。**alter function** 仅用于“RSSD 存储过程”中介绍的异步存储过程。

权限

alter function 需要“create object”权限。

另请参见

- admin quiesce_check （第 57 页）
- alter function string （第 143 页）
- create function （第 231 页）
- create function string （第 236 页）
- drop function （第 300 页）
- drop function string （第 302 页）

alter function replication definition

更改 **create function replication definition** 命令所创建的现有函数复制定义。

注意： 对 **create function replication definition** 和 **alter function replication definition** 的支持按计划已被中止。Sybase 建议您改用下面这些命令：

- **create applied function replication definition** 和 **alter applied function replication definition**
- **create request function replication definition** 和 **alter request function replication definition**

语法

```
alter function replication definition function_rep_def
{
   deliver as 'proc_name' |
   add @param_name datatype [, @param_name datatype]... |
   add searchable parameters @param_name[, @param_name]... |
   send standby {all | replication definition}
   parameters
}
```

参数

- **function_rep_def** – 要更改的函数复制定义的名称。
- **deliver as** – 指定要在传递复制函数的数据库上执行的存储过程的名称。*proc_name* 是一个字符串，最多可包含 200 个字符。如果不使用此可选子句，函数将作为一个与函数复制定义同名的存储过程进行传递。
- **add** – 指定函数复制定义的其它参数及其数据类型。
- **@param_name** – 要添加到复制参数或可搜索参数列表中的参数名。每个参数名必须以 @ 字符开头。
- **数据类型** – 要添加到参数列表的参数的数据类型。有关支持的数据类型及其语法的列表，请参见“数据类型”。Adaptive Server 存储过程和函数复制定义可能不包含数据类型为 *text*、*unitext* 和 *image* 的参数。
- **add searchable parameters** – 指定可在 **define subscription** 或 **define subscription** 命令的 **where** 子句中使用的其它参数。
- **send standby** – 在热备份应用程序中，指定是将此函数中的所有参数都发送到备用数据库 (**send standby all parameters**)，还是只将复制定义中指定的参数发送到备用数据库 (**send standby replication definition parameters**)。缺省值是 **send standby all parameters**。

示例

- **示例 1** – 将以下三个参数添加到 *titles_frep* 函数复制定义：名为 *@notes* 的 *varchar* 参数、名为 *@pubdate* 的 *datetime* 参数和名为 *@contract* 的 *bit* 参数：

```
alter function replication definition titles_frep
 add @notes varchar(200), @pubdate datetime,
 @contract bit
```

- **示例 2** – 将 *@type* 和 *@pubdate* 参数添加到 *titles_frep* 函数复制定义的可搜索参数列表中：

```
alter function replication definition titles_frep
 add searchable parameters @type, @pubdate
```

- **示例 3** – 更改 *titles_frep* 函数复制定义，以使其作为 *newtitles* 存储过程在目标数据库（通常为主数据库，用于传递请求函数）中传递。

```
alter function replication definition titles_frep
 deliver as 'newtitles'
```

用法

- **alter function replication definition** 通过以下方法更改函数复制定义：添加复制参数，添加可搜索参数，指定是否将所有参数发送到热备份，或者为要在目标数据库中执行的存储过程指定另一个名称。
- 指定给要更改的函数复制定义的名称、参数和数据类型必须与要复制的存储过程相匹配。您可以只指定那些在复制过程中要用到的参数。
- 必须在管理主数据库（在该数据库上创建了函数复制定义）的 Replication Server 上执行 **alter function replication definition**。
- 在任何子句中，一个参数名只能出现一次。
- 如果要添加参数，应通过对函数复制定义进行分发来协调 **alter function replication definition**。请按照“更改函数复制定义”中的步骤进行操作，以避免发生错误。
- 可以使用可选的 **deliver as** 子句指定要在目标数据库（在该数据库上传递复制函数）上执行的存储过程的名称。通常会在请求函数传递中使用此选项。有关详细信息，请参见 **create connection**。

  有关 **alter function replication definition** 的详细信息，请参见《Replication Server 管理指南第一卷》。

更改函数复制定义：

1. 使用 Sybase Central Replication Manager 插件或《Replication Server 故障排除指南》中所述的步骤，停顿复制系统。

   最理想的情况是，应当首先停顿主数据库的更新，确保复制系统已对所有主数据库更新进行了处理。如果无法做到这一点，则主日志中旧的更新将没有新参数的值，而复制系统将使用 null 值代替。在下面的步骤 4 中更改函数字符串时，可能就需要考虑这个问题。
2. 更改主节点和复制节点上的存储过程。

3. 更改函数复制定义。等待经过修改的函数复制定义到达复制节点。
4. 如果需要，更改任何属于函数复制定义的函数字符串。等待经过修改的函数字符串到达复制节点。
5. 如果需要，修改复制节点上对此函数复制定义的预订。要修改预订，请使用 **drop subscription** 删除预订，然后使用 **create subscription**（不带实现选项）重新创建预订。

   更改复制定义不会影响当前预订。如果将新参数添加到函数复制定义中，新参数的复制将包括全部现有预订的所有新的更新。
6. 恢复对主数据库中数据的更新。

权限

alter function replication definition 需要“create object”权限。

另请参见

- alter function string （第 143 页）
- create function replication definition （第 232 页）
- drop function replication definition （第 301 页）

alter function string

替换现有的函数字符串。

语法

```
alter function string {replication_definition |
      [owner.]table |
      stored_procedure}.function[;function_string]
      for {[function_class] function_class |
      [database] data_server.database}
      [scan 'input_template']
   [output
      {language 'lang_output_template' | rpc 'execute procedure
       [@param_name=]{constant |?variable!mod?}
       [, [@param_name=]{constant |?variable!mod?}]...' |
       writetext [use primary log | with log |
       no log] |
        none}]
```

示例

- **示例 1**

  在 NY_DS 数据服务器的 rdb1 目标数据库中更改 authors 表的 **rs_update** 自定义函数字符串：

  ```
  alter function string authors.rs_update
          for database NY_DS.rdb1
  ```

```
        output language
        'update authors set
                au_lname = ?au_lname!param?,
                au_fname = ?au_fname!param?,
                phone = ?phone!param?,
                address = ?address!param?,
                city = ?city!param?,
                state = ?state!param?,
                zip = ?zip!param?,
                contract = ?contract!param?
```

用法

- **alter function string** 与 **create function string** 相同，只是前者先要执行 **drop function string**。函数字符串将在单个事务中删除并重新创建，以防止发生因丢失函数字符串而导致的错误。
- 在函数字符串类的主节点更改具有类范围的函数的函数字符串。有关函数字符串类的主节点的详细信息，请参见 **create function string class**。
- 在创建复制定义的节点上更改具有复制定义范围的函数（包括用户定义函数）的函数字符串。每个复制定义都有其自己的函数字符串集。
- 在控制目标数据库（备用数据库或复制数据库）的 Replication Server 中为目标作用域函数字符串执行 **alter function string**。
- 对于 **rs_select**、**rs_select_with_lock**、**rs_datarow_for_writetext**、**rs_get_textptr**、**rs_textptr_init** 和 **rs_writetext** 函数字符串，Replication Server 使用 *function_string* 名称来确定要更改的字符串。如果创建函数字符串时提供了 *function_string* 名称，则必须用 **alter function string** 指定此名称，以便找到要更改的函数字符串。
- 有关要与 **alter function sting** 一起使用的关键字和选项的各种参数说明的详细信息，请参见 **create function string**。
- 要恢复函数的缺省函数字符串，应省略 **output** 子句。

权限

alter function string 需要“create object”权限。

另请参见

- alter connection （第 109 页）
- create connection （第 214 页）
- create function （第 231 页）
- create function string （第 236 页）
- create function string class （第 249 页）
- define subscription （第 290 页）
- drop function string （第 302 页）

alter function string class

更改函数字符串类，指定此类应是基类还是派生类。

语法

```
alter function string class function_class
   set parent to {parent_class | null}
```

参数

- **function_class** – 要更改的现有函数字符串类的名称。
- **set parent to** – 将现有的类指定为要更改的类的父类；或者用 **null** 关键字指定此类应当为基类。
- **parent_class** – 指定为新派生类的父类的现有函数字符串类的名称。*rs_sqlserver_function_class* 不可以用作父类。
- **null** – 指定此类应当为基类。

示例

- **示例 1** – 指定 *sqlserver2_function_class* 应当成为派生类，并从父类 *rs_default_function_class* 继承函数字符串：

```
alter function string class
 sqlserver2_function_class
 set parent to rs_default_function_class
```

- **示例 2** – 指定名为 *rpc_xact* 的派生函数字符串类应当为基类：

```
alter function string class rpc_xact
 set parent to null
```

用法

- 使用 **alter function string class** 将派生函数字符串类更改为基类、更改派生类的父类或者将基类更改为派生类。
- 派生类的主节点与其父类的主节点相同。在父类的主节点上更改派生类。但是，如果父类是系统提供的类 *rs_default_function_class* 或 *rs_db2_function_class*，则派生类的主节点就是创建此派生类的 Replication Server。
- 有关 **alter function string class** 的详细信息，请参见 **create function string**。
- 有关函数字符串类、函数字符串和函数的详细信息，请参见《Replication Server 管理指南第二卷》。
- Replication Server 通过复制系统将更改的函数字符串类分发到合格的节点。通常由于复制系统会有一定时间的滞后，因此更改不会立即显示在所有这些节点上。

权限

alter function string class 需要“sa”权限。

另请参见

- alter connection （第 109 页）
- create connection （第 214 页）
- create function （第 231 页）
- create function string （第 236 页）
- create function string class （第 249 页）
- drop function string class （第 304 页）

alter logical connection

启用或禁用逻辑连接的分配器线程，更改逻辑连接的属性，以及允许或禁止将 **truncate table** 复制到备用数据库。

语法

```
alter logical connection
   to logical_ds.logical_db {
   set distribution {on | off} |
   set logical_database_param to 'value'}
```

参数

- **logical_ds** – 逻辑连接的数据服务器名称。
- **logical_db** – 逻辑连接的数据库名称。
- **distribution on** – 启用逻辑连接的分配器线程。
- **distribution off** – 禁用逻辑连接的分配器线程。
- **logical_database_param** – 影响逻辑连接的配置参数的名称。表 19. 影响逻辑连接的配置参数介绍了可使用 **alter logical connection** 设置的参数。
- **value** – 与此参数相匹配的配置参数的设置。*value* 为字符串。

表 19. 影响逻辑连接的配置参数

logical_database_param	value
dist_stop_unsupported_cmd	可以使用 **dist_stop_unsupported_cmd** 设置在遇到下游 Replication Server 不支持的命令时 DIST 是自我挂起还是继续运行。如果 **dist_stop_unsupported_cmd** 为 on，并且下游 Replication Server 不支持某个命令，DIST 将会自我挂起。如果为 off，DIST 将忽略不支持的命令。 无论 **dist_stop_unsupported_cmd** 设置是什么，Replication Server 在看到较高版本的命令的第一个实例而该实例无法发送到较低版本的 Replication Server 时，总是会记录错误消息。 缺省值：off
materialization_save_interval	实现队列保存间隔。此参数仅用于热备份应用程序中的备用数据库。 缺省值：“strict”（对于备用数据库）
replicate_minimal_columns	指定 Replication Server 是发送所有事务的所有复制定义列，还是仅发送在备用数据库中执行更新或删除操作所需要的复制定义列。值为“on”和“off”。 仅在复制定义不包含带有任何参数的 **send standby** 选项或根本没有复制定义时，Replication Server 才在备用情况下使用此值。 否则，Replication Server 将使用复制定义中的“replicate minimal columns”或“replicate all columns”参数的值。 缺省值：on 在将 **dsi_compile_enable** 设置为“on”时，Replication Server 会忽略您为 **replicate_minimal_columns** 设置的内容。
save_interval	消息成功传递到目标数据服务器之后，Replication Server 对其进行保存的分钟数。有关详细信息，请参见《Replication Server 管理指南第二卷》。 缺省值：0 分钟
send_standby_repdef_cols	指定 Replication Server 应将哪些列发送到备用数据库以建立逻辑连接。替换复制定义中的“send standby”选项，此选项用于通知 Replication Server 要将哪些表列发送到备用数据库。这些值为： • **on** – 只发送匹配复制定义中出现的表列。忽略复制定义中的“send standby”选项。 • **off** – 将所有表列发送到备份数据库。忽略复制定义中的“send standby”选项。 • **check_repdef** – 根据“send standby”选项，将所有表列发送到备用数据库。 缺省值：**check_repdef**

logical_database_param	value
send_truncate_table	指定是允许还是禁止将 **truncate table** 复制到备用数据库。这些值为： • **on** – 允许将 **truncate table** 复制到备用数据库。这是缺省值。 • **off** – 禁止将 **truncate table** 复制到备用数据库。
ws_sqldml_replication	指定是否将 SQL 语句复制到热备份数据服务器中。这些值为： • **on** – 复制 SQL 语句。复制的缺省语句为 **update**、**delete**、 **insert select** 和 **select into**。 • **off** – 忽略所有 SQL 语句。 **注意：** **ws_sqldml_replication** 的优先级比 SQL 复制的表复制定义低。如果表的表复制定义包含 **send standby** 子句，该子句将确定是否复制 DML 语句（**select into** 除外），而无论 **ws_sqldml_replication** 参数设置是什么。

示例

- **示例 1** – 禁用 LDS.*pubs2* 逻辑连接的分配器线程：

```
alter logical connection to LDS.pubs2
 set distribution off
```

- **示例 2** – 将 LDS.*pubs2* 逻辑连接的保存间隔更改为“0”，以便删除该逻辑连接的 DSI 队列中的消息：

```
alter logical connection to LDS.pubs2
 set save_interval to '0'
```

- **示例 3** – 允许将 **truncate table** 复制到备用数据库：

```
alter logical connection to LDS.pubs2
 set send_truncate_table to 'on'
```

用法

- 要将 truncate table 复制到热备份数据库，请将 **send_truncate_table** 选项设置为“on”。
- 仅当 Adaptive Server 11.5 版或更高版本中同时包含活动数据库和热备份数据库时，才将 **send_truncate_table** 选项设置为“on”。
- 如果指定了 **send_truncate_table to on** 子句，Replication Server 会将所有标记为复制的表的 **truncate table** 执行复制到热备份数据库。
- 设置热备份应用程序之后，使用 **alter logical connection** 命令禁用分配器线程。将数据库添加到复制系统时，Replication Server 创建分配器线程来处理数据预订。
- 使用 **set distribution off** 子句可禁用逻辑连接的分配器线程。如果设置了数据库的热备份但数据库中无数据预订，并且数据库不是复制存储过程执行源，则应使用

此选项。此类逻辑数据库可以是不参与正常复制的热备份应用程序，也可以是逻辑复制数据库。

- 使用 **set distribution off** 禁用逻辑的分配器线程后，可以使用 **set distribution on** 启动此线程。这样做可为逻辑数据库中的数据创建复制定义和预订，也可启动逻辑数据库中的复制存储过程。
- 可以使用 **suspend distributor** 和 **resume distributor** 命令挂起或恢复物理或逻辑数据库连接的分配器线程。
- 有关设置和管理热备份应用的详细信息，请参见《Replication Server 管理指南第一卷和第二卷》。
- 可以使用 **configure replication server** 命令设置影响源于当前 Replication Server 的所有逻辑连接的参数。
- 如果创建了逻辑连接，缺省情况下，将该逻辑连接的 **save_interval** 参数设置为“**strict**”。这可确保消息在应用到备用数据库之前，不会从 DSI 队列中删除。

  如果备用数据库长时间不可用，则 Replication Server 的队列可能已满。要避免出现这种情况，请将 **save_interval** 从“**strict**”更改为“0”（分钟）。该设置允许 Replication Server 删除队列。

  警告！ **save_interval** 参数仅影响 DSI 队列。**materialization_save_interval** 参数仅影响当前现有的实现队列。只有 在因缺少稳定队列空间而导致出现严重情况时，才应重新设置这些参数。重置此参数（从“**strict**”设置为给定的分钟数）可能导致备用数据库的消息丢失。Replication Server 无法检测到这种类型的丢失；您必须自己检验备用数据库的完整性。
- 如果创建了逻辑连接，缺省情况下，将该逻辑连接的 **materialization_save_interval** 参数设置为“**strict**”。这可确保消息在应用到备用数据库之前，不会从实现队列中删除。

  如果备用数据库长时间不可用，则 Replication Server 的队列可能已满。要避免出现这种情况，请将 **materialization_save_interval** 从“**strict**”更改为“0”（分钟）。该设置允许 Replication Server 删除队列。

另请参见

- admin logical_status （第 54 页）
- configure replication server （第 181 页）
- create logical connection （第 252 页）
- resume distributor （第 323 页）
- suspend distributor （第 335 页）

alter partition

更改分区的大小。

语法

```
alter partition logical_name [expand [size =size]]
```

参数

- **logical_name** – 分区的名称。名称必须符合标识符规则。**drop partition** 和 **create partition** 命令中也会使用此名称。
- **expand** – 指定分区将要增加大小。
- **size** – 指定分区要增加的大小（以 MB 为单位）。缺省值为 2MB。

示例

- **示例 1** – 本示例将逻辑分区 *P1* 的大小增加了 50MB：

```
alter partition P1 expand size = 50
```

- **示例 2** – 本示例将逻辑分区 *P2* 的大小增加了 2MB：

```
alter partition P2
```

用法

- 用户可以使用 **alter partition** 扩大当前使用的分区。当 Replication Server 需要更多磁盘空间并且现有分区的相同磁盘中仍有可用空间时，此功能非常有用。
- 出现物理磁盘空间不足时，将中止 **alter partition** 并显示错误消息。为分区分配的空间与应用此命令之前的空间相同。
- 可为分区分配的最大大小为 1TB,，大约为 1,000,000MB。

权限

只有“sa”用户能够执行 **alter partition**。

另请参见

- admin disk_space （第 50 页）
- create partition （第 253 页）
- drop partition （第 306 页）

alter queue

指定稳定队列在遇到超过 16K 个字节的大消息时的行为。仅当 Replication Server 版本为 12.5 或更高版本且 Replication Server 节点版本为 12.1 或更低版本时才适用。

语法

```
alter queue, q_number, q_type,
      set sqm_xact_with_large_msg [to]     {skip | shutdown}
     set sqm_cache_enable to “on | off”
      set sqm_page_size to “numblocks"
      set sqm_cache_size to "numpages"
```

参数

- **q_number** – 稳定队列的队列号。
- **q_type** – 稳定队列的队列类型。值为“0”表示出站队列；值为“1”表示入站队列。
- **sqm_xact_with_large_msg {skip | shutdown}** – 指定在遇到超过 16K 个字节的消息时，SQM 是应当跳过该消息还是关闭。
- **sqm_cache_enable to “on" | "off”** – 允许或禁止稳定队列高速缓存。队列级别高速缓存会覆盖使用 **configure replication server** 设置的服务器级别高速缓存。**sqm_cache_enable** 的缺省值为“on”。
- **sqm_page_size** – 设置稳定队列的页大小。如果设置队列级别的页大小，将覆盖使用 **configure replication server** 设置的服务器级别页大小。**sqm_page_size** 的缺省值为 4。
- **"numblocks"** – 指定一个页面中包含的大小为 16K 的块数。配置页大小也会设置 Replication Server 的 I/O 大小。例如，将页大小设置为 4 会指示 Replication Server 以 64K 块的形式向稳定队列中写入数据。*numblocks* 接受从 1 到 64 的值。
- **sqm_cache_size** – 设置稳定队列的高速缓存大小。如果设置队列级别的高速缓存大小，将覆盖使用 **configure replication server** 设置的服务器级别高速缓存大小。**sqm_cache_size** 的缺省值为 16。
- **"numpages"** – 指定高速缓存中的页数。范围是 1 到 512 页。

示例

- **示例 1** – 如果向队列中传递了一条大消息，则会关闭编号为 2 的队列：

```
alter queue, 2, 0, set sqm_xact_with_large_msg to
    shutdown
```

用法

- 如果对 **sqm_cache_enable**、**sqm_page_size** 和 **sqm_cache_size** 参数进行更改，则需要重新启动服务器，更改才能生效。
- 如果节点版本为 12.5 或更高版本，**alter queue** 将会失败。

权限

alter queue 需要“sa”权限。

另请参见

- alter route （第 161 页）
- resume queue （第 325 页）
- resume route （第 326 页）

alter replication definition

更改现有的复制定义。

语法

```
alter replication definition replication_definition
{with replicate table named [table_owner.]'table_name' |
add column_name [as replicate_column_name]
       [datatype [null | not null]]
       [map to published_datatype] [quoted],... |
alter columns with column_name
       [as replicate_column_name] [quoted | not quoted],...|
alter columns with column_name
        datatype [null | not null]
        [map to published_datatype],...|
        references {[table_owner.]table_name [(column_name) | null}]
alter columns column_name {quoted | not quoted}
add primary key column_name [, column_name]... |
drop primary key column_name [, column_name]... |
add searchable columns column_name [, column_name]... |
drop searchable columns column_name [, column_name]... |
drop column_name[, column_name] ... |
send standby [off | {all | replication definition} columns] |
replicate {minimal | all} columns |
replicate {SQLDML [ 'off' ] | 'options' } |
replicate_if_changed column_name [, column_name]...|
always_replicate column_name [, column_name]... |
{with | without} dynamic sql |
alter replicate table name {quoted | not quoted}}
[with DSI_suspended]
```

参数

- **replication_definition** – 要更改的复制定义的名称。
- **with replicate table named** – 指定复制数据库的表的名称。*table_name* 是一个字符串，最多可包含 200 个字符。*table_owner* 是表名的可选限定符，表示表的所有者。如果实际表的所有者与复制定义中指定的所有者不对应，数据服务器操作可能会失败。
- **add columns column_name** – 指定复制定义的其它列及其数据类型。*column_name* 是要添加到复制列的列表中的列名。复制定义的列名必须是唯一的。

  还有 **add columns** *declared_column_name*。请参见“使用列级数据类型转换”。
- **as replicate_column_name** – 对于要添加到复制定义的列，指定要将主列数据复制到其中的复制表中的列名。*replicate_column_name* 是复制表中的列名，与主表中的指定列相对应。如果复制列和主列的名称不同，请使用此子句。
- **数据类型** – 要添加到复制定义列的列表中的列的数据类型或要更改的现有列的数据类型。有关支持的数据类型及其语法的列表，请参见“数据类型”。

如果列已在主表的现有复制定义中列出，则同一主表的后续复制定义必须指定相同的数据类型。

如果要为此列指定列级别的数据类型转换，则用作 *declared_datatype*。声明的数据类型必须是 Replication Server 的本机数据类型，或者是主数据类型的数据类型定义。

- **null 或 not null** – 仅应用于 *text*、*unitext*、*image* 和 *rawobject* 列。指定复制表中是否允许空值。缺省值为 **not null**，表示复制表不接受空值。

  对于同一主表的所有复制定义，每个 *text*、*unitext*、*image* 和 *rawobject* 列的空值状态必须相匹配，并且必须与实际表中的设置相匹配。如果同一主表的现有复制定义包含 *text*、*unitext*、*image* 或 *rawobject* 列，则是否指定空值状态是可选的。
- **quoted | not quoted** – 指定表或列名是否为带引号的标识符。对于复制需要具有引号的每个对象使用 quoted 子句。
- **alter columns column_name** – 在复制定义中指定要更改的列及其数据类型。*column_name* 是要更改的列的名称。复制定义的列名必须是唯一的。

  指定列级数据类型转换时，使用 **alter columns** *declared_column_name*。
- **map to published_datatype** – 指定列级别数据类型转换后列的数据类型。*published_datatype* 必须是 Replication Server 本机数据类型或发布的数据类型的数据类型定义。
- **references table owner.table name column name** – 指定在主数据库中具有参照约束并且您要添加或更改为引用表的表的表名称。使用 **null** 选项可删除引用。*table_name* 是最多 200 个字符的字符串。*table_owner* 是可选的，表示表所有者。*column name* 是可选的。如果表的实际所有者与复制定义中指定的所有者不相符，数据服务器操作可能会失败。有关用法的详细信息，请参见“create replication definition”中的“处理具有参照约束的表”。
- **add/drop primary key** – 用于在主键列的列表中添加列，或从此列表中删除列。Replication Server 依赖于主键在复制表或备份表上查找正确的行。要删除所有主键列，请首先更改相应的复制定义来添加新的主键，然后删除表中旧的主键列。如果所有主键均丢失，DSI 将会关闭。有关主键的其它信息，请参见 **create replication definition**。
- **add searchable columns column_name** – 指定可在 **create subscription** 或 **define subscription** 命令的 **where** 子句中使用的其它列。*column_name* 是要添加到可搜索列的列表中的列的名称。在每个子句中，同一列名只能出现一次。

  不能将 *text*、*unitext*、*image*、*rawobject*、*rawobject in row* 或加密列指定为可搜索列。
- **drop searchable columns column_name** – 指定要从可搜索列的列表中删除的列。只有列未在预订或项目 **where** 子句中使用时，才可以从可搜索列的列表中将其删除。
- **drop column_name** – 指定要删除的列。

- **send standby** – 指定在将数据复制到热备份应用程序的备用数据库的过程中如何使用复制定义。有关使用此子句及其选项的详细信息，请参见“复制到备用数据库”。
- **replicate minimal columns** – 只将需要在复制数据库中执行更新或删除操作的列发送到复制 Replication Server。要复制所有列，应使用 **replicate all columns**。
- **replicate SQLDML [‘off’]** – 打开或关闭指定的 DML 选项的 SQL 语句复制。
- **replicate ‘options’** – 复制以下 DML 操作的任意组合：
  - U – **update**
  - D – **delete**
  - I – **insert select**
- **replicate_if_changed** – 指定要添加到 **replicate_if_changed** 列的列表中的 *text*、*unitext*、*image* 或 *rawobject* 列。如果同一主表具有多个复制定义，则当使用此子句更改一个复制定义时，将会更改同一主表的所有复制定义。
- **always_replicate** – 指定要添加到 **always_replicate** 列的列表中的 *text*、*image* 或 *rawobject* 列。如果同一主表具有多个复制定义，则当使用此子句更改一个复制定义时，将会更改同一主表的所有复制定义。
- **with dynamic sql** – 指定让 DSI 在命令合格并且有足够的高速缓存空间可用时将动态 SQL 应用到表。这是缺省值。

  有关命令必须满足哪些条件才能应用动态 SQL 的信息，请参见《Replication Server 管理指南第二卷》。
- **without dynamic sql** – 指定 DSI 不得使用动态 SQL 命令。
- **with DSI_suspended** – 允许您挂起备份 DSI（如果有）和每个预订复制 DSI 线程。在 Replication Server 将旧复制定义版本的所有数据应用于备用数据库或复制数据库后，Replication Server 将会在备用数据库或复制数据库中挂起 DSI 线程。

  在 Replication Server 挂起 DSI 线程后，您可以对目标模式以及任何自定义函数字符串进行更改。当您恢复该 DSI 线程时，Replication Server 将使用变更的复制定义复制主更新。

  在以下情况下，您不需要使用 **with DSI_suspended**：
  - 没有对复制定义的预订。
  - 不需要更改自定义函数字符串。
  - 不需要更改复制数据库模式或备用数据库模式。

  注意： 如果有来自节点版本早于 1550 的复制 Replication Server 的预订，则不会挂起该 Replication Server 的复制 DSI 线程。

示例

- **示例 1** – 将 *state* 作为可搜索的列添加到 *authors_rep* 复制定义中：

```
alter replication definition authors_rep
 add searchable columns state
```

- **示例 2** – 更改 *titles_rep* 复制定义，以指定仅发送最少数量的列来执行删除和更新操作：

```
alter replication definition titles_rep
 replicate minimal columns
```

- **示例 3** – 更改 *titles_rep* 复制定义，指定可由用户“joe”拥有的名为 *copy_titles* 的复制表预订的复制定义：

```
alter replication definition titles_rep
 with replicate table named joe.'copy_titles'
```

- **示例 4** – 更改 *pubs_rep* 复制定义，以指定主列 *pub_name* 将复制到复制列 *pub_name_set*：

```
alter replication definition pubs_rep
alter columns with pub_name as pub_name_set
```

- **示例 5** – 引入列级别转换，此转换使 *hire_date* 列的值从 *rs_db2_date*（主）格式转换为本机数据类型 *smalldatetime*（复制）格式：

```
alter replication definition employee_repdef
alter columns with hire_date as rs_db2_date
map to smalldatetime
```

- **示例 6** – 在发送到复制节点时，将名为 *foo* 的表用引号括起：

```
alter replication definition repdef
   alter replicate table name foo quoted
```

- **示例 7** – 从 *foo_col2* 列中删除带引号的标识符标记。

```
alter replication definition repdef
      alter columns “foo_col2” not quoted
```

- **示例 8** – 指示 Replication Server 将复制定义列名更改为 `pub_name_set`，使用旧列名 `pub_name` 处理队列中的当前内容，然后在 Replication Server 处理完队列中的数据后挂起目标 DSI。一旦恢复 DSI 后，Replication Server 将对目标数据库使用更改的复制定义：

```
alter replication definition pubs_rep
alter columns with pub_name as pub_name_set
with DSI_suspended
```

- **示例 9** – 从“authors”复制定义中删除 *address*、*city*、*state* 和 *zip* 列：

```
alter replication definition authors
drop address, city, state, zip
```

用法

- 使用 **alter replication definition** 命令可更改复制定义，方法如下：
  - 添加或删除主键
  - 更改目标复制表的名称
  - 更改目标复制列的名称

  - 添加列并指示对应的目标复制列的名称
  - 添加或删除可搜索列
  - 更改热备份应用程序使用复制定义的方式
  - 更改列数据类型
  - 在复制所有列或复制最少列之间切换
  - 更改 *text*、*unitext*、*image* 或 *rawobject* 列的复制状态
  - 引入或删除列级别数据类型转换
  - 在 DSI 的动态 SQL 应用中包括或排除表
- 在复制定义的主节点执行 **alter replication definition**。
- 若要在不使用表级别复制定义的情况下使用数据库复制定义复制加密列，必须使用 INIT_VECTOR NULL 和 PAD NULL 为加密列定义加密密钥。
- 在主 Replication Server 版本高于复制 Replication Server 版本的混合版本环境中，如果复制 Replication Server 无法支持修改，则不能更改复制 Replication Server 支持和预订的复制定义。但是，如果复制 Replication Server 支持预订复制定义但并没有这样做，则会修改复制定义并将其从复制 Replication Server 中删除。
- 有关复制 SQL 语句的详细信息，请参见“复制 SQL 语句”。
- 有关 **alter replication definition** 命令中选项的详细信息，请参见 create replication definition 命令。

添加列

- 如果要添加列，应通过对复制定义进行分发来协调 **alter replication definition**。为了避免错误，请按照“更改复制定义的过程”中的步骤操作。
- 如果要添加到复制定义中的列包含 *identity* 列，维护用户必须是复制数据库中的表所有者（或必须是“dbo”，或别名为“dbo”）才能使用 Transact-SQL **identity_insert** 选项。主表只能包含一个 *identity* 列。
- 如果要添加到复制定义的列包含 *timestamp* 列，则维护用户必须是复制数据库的表的所有者（或必须是“dbo”，或别名为“dbo”）。主表只能包含一个 *timestamp* 列。

删除列

- 如果有来自节点版本早于 1550 的复制 Replication Server 的预订，则主 Replication Server 会拒绝请求删除列的 alter replication definition。

  注意： 如果您更改复制定义以删除某个列，可能需要在节点版本早于 1550 的复制 Replication Server 上重置自动更正或动态 SQL 设置。
- 如果主表有多个复制定义，**alter replication definition** 只从您通过命令行在 *repdef_name* 中指定的复制定义中删除列。
- **drop** 参数从表复制定义中删除一列或多列。如果某列是主键或可搜索列的一部分，**drop** 将从主键列表或可搜索列的列表中删除该列。如果某列符合以下条件，Replication Server 将拒绝删除该列的 alter replication definition 请求：
  - 唯一的列

- 复制定义的唯一主键列
- 位于预订或项目的 **where** 子句中
- 位于在项目或预订的 **where** 子句中指定的可搜索列之前。

更改列数据类型

- 如果预订或项目 **where** 子句中正在使用某个列，则无法更改此列的数据类型。
- 不能更改 *rs_address* 数据类型。
- 只有在列不是主键列或可搜索列的情况下，才可以将列数据类型更改为 *text*、*unitext*、*image*、*rawobject* 或 *rawobject in row*。
- 要更改已发布的列数据类型，必须指定已声明的数据类型以及 **map to** 选项。
- 如果主表有多个复制定义，则列的声明数据类型和可为空性应该在表的所有复制定义间保持一致。
- 请参见《Replication Server 管理指南第一卷》，其中介绍了如何更改数据类型。
- 只有对于 *text*、*unitext*、*image* 和 *rawobject* 列，才能在 **null** 和 **not null** 之间切换。

使用列级数据类型转换

- 要实现列级别数据类型转换，必须先按照适用于您的平台的《Replication Server 配置指南》中的说明设置和安装异构数据类型支持 (HDS) 对象。
- 不能将 *text*、*unitext*、*image* 或 *rawobject* 数据类型作为基本数据类型或数据类型定义使用，也不能将其作为列级别转换或类级别转换的源或目标使用。
- *declared_datatype* 取决于传递给 Replication Server 的值的数据类型：
  - 如果 Replication Agent 传递的是基本 Replication Server 数据类型，则 *declared_datatype* 为基本 Replication Server 数据类型。
  - 如果 Replication Agent 传递的是任何其它数据类型，则 *declared_datatype* 必须是主数据库中原始数据类型的数据类型定义。
- *published_datatype* 是在列级别转换后，但尚未进行任何类级别转换前值的数据类型。*published_datatype* 必须是 Replication Server 本机数据类型，或其它数据库中数据类型的数据类型定义。
- 在多个复制定义中声明的列所使用的 *declared_datatype* 在每个复制定义中必须相同。*published_datatype* 可以不同。

复制全部列或最少列

- 如果将 **replicate minimal column** 选项用于复制定义，只有删除或更新操作所需的最少数量列的数据将发送到复制 Replication Server。要复制所有的列，需指定 **replicate all columns**。有关此功能的其它信息，请参见 **create replication definition**。

  注意： 如果复制定义具有 **replicate all columns** 并且 **replicate minimal columns** 连接属性设置为“on”，连接将复制最少列。如果想要将所有列复制到目标数据库，则将 DSI 连接的 **replicate minimal columns** 值设置为“off”。

复制到备用数据库

- Replication Server 不要求使用复制定义来维护热备份应用程序中的备用数据库。使用复制定义可以提高复制到备用数据库操作的性能。可以只为实现此目的而为逻辑数据库中的每个表创建复制定义。
- 通过带有除 **off** 之外任何选项的 **send standby**，可以使用此复制定义将此表的事务复制到备用数据库。复制定义的主键列和 **replicate minimal columns** 设置用于将数据复制到备用数据库。此方法的选项包括：
  - 使用 **send standby** 或 **send standby all columns**，可将所有主表列复制到备用数据库。
  - 使用 **send standby replication definition columns**，可仅将复制定义的列复制到备用数据库。
- 使用 **send standby off**，可指出在复制到备用数据库时，不应使用此表的单个复制定义。此表中的所有列均复制到备用数据库，在复制到备用数据库时，将组合使用此表所有复制定义中的全部主键列。逻辑连接的 **replicate_minimal_columns** 设置确定是发送最少数量的列还是所有列以进行更新和删除。请参见 **alter logical connection**。

  如果表没有复制定义，则此表中的所有列都将复制到备用数据库，并且 Replication Server 将构建主键。在这种情况下，将启用 **replicate_minimal_columns**。

处理具有参照约束的表

您可以使用复制定义来指定具有参照约束（例如外键及其它检查约束）的表，以便当您启用 RTL 或 HVAR 时，Replication Server 知道这些表。请参见《Replication Server 管理指南第二卷》的“性能调优”的“高级服务选件”中的“High Volume Adaptive Replication to Adaptive Server”（向 Adaptive Server 进行高容量自适应复制）和《Replication Server 异构复制指南》的“Sybase IQ as Replication Data Server”（Sybase IQ 作为复制数据服务器）中的“Sybase IQ Replicate Database Configuration”（Sybase IQ 复制数据库配置）。

更改复制定义的过程

当您更改复制定义时，Replication Server 会自动协调复制定义更改和数据复制的传播。您可以直接在主 Replication Server 上请求复制定义更改，也可以在对数据库模式进行更改的同时在主数据库中使用 **alter replication definition**、**alter applied replication definition** 或 **alter request function replication definition** 命令请求复制定义更改。

如果主数据库日志不包含所更改复制定义的数据，您可以直接在主 Replication Server 上发出复制定义请求。除此以外，使用 **rs_send_repserver_cmd** 存储过程在主数据库中发出复制定义请求始终是安全的。

如果数据库不支持 **rs_send_repserver_cmd**，您需要等待，直到主数据库日志不包含您所更改的模式的任何数据行，然后在主 Replication Server 上执行 **alter replication definition** 请求。

请参见《Replication Server 管理指南第一卷》的“管理复制的表”中的“复制定义更改请求过程”。

权限

alter replication definition 需要“create object”权限。

另请参见

- admin verify_repserver_cmd （第 84 页）
- alter function string （第 143 页）
- create replication definition （第 258 页）
- drop replication definition （第 308 页）
- rs_set_quoted_identifier （第 430 页）
- rs_send_repserver_cmd （第 546 页）
- rs_helprepversion （第 540 页）

alter request function replication definition

更改使用 **create request function replication definition** 命令创建的函数复制定义。

语法

```
alter request function replication definition repdef_name
        {with replicate function named 'func_name' |
        add @param_name datatype[, @param_name datatype]… |
        add searchable parameters @param_name[, @param_name]… |
        send standby {all | replication definition} parameters}
         [with DSI_suspended]
```

参数

- **repdef_name** – 要更改的请求函数复制定义的名称。
- **with replicate function named 'func_name'** – 指定要在复制数据库中执行的存储过程的名称。此复制定义的复制函数名称必须与其主函数名称不同。*func_name* 是最多包含 255 个字符的字符串。
- **add** – 指定函数复制定义的其它参数及其数据类型。
- **@param_name** – 要添加到复制参数或可搜索参数的列表中的参数的名称。每个参数名必须以 @ 字符开头。
- **数据类型** – 要添加到参数列表的参数的数据类型。Adaptive Server 存储过程和函数复制定义不能包含数据类型为 *text*、*unitext*、*rawobject* 和 *image* 的参数。
- **add searchable parameters** – 指定可在 **create subscription** 或 **define subscription** 命令的 **where** 子句中使用的其它参数。
- **send standby** – 在热备份应用程序中，指定是将此函数中的所有参数都发送到备用数据库 (**send standby all parameters**)，还是只将复制定义中指定的参数发送到备用数据库 (**send standby replication definition parameters**)。缺省值是 **send standby all parameters**。

- **with DSI_suspended** – 允许您挂起备份 DSI（如果有）和每个预订复制 DSI 线程。在 Replication Server 将旧复制定义版本的所有数据应用于备用数据库或复制数据库后，Replication Server 将会在备用数据库或复制数据库中挂起 DSI 线程。

  在 Replication Server 挂起 DSI 线程后，您可以对目标存储过程以及任何自定义函数字符串进行更改。当您恢复该 DSI 线程时，Replication Server 将使用变更的复制定义复制主更新。

  在以下情况下，您不需要使用 **with DSI_suspended**：

  - 没有对复制定义的预订。
  - 不需要更改自定义函数字符串。
  - 不需要更改复制数据库存储过程或备用数据库存储过程。

  注意： 如果有来自节点版本早于 1550 的复制 Replication Server 的预订，则不会挂起该 Replication Server 的复制 DSI 线程。

示例

- **示例 1** – 将 *@notes*、*@pubdate* 和 *@contract* 参数添加到 **titles_frep** 函数复制定义：

```
alter request function replication definition
    titles_frep
add @notes varchar(200), @pubdate datetime,
    @contract bit
```

- **示例 2** – 将 *@type* 和 *@pubdate* 参数添加到 **titles_frep** 函数复制定义的可搜索参数列表中：

```
alter request function replication definition
    titles_frep
add searchable parameters @type, @pubdate
```

- **示例 3** – 在复制数据库中将要复制的 **titles_frep** 函数复制定义更改为 **newtitles** 存储过程并指示 Replication Server 在将执行 **alter request replication definition** 之前已存在的主数据复制到复制数据库之后挂起目标 DSI：

```
alter request function replication definition titles_frep
with replicate function named 'newtitles'
with DSI suspended
```

用法

- 使用 **alter request function replication definition** 可更改现有的请求函数复制定义。您可以添加复制参数和可搜索参数，选择要发送到热备份的参数，并为要在复制数据库中执行的存储过程指定不同的名称。
- **alter request function replication definition** 只能更改使用 **create request function replication definition** 命令创建的复制定义。

- 更改函数复制定义时，为函数复制定义指定的名称、参数和数据类型必须与您要复制的存储过程匹配。只有在函数复制定义中指定的参数才会得到复制。
- 同一存储过程的多个函数复制定义必须具有相同的参数列表。如果添加新参数，则会自动将新参数添加到为该存储过程创建的所有函数复制定义。
- 必须在创建函数复制定义的 Replication Server 上执行 **alter request function replication definition** 命令。
- 在任何子句中，一个参数名只能出现一次。
- 在添加参数时，必须指示 Replication Server 将 **alter request function replication definition** 与函数复制定义的分发进行协调。此外，您必须指示 Replication Server 协调对存储过程和复制定义进行的更改。

  要变更复制定义，请参见《Replication Server 管理指南第一卷》的“管理复制的表”中的“复制定义更改请求过程”。
- 使用 **with replicate function named** 子句可以为要在复制数据库中执行的存储过程指定名称。请参见 **create request function replication definition**。

有关更改请求函数复制定义的详细信息，请参见《Replication Server 管理指南第一卷》。

权限

alter request function replication definition 需要“create object”权限。

另请参见

- alter function string （第 143 页）
- alter applied function replication definition （第 107 页）
- create applied function replication definition （第 206 页）
- create request function replication definition （第 269 页）
- drop function replication definition （第 301 页）
- rs_send_repserver_cmd （第 546 页）
- rs_helprepversion （第 540 页）

alter route

更改从当前 Replication Server 到远程 Replication Server 的路由的属性。

语法

```
alter route to dest_replication_server {
    set next site [to] thru_replication_server |
    set username [to] 'user' set password [to] 'passwd' |
    set password [to] 'passwd' |
    set route_param [to] 'value' |
    set security_param [to] 'value' |
    set security_services [to] 'default'}
```

参数

- **dest_replication_server** – 要更改其路由的目标 Replication Server 的名称。
- **thru_replication_server** – 中间 Replication Server 的名称，通过它传递目标 Replication Server 的消息。
- **user** – 路由要使用的登录名。
- **passwd** – 与登录名一起使用的口令。
- **route_param** – 影响路由的参数。有关参数和值的列表，请参见表 20. 影响路由的配置参数。
- **value** – *route_param* 的设置。它是一个字符串。

表 20. 影响路由的配置参数

route_param	值
disk_affinity	指定用于分配下一个分区的分配提示。输入一个分区的逻辑名称，在当前分区已满时应该将下一个段分配给此分区。 缺省值：off
rsi_batch_size	请求截断点之前发送到另一 Replication Server 的字节数。 缺省值：256KB 最小值：1KB 最大值：128MB
rsi_fadeout_time	Replication Server 关闭与目标 Replication Server 的连接之前的空闲秒数。 缺省值：-1（指定 Replication Server 不关闭连接）
rsi_packet_size	用于与其它 Replication Server 通信的包的大小（以字节为单位）。范围是 1024 到 16384 个字节。 缺省值：4096 字节
rsi_sync_interval	RSI 同步查询消息之间的秒数。Replication Server 使用这些消息使 RSI 出站队列与目标 Replication Server 同步。值必须大于 0。 缺省值：60 秒
rsi_xact_with_large_msg	指定遇到大消息时的路由行为。此参数仅适用于复制节点上节点版本为 12.1 或更低版本的直接路由。值为“skip”和“shutdown”。 缺省值：shutdown
save_interval	消息成功传递到目标数据服务器之后，Replication Server 对其进行保存的分钟数。有关详细信息，请参见《Replication Server 管理指南第二卷》。 缺省值：0 分钟

- **security_param** – 指定安全性参数的名称。有关可以使用 **alter route** 设置的安全性参数的列表和说明，请参见表 20. 影响路由的配置参数。

- **set security_services [to] 'default'** – 重置所有基于网络的连接安全性功能，使其与 Replication Server 的全局设置相匹配。

示例

- **示例 1** – 在示例 1 和 2 中，存在从东京 Replication Server (TOKYO_RS) 到旧金山 Replication Server (SF_RS) 以及到悉尼 Replication Server (SYDNEY_RS) 的直接路由。以下命令将一个直接路由更改为间接路由，以便 TOKYO_RS 可通过 SF_RS 将消息传递到 SYDNEY_RS。

  此命令是在 SF_RS 上输入的，它创建一个到 SYDNEY_RS 的直接路由，新的间接路由将使用该路由：

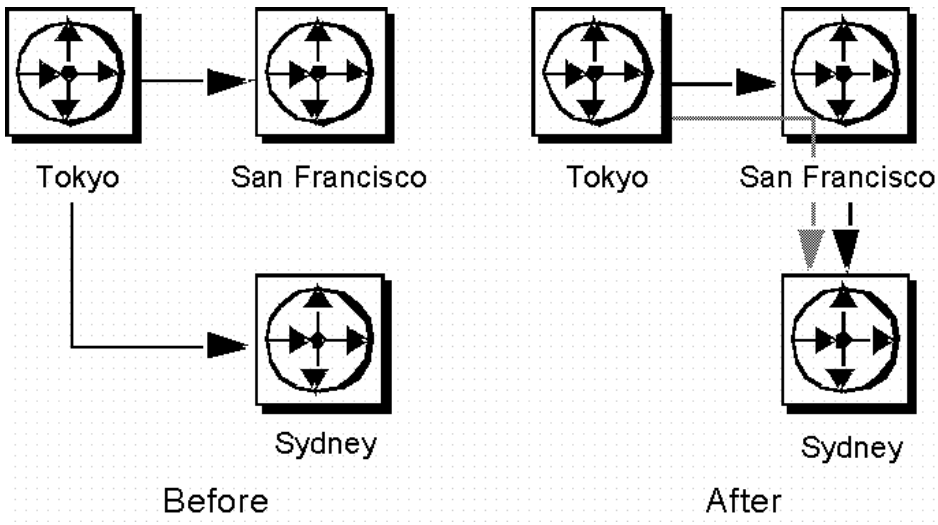
  ```
  create route to SYDNEY_RS
      set username SYDNEY_rsi_user
      set password SYDNEY_rsi_passwd
  ```

- **示例 2** – 此命令是在 TOKYO_RS 上输入的，它将从 TOKYO_RS 到 SYDNEY_RS 的直接路由更改为间接路由，并将 SF_RS 指定为中间 Replication Server：

  ```
  alter route to SYDNEY_RS
   set next site SF_RS
  ```

  此图显示了更改路由方案前后的路由。

图 1： 示例 1 和 2 在更改路由方案前后的路由



示例 3 和 4 将更改路由，以使 TOKYO_RS 重新将消息直接发送到 SYDNEY_RS，而不是经过 SF_RS 传递。

- **示例 3** – 此命令是在 TOKYO_RS 上输入的，它将从 TOKYO_RS 到 SYDNEY_RS 的路由从间接路由更改为直接路由：

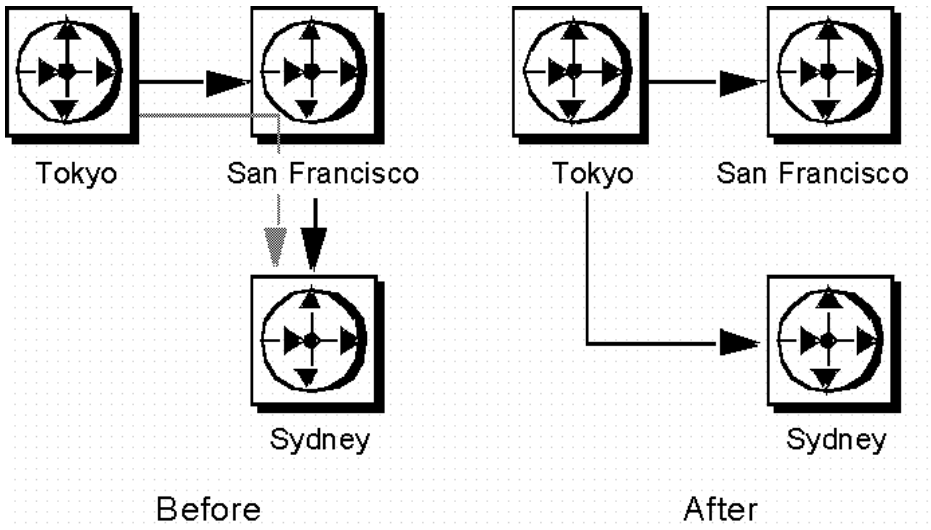
```
alter route to SYDNEY_RS
 set username SYDNEY_rsi
 set password SYDNEY_rsi_passwd
```

- **示例 4** – 此命令是在 SF_RS 上输入的，它删除从 SF_RS 到 SYDNEY_RS 的直接路由：

```
drop route to SYDNEY_RS
```

总之，示例 3 和 4 中的命令消除了示例 1 和 2 中的影响。此图显示了输入第二组命令后的路由。

图 2： 更改路由后



在示例 5 中，存在从 TOKYO_RS 到 SYDNEY_RS 以及从 SYDNEY_RS 到 SF_RS 的直接路由，并且存在从 TOKYO_RS 经由 SYDNEY_RS 到 SF_RS 的间接路由。此示例将更改此路由方案，以便 TOKYO_RS 可通过其它 Replication Server（在洛杉矶的 LA_RS）将消息传递到 SF_RS。

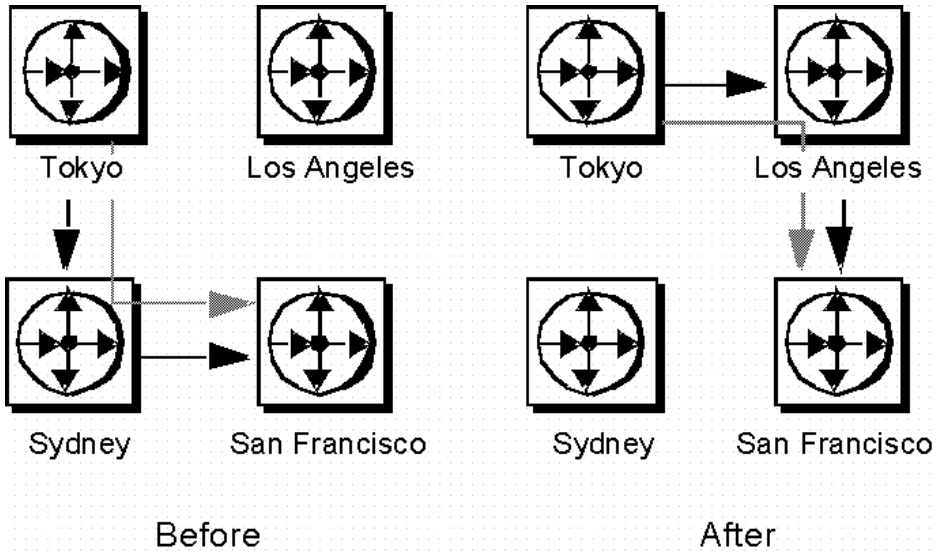
- **示例 5** – 此命令是在 TOKYO_RS 上输入的，它将间接路由的中间 Replication Server 更改为 LA_RS，而不是 SYDNEY_RS。

```
alter route to SF_RS
 set next site LA_RS
```

必须先创建从 TOKYO_RS 到 LA_RS 以及从 LA_RS 到 SF_RS 的直接路由，才能更改此路由。

此图显示了输入必要的命令前后的路由。（由于可能已删除了到 SYDNEY_DS 以及源自 SYDNEY_DS 的直接路由，因而没有显示这些路由。）

图 3: 执行必要的命令前后



- **示例 6** – 此命令是在 TOKYO_RS 上输入的，它更改从 TOKYO_RS 到 LA_RS 的直接路由的口令。新口令为“LApass”。

```
alter route to LA_RS
 set password LApass
```

更改直接路由的口令之前，必须使用 **suspend route** 挂起该路由。

- **示例 7** – 将到 LA_RS 的路由的安全服务设置为 DCE:

```
suspend route to LA_RS

 alter route to LA_RS
 set security_mechanism to 'dce'

 resume route to LA_RS
```

用法

- 使用 **alter route** 进行以下更改:
  - 将直接路由更改为间接路由。
  - 将间接路由更改为直接路由。
  - 更改现有路由中的下一个中间节点。
  - 更改现有直接路由的 RSI 用户的口令。
  - 更改路由配置参数。
  - 更改基于网络的安全性参数。

  有关路由的概述，请参见《Replication Server 管理指南第一卷》。

- 在直接路由的源 Replication Server 上执行 **alter route**。
- 要将直接路由更改为间接路由，或更改间接路由的中间节点，可使用 **set next site** *thru_replication_server*。
- 如果要将直接路由更改为间接路由，必须首先创建从源节点到中间节点以及从中间节点到目标节点的直接路由。使用 **create route** 完成此操作。
- 如果要更改间接路由的中间节点，必须首先创建从新中间节点到目标节点以及从新中间节点到目标节点的直接路由。使用 **create route** 完成此操作。
- 间接路由可以具有一个或多个中间 Replication Servers。例如，从 A_RS 到 D_RS 的间接路由可以通过中间节点 B_RS 和 C_RS 传递。
- 要将间接路由更改为直接路由，请使用不带 **set next site** 子句的 **alter route**，并指定要在目标 Replication Server 上使用的登录名和口令。例如，将间接路由从 A_RS->B_RS->C_RS 更改为直接路由 A_RS->C_RS。
- 要将一个中间节点替换为下一个中间节点，请执行带有 **set next site** 子句的 **alter route**。例如，将间接路由从 A_RS->B_RS->C_RS->D_RS 更改为 A_RS->C_RS->D_RS。
- 您可以使用 **configure route** 或 **alter route** 参数来设置路由参数。
- 使用 **suspend route** 在更改路由之前挂起路由上的活动。

设置口令和用户名

- 只有要将间接路由更改为直接路由的情况下，才使用 **set username** *user* 和 **set password** *passwd*。您不能更改间接路由的用户名或口令；如果尝试这样操作，会使间接路由更改为直接路由。
- 只有要更改直接路由的口令的情况下，才使用 **set password** *passwd*。更改直接路由的口令之前，先使用 **suspend route**。

路由参数

- 设置保存间隔使系统可以承受目标 Replication Server 上的分区或稳定队列故障。在恢复期间，可以使用 **rebuild queues** 命令将积压的消息发送到目标 Replication Server。
  有关保存间隔和稳定队列恢复的详细信息，请参见《Replication Server 管理指南第二卷》。
- Sybase 建议将 **rsi_batch_size**、**rsi_fadeout_time**、**rsi_packet_size** 和 **rsi_sync_interval** 参数的值保留为其缺省值以优化性能。
- 在使用 **alter route** 更改路由参数之前，必须先挂起连接。执行 **alter route** 命令后，必须恢复路由以使更改生效。

基于网络的安全性参数

- 路由两端必须使用具有相同安全性机制和安全性功能的兼容安全控制层 (SCL) 驱动程序。复制系统管理员负责为每个服务器选择和设置安全性功能。在尝试建立连接之前，Replication Server 不会查询远程服务器的安全性功能。如果路由两端的安全性功能不兼容，连接将失败。

- **alter route** 更改出站连接（从 Replication Server 到目标 Replication Server）的基于网络的安全性设置。**alter route** 所设置的安全性参数覆盖由 **configure replication server** 设置的缺省值。
- 如果将 **unified_login** 设置为“required”，则只有“sa”用户可以无需认证而登录到 Replication Server。如果安全性机制失败，“sa”用户可以使用口令登录到 Replication Server，并禁用 **unified_login**。
- Replication Server 可具有多种安全性机制；所支持的每种机制都在 `libtcl.cfg` 文件中的 SECURITY 下列出。
- 消息加密进程占用的资源较多，会造成严重的性能下降。大多数情况下，明智的做法是只将某些连接的 **msg_confidentiality** 设置为“on”。或者，选择占用资源较少的安全性功能，如 **msg_integrity**。
- 在使用 **alter route** 更改安全性参数之前，必须先挂起连接。执行 **alter route** 命令后，应恢复路由以使更改生效。

更改路由的过程

注意： 如果要更改配置参数，只需在执行 **alter route** 之前挂起路由。

1. 停顿复制系统。有关详细信息，请参见《Replication Server 故障排除指南》。
2. 在每个使用 RepAgent 管理数据库的 Replication Server 上，使用 **suspend log transfer** 挂起日志传送。
3. 在源 Replication Server 上执行 **alter route** 命令。可以根据需要更改任意多个路由。
4. 使用 **resume log transfer** 恢复到每个 RSSD 和用户数据库的 RepAgent 连接。
   有关更改路由的完整过程，请参见《Replication Server 管理指南第一卷》。

权限

alter route 需要“sa”权限。

另请参见

- admin quiesce_check （第 57 页）
- admin quiesce_force_rsi （第 58 页）
- alter connection （第 109 页）
- alter logical connection （第 146 页）
- alter queue （第 150 页）
- configure connection （第 180 页）
- create logical connection （第 252 页）
- create replication definition （第 258 页）
- configure replication server （第 181 页）
- drop logical connection （第 305 页）
- create connection （第 214 页）
- create route （第 273 页）
- drop connection （第 297 页）

- drop route （第 309 页）
- resume log transfer （第 324 页）
- set proxy （第 331 页）
- suspend log transfer （第 336 页）
- suspend route （第 337 页）

alter schedule

启用或禁用执行命令的日程表。

语法

alter schedule *sched_name* set [on|off]

参数

- **sched_name** – 要更改的日程表的名称。
- **set [on | off}** – 启用或禁用日程表。缺省情况下，创建的日程表处于 on 状态。

示例

- **示例 1** – 若要禁用 schedule1，请输入：

```
alter schedule schedule1 set off
```

用法

在 Replication Server 上启用或禁用日程表。

权限

alter schedule 需要“sa”权限。

另请参见

- alter connection （第 109 页）
- drop schedule （第 311 页）
- configure replication server （第 181 页）
- create schedule （第 277 页）

alter subscription

在使用同一 Replication Server 的同一复制数据库的复制连接之间移动预订而无需重新实现。预订的对象可以是数据库复制定义、表复制定义、函数复制定义或发布。

语法

```
alter subscription sub_name
     for {table_repdef | func_repdef | {publication pub | database
replication definition db_repdef}
     with primary at pri_dataserver.pri_database
     move replicate from data_server1.database1 to
data_server2.database2
```

参数

- **sub_name** – 预订的名称，此名称必须符合标识符的命名规则。预订的名称对于复制定义（如果适用）、复制数据服务器和数据库必须是唯一的。
- **for table_rep_def** – 指定所预订的表复制定义。
- **for function_rep_def** – 指定所预订的函数复制定义的名称。
- **for publication pub_name** – 指定预订的发布。
- **for database replication definition db_repdef** – 指定所预订的数据库复制定义。
- **with primary at *data_server.database*** – 对于发布的预订或数据库复制定义，使用此子句。指定主数据的位置。如果主数据库是使用逻辑连接的热备份应用程序的一部分，则 *data_server.database* 是逻辑数据服务器和数据库的名称。如果配置的是多路径复制系统，则也可以在该子句中指定替代主连接名。
- **move replicate from *data_server1.database1* to *data_server2.database2*** – 指定您要将 *sub_name* 预订从 *data_server1.database1* 复制连接移动到 *data_server2.database2* 连接。

  如果复制数据库是热备份应用程序的一部分，则 *data_server.database* 是逻辑数据服务器和数据库的名称。如果配置的是多路径复制系统，则也可以在该子句中指定替代复制连接名。

示例

- **示例 1** – 例如，要将 **rep1** 复制定义的 **sub1** 预订从 `RDS.rdb1` 连接移动到 `RDS.rdb2` 连接，请输入：

  ```
  alter subscription sub1 for rep1
  move replicate from RDS.rdb1
  to RDS.rdb2
  ```

用法

- 如果已经创建了一个或多个替代复制连接，则使用 **alter subscription** 在复制连接之间移动预订。请参见《Replication Server 管理指南第二卷》的“性能调优”的“Multi-Path Replication”中的“Moving Subscriptions Between Connections”（在连接之间移动预订）。
- 在存储复制数据的数据库所在的 Replication Server 上执行 **alter subscription**。
- 有关预订及其在复制中所起作用的详细信息，请参见《Replication Server 管理指南第一卷》。
- 如果主 Replication Server 版本早于 1570，则无法使用 **alter subscription**，而必须删除并在所需的连接中重新创建预订。
- 要删除必须通过同一路径进行复制的多个预订，请挂起主连接的日志传送，并在移动所有预订后重新开始日志传送。

权限

要执行 **create subscription**，必须具有以下登录名和权限：

- 在复制 Replication Server、主 Replication Server 和主 Adaptive Server 数据库上具有相同的登录名和口令。
- 在输入此命令的复制 Replication Server 上具有“创建对象”或“sa”权限。
- 在主 Replication Server 上具有“create object”、“primary subscribe”或“sa”权限。
- 在主 Adaptive Server 数据库中的主表上具有 **select** 权限。
- 在主 Adaptive Server 数据库中的 **rs_marker** 存储过程上，具有 **execute** 权限。
- 复制数据库的维护用户在复制表上必须具有 **select**、**insert**、**update** 和 **delete** 权限，而且对复制中使用的函数必须具有 **execute** 权限。

另请参见

- create alternate connection （第 202 页）
- create subscription （第 280 页）

alter user

更改用户口令。

语法

```
alter user user
   set password {new_password | null}
   [verify password old_password]
   [set password_parameter to 'parameter_value']
```

参数

- **user** – 用户的登录名。
- **new_password** – 如果创建或更改口令，则为新口令。
- **old_password** – 如果使用 *verify password* 参数，则为当前用户口令。
- **parameter 和 parameter_value** – 您可以设置的参数和相应值。

表 21. 口令参数

password_parameter	说明和*值*
password_expiration	口令到期之前经过的天数。 缺省值为 0，指示口令永不过期。 如果口令已过期，Replication Server 将锁定用户帐户并通知用户。如果不更改口令，则断开连接后，您将无法登录，直到管理员重置口令。新口令必须满足所有口令要求。 对于 **rs_init** 创建的具有 **connect source** 权限的任何用户或 ID 用户，口令不会过期。这些口令会覆盖您在 Replication Server 中为所有用户设置的任何 **password_expiration** 设置。数据库、其它 Replication Server 和 Replication Agent 使用具有 **connect source** 权限的用户 ID。 管理员应考虑对为复制代理或 RSI 创建的任何用户进行适当设置，以使口令不会过期。

示例

- **示例 1** – 用户“louise”将口令从“EnnuI”更改为“somNIfic”：

```
alter user louise
 set password somNIfic
 verify password EnnuI
```

- **示例 2** – 将用户 jsmith 的口令有效期更改为 60 天：

```
alter user jsmith
set password to newpass
set password_expiration to '60'
```

用法

- 如果 Replication Server 使用 ERSSD，则可以使用以下命令更改 ERSSD 主用户口令：

```
alter user user
set password new_password
```

  如果此用户名与 ERSSD 主用户名相匹配，ERSSD 将更新 *rs_users* 表，在 ERSSD 上发出 **sp_password** 以更改口令并更新配置文件行 *RSSD_primary_pw_enc*。

- 具有“sa”权限的用户可以省略 **verify password** 子句。其他用户必须提供此子句才能更改各自的口令。
- 使用 **alter user** 为单个用户指定的口令设置覆盖使用 **configure replication server** 设置的任何值。

  请参见 **configure replication server** 中的表 24. 口令参数表。

权限

更改其它用户的口令时，**alter user** 需要“sa”权限。

另请参见

- create user （第 288 页）
- drop user （第 315 页）

assign action

将 Replication Server 错误处理操作指派给 DSI 线程接收的数据服务器或 Replication Server 错误。

语法

```
assign action
   {ignore | warn | retry_log | log | retry_stop | stop_replication}
   for error_class
   to server_error1 [, server_error2]...
```

参数

- **ignore** – 指示 Replication Server 忽略错误并继续处理。如果数据服务器错误代码表示执行成功或表示某一不合理的警告，则应当使用 **ignore**。
- **warn** – 指示 Replication Server 在其日志文件中显示警告消息，而无需回退事务或中断操作执行。
- **retry_log** – 指示 Replication Server 回退事务并重试此事务。用 **alter connection** 设置尝试进行重试的次数。如果重试后仍存在错误，Replication Server 则将事务写入例外日志并执行下一个事务。
- **log** – 指示 Replication Server 回退当前的事务，并将其记录在例外日志中，然后执行下一个事务。
- **retry_stop** – 指示 Replication Server 回退事务并重试此事务。用 **alter connection** 设置尝试进行重试的次数。如果重试后仍存在错误，Replication Server 将挂起此数据库的复制。
- **stop_replication** – 指示 Replication Server 回退当前事务并挂起此数据库的复制。此操作相当于使用 **suspend connection**。
- **error_class** – 将操作指派给的错误类名称。

- **server_error** – 数据服务器或 Replication Server 错误号。

示例

- **示例 1** – 指示 Replication Server 忽略数据服务器错误 5701 和 5703：

```
assign action ignore
 for pubs2_db_err_class
 to 5701, 5703
```

- **示例 2** – 在 Replication Server 遇到行数错误（由错误号 5186 指示）时发出警告：

```
assign action warn
   for rs_repserver_error_class to 5186
```

如果出现行数错误，则会显示下面的错误消息：

```
DSI_SQLDML_ROW_COUNT_INVALID 5186
Row count mismatch for the SQL Statement Replication
command executed on 'mydataserver.mydatabase'. The
command impacted 10 rows but it should impact 15 rows."
```

用法

- 使用 **assign action** 指示 Replication Server 如何处理数据服务器返回的错误。此命令将覆盖以前指派给数据服务器错误的所有操作。
- 在执行 **create error class** 的主节点执行 **assign action**。
- 在 Replication Server 15.6 和更高版本中，行计数验证错误消息显示表名。请参见《Replication Server 管理指南第二卷》的“错误和例外处理”的“数据服务器错误处理”的“非 SQL 语句复制的行计数验证”中的“在行计数验证错误消息中显示表名”。
- 首先指派错误类的操作，然后创建任何使用此错误类的分发。指派活动分发的操作会导致不可预料的结果。
- 如果没有为数据服务器错误指派操作，则会执行缺省操作 **stop_replication**。对于 Replication Server 错误，执行的缺省操作取决于出现的错误类型。有关支持的 Replication Server 错误以及这些错误的缺省操作的列表，请参见表 22. Replication Server 错误类错误号更新。
- 确保指派适合于错误情况。例如，如果在 **begin transaction** 命令失败时，将 **ignore** 操作指派给数据服务器返回的错误，则后续的 **commit** 或 **rollback** 命令就可能会产生不可预料的错误。
- 通过客户机/服务器接口错误处理机制，数据服务器将错误返回到 Replication Server。警告和错误消息被写入 Replication Server 日志文件中。
- 通过复制系统，Replication Server 将错误操作分发到合格的节点。通常由于复制系统有一定时间的滞后，因此更改不会立即显示。

多个错误的错误操作

- 如果操作导致多个错误，Replication Server 将选择对这组错误执行最严重的错误操作。例如，如果一个错误表示已回退事务，并为此错误指派了 **retry_log** 操作，而另一个错误表示事务日志已满，并为此错误指派了 **stop_replication** 操作，则对

于返回这两个错误的事务，Replication Server 将执行 **stop_replication** 操作。错误操作按严重级别从低级到高级排列如下：

1.**ignore**

2.**warn**

3.**retry_log**

4.**log**

5.**retry_stop**

6.**stop_replication**

rs_sqlserver_error_class 错误操作

- 为 *rs_sqlserver_error_class* 错误类提供了 Adaptive Server 预定义错误操作。
- 要在 *rs_sqlserver_error_class* 中指派其它错误操作，必须首先选择此错误类的主节点。在此节点登录到 Replication Server，然后使用 **create error class** 创建错误类。

rs_repserver_error_class 错误操作

- **rs_repserver_error_class** 错误类提供了 Replication Server 预定义错误操作。
- 要为 **rs_repserver_error_class** 指派其它错误操作，必须首先选择此错误类的主节点。登录到主节点上的 Replication Server，然后使用 **create replication server error class** 创建错误类。
- “Replication Server 错误类错误号更新”表列出了有效的 Replication Server 错误及其缺省错误操作。

表 22. Replication Server 错误类错误号更新

server_error	错误消息	缺省错误操作	说明
5185	`Row count mismatch for the command executed on ‘data-server.database’. The command impacted x rows but it should impact y rows.`	**stop_replication**	如果在将不属于 SQL 语句复制的一部分的命令、存储过程或启用了自动更正的行更改发送到数据服务器之后，受影响的行数与预期行数不同，则会显示此消息。
5186	`Row count mismatch for the command executed on ‘data-server.database’. The command impacted x rows but it should impact y rows.`	**stop_replication**	SQL 语句复制影响的行数与预期的影响行数不相同时发生的行数验证错误。

server_error	错误消息	缺省错误操作	说明
5187	Row count mismatch for the autocorrection delete command executed on '*data-server.database*'. The command deleted *x* rows but it should delete *y* rows.	**stop_replication**	如果在将 delete 命令发送到数据服务器并启用了自动更正之后，受影响的行数与预期行数不同，则会显示此消息。
5193	You cannot enable autocorrection if SQL Statement Replication is enabled. Either enable SQL Statement Replication only or disable SQL StatementReplication before you enable autocorrection.	**stop_replication**	在启用 SQL 语句复制的情况下，无法启用自动更正。请仅启用 SQL 语句复制，或者在启用自动更正之前禁用 SQL 语句复制
5203	Row count mismatch on '*data-server.database*'. The delete command generated by dsi_command_convert deleted *x* rows, whereas it should delete *y* rows.	**stop_replication**	如果已删除的行数与预期要删除的行数不同，则会显示此消息。

- 有关 **rs_repserver_error_class** 的信息，请参见“错误类和函数类”表。

显示错误操作

rs_helperror 存储过程显示映射到给定数据服务器错误号的 Replication Server 错误操作。

权限

assign action 需要“sa”权限。

另请参见

- alter error class （第 138 页）
- configure connection （第 180 页）
- create connection （第 214 页）
- create error class （第 228 页）
- drop error class （第 299 页）
- rs_helperror （第 518 页）
- suspend connection （第 334 页）

check publication

查找发布状态以及发布所包含的项目数。

语法

```
check publication pub_name
   with primary at data_server.database
```

参数

- **pub_name** – 要检查的发布的名称。
- **with primary at data_server.database** – 指定主数据的位置。如果主数据库是热备份应用程序的一部分，则 *data_server.database* 是逻辑数据服务器和数据库的名称。

示例

- **示例 1** – 检查发布 *pubs2_pub* 的状态，其中主数据库是 TOKYO_DS.*pubs2*:

```
check publication pubs2_pub
 with primary at TOKYO_DS.pubs2
```

用法

- 使用 **check publication** 查找发布状态以及发布所包含的项目数。
  有关发布的详细信息，请参见《Replication Server 管理指南第一卷》。
- 在管理复制数据库的 Replication Server 或管理主数据库的 Replication Server 上执行 **check publication**。
- 如果在复制 Replication Server 上执行 **check publication**，将在主 Replication Server 上使用当前用户名和口令来检查发布。在主 Replication Server 上必须具有相同的登录名和口令，才能显示有关发布的当前信息。
- 要检查预订状态，请使用 **check subscription**。有关详细信息，请参见 **check subscription** 命令。

check publication 所返回的消息

- 如果在主 Replication Server 或复制 Replication Server 上执行 **check publication**，将返回以下消息之一:

```
Publication pub_name for primary database
data_server.database is valid. The number of
articles in the publication is number_articles.
Publication pub_name for primary database
data_server.database is invalid. The number of
articles in the publication is number_articles.
```

- 如果在复制 Replication Server 上执行 **check publication**，但无法与主 Replication Server 联系，将返回以下消息:

```
Failed to get publication information from primary.
```

权限

任何用户都可以执行此命令。在复制 Replication Server 上输入此命令的用户必须在主 Replication Server 中具有相同的登录名和口令。

另请参见

- check subscription （第 177 页）
- create publication （第 254 页）
- validate publication （第 389 页）

check subscription

查找复制定义或发布的预订的实现状态。

语法

```
check subscription sub_name
    for {table_rep_def | function_rep_def |
    [publication pub_name | database replication definition
db_repdef]
    with primary at data_server.database}
    with replicate at data_server.database
```

参数

- **sub_name** – 要检查的预订的名称。
- **for table_rep_def** – 指定所预订的表复制定义的名称。
- **for function_rep_def** – 指定所预订的函数复制定义的名称。
- **for publication pub_name** – 指定所预订的发布的名称。
- **database replication definition db_repdef** – 指定所预订的数据库复制定义的名称。
- **with primary at data_server.database** – 指定主数据的位置。如果主数据库是热备份应用程序的一部分，则 *data_server.database* 是逻辑数据服务器和数据库的名称。只有对发布的预订才使用此子句。
- **with replicate at data_server.database** – 指定复制数据的位置。如果复制数据库是热备份应用程序的一部分，则 *data_server.database* 是逻辑数据服务器和数据库的名称。

示例

- **示例 1** – 检查对复制定义 *titles_sub* 的预订 *titles_rep* 的状态，其中复制数据库是 SYDNEY_DS.*pubs2*:

```
check subscription titles_sub
 for titles_rep
 with replicate at SYDNEY_DS.pubs2
```

- **示例 2** – 检查对发布 *pubs2_sub* 的预订 *pubs2_pub* 的状态，其中主数据库是 TOKYO_DS.*pubs2*，复制数据库是 SYDNEY_DS.*pubs2*:

```
check subscription pubs2_sub
 for publication pubs2_pub
 with primary at TOKYO_DS.pubs2
 with replicate at SYDNEY_DS.pubs2
```

用法

- 在预订实现或取消实现期间或刷新发布预订的过程中，使用 **check subscription** 查找预订的状态。预订的对象可以是表复制定义、函数复制定义或发布。
  有关预订的详细信息，请参见《Replication Server 管理指南第一卷》。
- 在管理存储复制数据的数据库的 Replication Server 或管理主数据库的 Replication Server 上执行 **check subscription**。
  根据执行命令的位置不同，**check subscription** 的结果会有所不同。如果 Replication Server 同时管理主数据库和复制数据库，**check subscription** 将返回两种状态消息。
- 要检查发布状态，请使用 **check publication**。有关详细信息，请参见 **check publication** 命令。
- 有关使用 **check subscription** 监控预订的详细信息，请参见《Replication Server 故障排除指南》。

check subscription 所返回的消息

- 如果在复制 Replication Server 上执行 **check subscription**，将返回以下消息之一。在热备份应用程序中，共有两行输出，它们显示活动数据库和备用复制数据库的状态。

INVALID	*sub_name* doesn’t exist
REMOVING	REMOVING subscription *sub_name* from system tables at the Replicate.
DEMATERIALIZING	Subscription *sub_name* is DEMATERIALIZING at the Replicate.
VALID	Subscription *sub_name* is VALID at the Replicate.
VALIDATING	Subscription *sub_name* is VALIDATING at the Replicate.
MATERIALIZED	Subscription *sub_name* has been MATERIALIZED at the Replicate.

ACTIVE	Subscription *sub_name* is ACTIVE at the Replicate.
ACTIVATING	Subscription *sub_name* is ACTIVATING at the Replicate.
ACTIVATING	Subscription *sub_name* is ACTIVATING at the Standby of the Replicate.
QCOMPLETE 和 ACTIVE	Subscription *sub_name* is ACTIVE at the Replicate and Materialization Queue has been completed.
QCOMPLETE	Materialization Queue for Subscription *sub_name* has been completed.
ACTIVE 和非 QCOMPLETE	Subscription *sub_name* is ACTIVE at the Replicate, but Materialization Queue for it has not been completed.
DEFINED	Subscription *sub_name* has been defined at the Replicate.

- 除以上消息外，在复制 Replication Server 上执行 **check subscription** 可能会返回以下消息之一：

ERROR	Subscription *sub_name* has experienced an unrecoverable error during Materialization or Dematerialization.Please consult the error log for more details.
PENDING	Other subscriptions are being created or dropped for the same replication definition/database.Subscription *sub_name* will be processed when previous requests are completed.
RECOVERING	Subscription *sub_name* has experienced a recoverable error during Materialization or Dematerialization.It will be recovered by Subscription Daemon (dSub).

- 如果在主 Replication Server 上执行 **check subscription**，将返回以下消息之一：

INVALID	*subscription_name* doesn’t exist
DEMATERIALIZING	Subscription *sub_name* is DEMATERIALIZING at the PRIMARY.
VALID	Subscription *sub_name* is VALID at the PRIMARY.

ACTIVE	Subscription *sub_name* is ACTIVE at the PRIMARY.
ACTIVATING	Subscription *sub_name* is ACTIVATING at the PRIMARY.
DEFINED	Subscription *sub_name* has been defined at the PRIMARY.

权限

任何用户都可以执行此命令。

另请参见

- activate subscription （第 44 页）
- check publication （第 176 页）
- create subscription （第 280 页）
- define subscription （第 290 页）
- drop subscription （第 312 页）
- validate subscription （第 390 页）

configure connection

更改数据库连接的属性。

注意： configure connection 等同于 **alter connection** 命令。

语法

有关语法信息，请参见 **alter connection**。

用法

有关用法信息，请参见 **alter connection**。

configure logical connection

更改逻辑连接的属性。

注意： configure logical connection 等同于 **alter logical connection**。

语法

有关语法信息，请参见 **alter logical connection**。

用法

有关用法信息，请参见 **alter logical connection**。

configure replication server

设置 Replication Server 的特性，包括基于网络的安全性。配置 ERSSD。

语法

```
configure replication server {
    set repserver_param to 'value' |
    set route_param to 'value' |
    set database_param to 'value' |
    set logical_database_param to 'value' |
    set password_param to 'value' |
    set security_param to 'value' |
    set id_security_param to 'value' |
    set security_services [to] 'default'} |
    set parameter to 'parameter_value'
```

参数

- **repserver_param** – 影响 Replication Server 的参数。请参见表 23. Replication Server 配置参数 和表 28. ERSSD 配置参数。
- **value** – 配置参数设置。

表 23. Replication Server 配置参数

repserver_param	值
audit_enable	将 **audit_enable** 设置为 on 可启用命令审计。缺省值为 off。
audit_dest	如果启用命令审计，指定命令审计日志的文件名和路径。 缺省值为 log。 在 Unix 中，如果日志文件不存在，Replication Server 将使用 0600 权限创建该文件。如果您在 UNIX 中使用不同权限（例如 0666）创建自己的日志文件，Replication Server 将保留您的权限。

repserver_param	值
block_size to '*value*' with shutdown	指定队列块大小，表示稳定队列结构使用的连续内存块中的字节数。 允许的值的范围：16KB、32KB、64KB、128KB 或 256KB 缺省值：16KB **注意：** 在执行此命令更改块大小之后，Replication Server 将关闭。在 Replication Server 15.6 之前的版本中，指定块大小后必须包括“**with shutdown**”子句。在 15.6 和更高版本中，“**with shutdown**”子句是可选的；您无需重新启动 Replication Server 便可使队列块大小的更改生效。只应使用 **configure replication server** 命令更改此参数。否则会损坏队列。 许可证：在高级服务选件下单独许可。 有关说明，请参见《Replication Server 管理指南第二卷》的“性能调优”中的“Replication Server – 高级服务选件”。
cm_fadeout_time	Replication Server 断开与 RSSD 的连接之前的空闲秒数。值 -1 指定从不关闭连接。 缺省值：300 秒 最小值：1 秒 最大值：2,147,483,648 秒
cm_max_connections	连接管理器最多可以使用的出站连接数。此值必须大于 0。 缺省值：64
current_rssd_version	此 RSSD 支持的 Replication Server 版本。Replication Server 在启动时检查此值。 **注意：** 不要更改此参数的值。只有在升级或降级的情况下才应使用 **rs_init** 程序修改此值。 缺省值：不适用
deferred_queue_size	Open Server 延迟队列的最大大小。如果超过 Open Server 限制，应增大此最大大小。**deferred_queue_size** 值必须大于 0。 **注意：** 必须重新启动 Replication Server 以使此参数的任何更改生效。 缺省值：Linux 和 HPIA32 上为 2048；其它平台上为 1024
dist_direct_cache_read	让分配器 (DIST) 线程能直接从稳定队列事务线程 (SQT) 高速缓存中读取 SQL 语句。这减少了 SQT 的负载以及二者之间的依赖性，并提高了 SQT 和 DIST 的效率 缺省值：off 许可证：在高级服务选件下单独许可。请参见《Replication Server 管理指南第二卷》的“性能调优”中的“高级服务选件”。

repserver_param	值
ha_failover	启用或禁用对新的数据库连接（从 Replication Server 到 Adaptive Server）的 Sybase 故障切换支持。这些值为： • on - 启用故障切换 • off – 禁用故障切换 缺省值：off
id_server	此 Replication Server 的 ID Server 的名称。 **注意：** 不要更改此参数的值。在运行 **rs_init** 时，将设置 **id_server**；在升级或降级 Replication Server 时，只应通过 **rs_init** 程序对其进行修改。 缺省值：不适用
init_sqm_write_delay	如果稳定队列正在被读取，则表示“稳定队列管理器”写队列的延迟时间。 缺省值：100 毫秒
init_sqm_write_max_delay	如果稳定队列没有被读取，则表示“稳定队列管理器”写队列的最大延迟时间。 缺省值：1000 毫秒
mem_reduce_malloc	启用后可以在更大的块中分配内存，这会达到内存分配数量，从而提高 Replication Server 性能。 缺省值：off 许可证：在高级服务选件下单独许可。请参见《Replication Server 管理指南第二卷》的“性能调优”中的“Replication Server – 高级服务选件”。
mem_thr_dsi	指定用于强制 DSI 线程停止填充 SQT 高速缓存的总内存的百分比。 缺省值：**memory_limit** 值的 80%。 范围：1 – 100
mem_thr_exec	指定用于强制 EXEC 线程停止接收来自 RepAgent 的命令的总内存的百分比。 缺省值：**memory_limit** 值的 90%。 范围：1 – 100
mem_thr_sqt	指定用于强制 SQT 线程刷新其高速缓存中的最大事务的总内存的百分比。 缺省值：**memory_limit** 值的 85%。 范围：1 – 100

repserver_param	值
mem_warning_thr1	指定在第一个警告消息生成之前使用的总内存阈值百分比。请参见 **memory_limit**。 缺省值：**memory_limit** 值的 80%。 范围：1 – 100
mem_warning_thr2	指定在第二个警告消息生成之前使用的总内存阈值百分比。请参见 **memory_limit**。 缺省值：**memory_limit** 值的 90%。 范围：1 – 100
memory_control	管理需要大量内存的线程的内存控制行为。请参见 **memory_limit**。 值为： • on – 启用内存控制 • off – 禁用内存控制 缺省值：on

repserver_param	值
memory_limit	Replication Server 可以使用的最大内存总量（以兆字节为单位）。 **memory_limit** 指示的内存池中的可用内存量与其它几个配置参数的值直接相关。这些参数包括 **md_sqm_write_request_limit**、**queue_dump_buffer_size**、**sqt_max_cache_size**、**sre_reserve** 和 **sts_cachesize**。 缺省值：2,047MB 对于 32 位 Replication Server： • 最小值 – 0 • 最大值 – 2,047MB 对于 64 位 Replication Server： • 最小值 – 0 • 最大值 – 2,147,483,647 如果 **memory_control** 为： • on – 当内存消耗超过 **memory_limit** 时，Replication Server 不关闭 • off – 当内存消耗超过 **memory_limit** 时，Replication Server 自动关闭 监控内存使用情况并在需要时增大 **memory_limit**。 当它超过 **memory_limit** 中定义的可用内存值时，Replication Server 关闭。监控内存使用情况并在需要时增大 **memory_limit**。 在 Replication Server 中，需要大量内存的线程是： • EXEC • SQT • DST 这些线程在接收或处理新数据之前通过执行内存使用量检查来执行内存控制。在内存控制期间，如果内存使用量很高，则会通过以下措施来调整线程运行情况： • 阻止线程对新数据进行分组，并清除和处理现有数据；或者， • 使线程进入休眠状态，以使它在内存可用之前不接收新数据。 如果您设置的值大于 2,047MB，则降级会将该值重置为 2,047MB 以防止溢出。
minimum_rssd_version	可使用此 RSSD 的 Replication Server 的最低版本。如果 **current_rssd_version** 大于 Replication Server 的版本时，Replication Server 启动时会检查该值。 **注意：** 不要更改此参数的值。只有在升级或降级的情况下才应使用 **rs_init** 程序修改此值。 缺省值：不适用

repserver_param	值
nrm_thread	启用 NRM 线程，Replication Server 可使用 NRM 线程在由 RepAgent 执行程序线程进行分析的同时并行规范化并打包日志传送语言 (LTL) 命令。NRM 线程进行的并行处理减少了 RepAgent 执行程序线程的响应时间。NRM 线程是从 RepAgent 执行程序线程拆分的线程。 在使用 **exec_nrm_request_limit** 之前使用 **configure replication server** 命令将 **nrm_thread** 设置为 on。 缺省值：off 许可证：在高级服务选件下单独许可。请参见《Replication Server 管理指南第二卷》的“性能调优”中的“Replication Server – 高级服务选件”。
num_client_connections	允许的最大传入客户端连接数。如果超过 Open Server 限制，应增大此最大值。此值必须大于或等于 30。 缺省值：30
num_concurrent_subs	所允许的最大并发预订实现/取消实现请求数。（限制仅适用于基本和非原子实现；不适用于批量实现。）如果超过此最大值，将在其它请求完成后执行超出的请求。最小值为 1。 缺省值：10
num_msg_queues	所允许的最大 Open Server 消息队列数。如果超过 Open Server 限制，应增大此最大值。此值必须大于 **num_threads** 的设置。 缺省值：300
num_msgs	所允许的最大 Open Server 消息队列的消息数。如果超过 Open Server 限制，应增大此最大值。 缺省值：91,136
num_mutexes	所允许的最大 Open Server 互斥项数。如果超过 Open Server 限制，应增大此最大值。此值必须大于 **num_threads** 的设置。 缺省值：128
num_stable_queues	所允许的最大稳定队列数（仅适用于 HP9000）。每个稳定队列使用 32,768 个字节共享内存。允许的最小稳定对列数是 32。 每个备用数据库连接使用额外的 16,384 个字节的共享内存。每两个备用数据库连接算作一个额外的稳定队列。 缺省值：32
num_threads	所允许的最大 Open Server 线程数。如果超过 Open Server 限制，应增大此最大值。此值必须大于或等于 20。 缺省值：150

repserver_param	值
oserver	当前 Replication Server 的名称。 **注意：** 不要更改此参数的值。在使用 **rs_init** 安装 Replication Server 时，就已指定了当前 Replication Server 的名称。 缺省值：不适用
password_encryption	不建议使用此参数。使用 **create user** 或 **alter user** 命令实现口令安全性。使用 **configure replication server** 在服务器级为所有 Replication Server 用户设置口令参数。
prev_min_rssd_version	进行 **rs_init** 安装升级后，此值将包含 **minimum_rssd_version** 的以前值。 **注意：** 不要更改此参数的值。只有在升级或降级的情况下才应使用 **rs_init** 修改此值。 缺省值：不适用
prev_rssd_version	进行 **rs_init** 安装升级后，此值将包含 **current_rssd_version** 的以前值。 **注意：** 不要更改此参数的值。只有在升级或降级的情况下才应使用 **rs_init** 修改此值。 缺省值：不适用
queue_dump_buffer_size	**sysadmin dump_queue** 命令所使用的最大命令长度（以字节为单位）。大于指定长度的命令将被截断。范围是 1000 到 32,768。 缺省值：32,768 字节
rec_daemon_sleep_time	指定恢复守护程序的休眠时间，该守护程序在热备份应用和某些其他操作中处理“strict”保存间隔消息。 缺省值：2 分钟
rssd_error_class	RSSD 的错误类。 缺省值：rs_sqlserver_error_class
send_enc_password	确保所有 Replication Server 客户端连接是使用加密口令创建的（与 RSSD 的第一次连接除外）。值为“on”和“off”。 有关详细信息，请参见《Replication Server 管理指南第一卷》。 缺省值：off
send_timestamp_to_standby	指定当没有复制定义时是否将 `timestamp` 列发送到复制数据库中。值为 on 和 off。 有关详细信息，请参见《Replication Server 管理指南第一卷》。 缺省值：off

repserver_param	值
smp_enable	启用对称多重处理 (SMP)。指定 Replication Server 线程应由 Replication Server 内部预定，还是由操作系统外部预定。当 Replication Server 线程由内部预定时，无论有多少个可用处理器，都会将 Replication Server 限定为一个计算机处理器。值为“on”和“off”。 缺省值：on
sqm_async_seg_delete	将 **sqm_async_seg_delete** 设置为 on 可启用用于删除段的专用守护程序并改进入站和出站队列处理的性能。 缺省值：on 必须重新启动 Replication Server 以使参数设置的任何更改生效。 如果 **sqm_async_seg_delete** 为 on，Replication Server 可能需要更大的分区。使用 **alter partition** 扩展分区。请参见： • 《Replication Server 配置指南》的“准备安装和配置 Replication Server”的“规划复制系统”中的“每个 Replication Server 的初始磁盘分区”。 • 《Replication Server 管理指南第一卷》的“Replication Server 技术概述”的“使用 Replication Server 处理事务”的“稳定队列”中的“稳定队列的分区”。
sqm_cache_enable	设置服务器端稳定队列高速缓存。值为 on 和 off。 缺省值：on
sqm_cache_size	设置服务器端稳定队列高速缓存大小。将页数用单引号或双引号引起来。范围是从 1 到 4096。 缺省值：16
sqm_page_size	以每页包含多少个块的形式设置服务器端稳定队列页大小。将页大小用单引号或双引号引起来。例如，将页大小设置为 4 会指示 Replication Server 以 64K 大块的形式向稳定队列中写入数据。 配置页大小也会设置 Replication Server 的 I/O 大小。范围是 1 到 64。 缺省值：4
sqm_recover_segs	指定 Replication Server 在使用恢复 QID 信息更新 RSSD 之前分配的稳定队列段数。 请参见《Replication Server 管理指南第二卷》的“性能调优”中的“指定分配的稳定队列段数”。 Sybase 建议您增大 **sqm_recover_segs** 的值以提高性能。 缺省值：1 最小值：1 最大值：2,147,483,648

repserver_param	值
sqm_warning_thr1	用于生成第一个警告的分区段（稳定对列空间）的百分比。范围是 1 到 100。 缺省值：75
sqm_warning_thr2	用于生成第二个警告的分区段的百分比。范围是 1 到 100。 缺省值：90
sqm_warning_thr_ind	单个稳定队列用于生成警告的总分区空间的百分比。范围是 51 到 100。 缺省值：70
sqm_write_flush	指定在写操作完成之前是否将写入内存缓冲区中的数据刷新到磁盘。这些值为： • on – 将写入内存缓冲区的数据刷新到磁盘中。 • off – 不将写入内存缓冲区的数据刷新到磁盘中。 • dio – 启用直接 I/O 并允许 Replication Server 在不使用文件系统缓冲的情况下对磁盘执行读写操作。仅适用于 Solaris (SPARC) 和 Linux。 缺省值：on
sqt_init_read_delay	SQT 线程在检查其命令队列中是否有新指令之前等待 SQM 读取时休眠的时间长度。每次到期时，如果命令队列为空，SQT 将加倍其休眠时间，直到达到为 **sqt_max_read_delay** 设置的值。 缺省值：1 毫秒 最小值：0 毫秒 最大值：86,400,000 毫秒（24 小时）
sqt_max_cache_size	最大 SQT（稳定队列事务）接口高速缓存（以字节为单位）。 对于 32 位 Replication Server： • 缺省值 – 1,048,576 • 最小值 – 0 • 最大值 – 2,147,483,647 对于 64 位 Replication Server： • 缺省值 – 20,971,520 • 最小值 – 0 • 最大值 – 2,251,799,813,685,247 如果设置的值大于 2,147,483,647 个字节，则降级会将该值重置为 2,147,483,647 个字节以防止溢出。

repserver_param	值
sqt_max_read_delay	SQT 线程在检查其命令队列中是否有新指令之前等待 SQM 读取时休眠的最大时间长度。 缺省值：1 毫秒 最小值：0 毫秒 最大值：86,400,000 毫秒（24 小时）
sre_reserve	分配给新预订的额外空间大小。例如，100 (100%) 表示与当前的空间相比增加了一倍。范围是 0 到 500。 要更新复制定义的 **sre_reserve** 参数，请直接插入或更新 *rs_config* 系统表。 缺省值：0
stats_reset_rssd	指示 RSSD 是截断上一次采样数据，还是使用新信息将其覆盖。 值：on – 使用新信息覆盖旧采样数据。 off – 保留以前的采样数据。 缺省值：on
stats_sampling	启用采样计数器。 缺省值：off
stats_show_zero_counters	指定 **admin stats** 命令是否报告指定采样周期内观测次数为零的计数器。 值为： • on – 报告观测次数为零的计数器。 • off – 不报告观测次数为零的计数器。 缺省值：off
sts_cachesize	为每个高速缓存的 RSSD 系统表高速缓存的总行数。将此数字增大到活动复制定义的数目，可防止 Replication Server 执行占用资源较多的表查询。 缺省值：1000

repserver_param	值
sts_full_cache_*RSSD_system_table_name*	指定要完全高速缓存的 RSSD 系统表。对于简单的 **select** 语句，完全高速缓存的表不需要访问 RSSD。 输入 **sts_full_cache_**，不要留下空格，后面跟随表名。 **sts_full_cache** 缺省设置为 on 并且 Replication Server 完全高速缓存的表： • rs_clsfunctions • rs_repobjs • rs_users 有关可以完全高速缓存的所有 RSSD 表的列表，请参见《Replication Server 管理指南第二卷》的“性能调优”的“使用调优参数的建议”的“高速缓存系统表”中的“可能被高速缓存的系统表”。
sub_daemon_sleep_time	预订守护程序在醒来恢复预订前的休眠时间，单位为秒。范围是 1 到 31,536,000。 缺省值：120 秒
unicode_format	支持以 `U&"` 格式发送 Unicode 数据。 将 **unicode_format** 设置为下列值之一： • `string` – Unicode 字符转换为字符串格式。例如，`string`“hello”将作为“hello”发送出去。 • `ase` – Unicode 字符以 `U&’’` 格式发送出去。例如，`string`“hello”将作为“U&'\0068\0065\006c\006c\006f’’”发送出去。双字节 Unicode 值按 Adaptive Server Enterprise 所要求的网络顺序发送。 缺省值：`string`
varbinary_strip_trailing_zeros	将 **varbinary_strip_trailing_zeros** 设置为 on 可去除 `varbinary` 值中的尾随零。如果设置为 off，Replication Server 将复制 `varbinary` 值中的尾随零。 无需重新启动 Replication Server 或挂起并恢复连接，即可让参数中的任何更改生效。 缺省值：on
varchar_truncation	在主 Replication Server 或复制 Replication Server 上启用 *varchar* 列截断。如果两个 Replication Server 上都进行字符集转换，则在复制 Replication Server 上设置 **varchar_truncation**。 缺省值：off

- **route_param** – 影响路由。有关路由参数的列表和说明，请参见表 20. 影响路由的配置参数。**configure replication server** 设置所有源自源 Replication Server 的路由的参数值。

- **database_param** – 影响连接。有关连接参数的列表和说明，请参见表 18. 影响数据库连接的参数。**configure replication server** 设置所有源自源 Replication Server 的连接的参数值。
- **logical_database_param** – 影响逻辑连接。有关参数的列表和说明，请参见表 20. 影响路由的配置参数。**configure replication server** 设置所有源自源 Replication Server 的逻辑连接的参数值。
- **password_param** – 影响口令安全性参数。有关参数的列表和说明，请参见表 24. 口令参数。

表 24. 口令参数

password_parameter	说明和*值*
min_password_len	所需的最小字符数。 • 0 – 无最小长度。 • 范围 – 6 到 16（缺省值 6）。
max_password_len	最大字符数。始终将 **max_password_len** 设置为大于 **min_password_len** 的值。 范围 – 13 到 30（缺省值 30）。
password_lowercase_required	是否必需是小写字符。 • True – 必需。 • False – 不是必需的（缺省值）。
password_uppercase_required	是否必需是大写字符。 • True – 必需。 • False – 不是必需的（缺省值）。
password_numeric_required	是否必需是数字字符。 • True – 必需。 • False – 不是必需的（缺省值）。
password_special_char_required	是否必需是特殊字符。 • True – 必需。 • False - 不是必需的（缺省值）。

password_parameter	说明和*值*

simple_passwords_allowed	如果将此选项（或“simple_passwords_allowed”）设置为 false，Replication Server 不允许口令中包含用户名或用户口令字典中的任何值。
	• True – 允许（缺省值）。
	• False – 不允许。
	您可以在 RSSD 中的 `rs_dictionary` 系统表中创建口令字典。该表不存储缺省值。您必须创建自己的脚本以便向该表中插入值。例如：
	`insert into rs_dictionary (words) values ("abcd");`
	`insert into rs_dictionary (words) values ("1234");`

disallowed_prev_passwords	用户更改其口令时不能重复使用的先前口令的数量。 • 0 – 允许先前的口令。 • 范围 – 0 到 32,767（缺省值 0）。 管理员重置口令时，该参数值不应用于用户口令。
password_expiration	口令到期之前经过的天数。 • 0 – 口令永不过期（缺省值）。 • 范围 – 0 到 32,767。 您可以将 **password_expiration** 与 **alter user** 和 **create user** 一起使用。 如果口令已过期，Replication Server 将锁定帐户并通知用户口令已过期。如果用户不重置其口令，则断开连接后，用户将无法登录，直到管理员重置口令。新口令必须满足所有口令要求。 对于 **rs_init** 创建的具有 **connect source** 权限的任何用户或 ID 用户，口令不会过期。这些口令会覆盖您在 Replication Server 中为所有用户设置的任何 **password_expiration** 设置。数据库、其它 Replication Server 和 Replication Agent 使用具有 **connect source** 权限的用户 ID。 管理员应将口令设置为对为 Replication Agent 或 RSI 创建的任何用户不会过期。
initial_password_expiration	初始口令到期之前经过的天数。 • 0 – 初始口令永不过期。 • 范围 – 0 到 32,767（缺省值 0）。 用户的初始口令是在创建用户或重置用户口令时由管理员设置的口令。

password_parameter	说明和*值*
max_failed_logins	Replication Server 在锁定帐户前允许的最大失败登录尝试次数。 • 0 – 帐户永不锁定。 • 范围 – 0 到 32,767（缺省值 0）。 Replication Server 根据 **password_lock_interval** 中设置的时间间隔锁定帐户。
password_lock_interval	用户达到 **max_failed_logins** 中设置的最大登录尝试次数后帐户保持锁定的分钟数。 • 0 – 帐户保持锁定，直到管理员重置口令。 • 范围 – 0 到 32,767（缺省值 0）。
unused_login_expiration	未使用的用户帐户到期之前经过的天数。 • 0 – 未使用的帐户永不到期。 • 范围 – 0 到 32,767（缺省值）。 帐户保持未使用状态的时间长于 **unused_login_expiration** 时，Replication Server 会将其锁定。管理员可以通过重置口令重新激活该帐户。

- **security_param** – 影响基于网络的安全性。请参见表 25. 影响基于网络的安全性的参数。

表 25. 影响基于网络的安全性的参数

security_param	*值*
msg_confidentiality	指示 Replication Server 是否发送和接收加密数据。如果设置为“required”，将对出站数据进行加密。如果设置为“not required”，则无论入站数据是否加密，Replication Server 均会接受。 缺省值：not_required
msg_integrity	指示是否检查数据的篡改情况。 缺省值：not_required
msg_origin_check	指示是否应当检验数据源。 缺省值：not_required
msg_replay_detection	指示是否应当检查数据以确保数据未被截取或重发。 缺省值：not_required
msg_sequence_check	指示是否应当检查数据以确保数据的接收顺序与发送顺序相同。 缺省值：not_required

security_param	*值*
mutual_auth	指示在建立连接之前，远程服务器是否必须提供身份证明。 缺省值：not_required
security_mechanism	为路径启用的第三方安全性机制的名称。 缺省值：在 `libtcl.cfg` 的 SECURITY 部分列出的第一种机制
send_enc_password	确保所有 Replication Server 客户端连接是使用加密口令创建的（与 RSSD 的第一次连接除外）。值为“on”和“off”。 缺省值：off
unified_login	指示 Replication Server 如何尝试登录到远程数据服务器并接受入站登录。 值为： • “required” – 始终尝试使用认证登录到远程服务器。 • “not_required” -- 始终尝试使用口令登录到远程服务器。 缺省值：not_required
use_security_services	指示 Replication Server 是否使用安全服务。如果 **use_security_services** 为“off”，则不使用任何安全性功能。 **注意：** 此参数只能由 **configure replication server** 设置。
use_ssl	指示是否为 Replication Server 启用基于会话的 SSL 安全性。 值为： • “on” – 对 Replication Server 启用 SSL。 • “off” -- 对 Replication Server 不启用 SSL。 缺省值：off

- **id_security_param** – 影响 ID Server 的基于网络的安全性。有关这些参数的列表和说明，请参见表 26. 连接到 ID Server 的安全性参数。

表 26. 连接到 ID Server 的安全性参数

security_param	*值*
id_msg_confidentiality	指示 Replication Server 是否发送和接收加密数据包。如果设置为“required”，将对出站数据进行加密。如果设置为“not required”，则无论入站数据是否加密，Replication Server 均会接受。 缺省值：not required
id_msg_integrity	指示是否检查数据包的篡改情况。 缺省值：not required

security_param	*值*
id_msg_origin_check	指示是否应当检验数据包源。 缺省值：not required
id_msg_replay_detection	指示是否应当检查数据包以确保数据包未被截取或重发。 缺省值：not required
id_ msg_sequence_ check	指示是否应当检查数据包以确保数据包接收顺序与发送顺序相同。 缺省值：not required
id_mutual_auth	要求 ID Server 在 Replication Server 建立连接之前提供身份证明。 缺省值：not required
id_security_mech	指定所支持的安全性机制的名称。 所支持的安全性机制在 `libtcl.cfg` 文件的 SECURITY 下列出。如果未指定名称，Replication Server 将使用缺省机制。 缺省值：列表中的第一个机制
id_unified_login	指示 Replication Server 如何尝试连接到 ID Server。值为： • required – 始终尝试使用认证登录到 ID Server。 • not required – 始终尝试使用口令登录到 ID Server。 **注意：** 如果将 **unified_login** 设置为“required”，则只有“sa”用户能够无需认证而登录到 Replication Server。如果安全机制失败，“sa”用户可以登录并禁用 **unified_login**。 缺省值：not required

- **set security_services [to] 'default'** – 重置所有基于网络的连接安全性功能，使其与 Replication Server 的全局设置相匹配。此命令不会重置 **use_security_services** 功能。

  如果 Replication Server 支持多个安全性机制，则 **set security_services [to] 'default'** 还会将安全性机制设置为缺省值，即 `libtcl.cfg` 文件的 SECURITY 部分中列出的第一种机制。

示例

- **示例 1** – 将 Replication Server 设置为以加密格式发送数据：

```
configure replication server
 set id_msg_confidentiality to 'required'
```

- **示例 2** – 将所有安全性功能设置为与全局设置相匹配：

```
configure replication server
 set security_services to 'default'
```

- **示例 3** – 对于所有源自当前 Replication Server 的路由，将 **rsi_save_interval** 参数更改为 2 分钟：

```
suspend route to each_dest_replication_server

 configure replication server
 set rsi_save_interval to '2'

 resume route to each_dest_replication_server
```

- **示例 4** – 将队列块大小设置为 64。

```
configure replication server
set block_size to ‘64’
```

  （可选）使用 **with shutdown** 子句设置块大小并关闭主 Replication Server

```
configure replication server
set block_size to ‘64’ with shutdown
```

- **示例 5** – 对所有用户强制实施 8 个字符的最小口令长度：

```
configure replication server
set min_password_len to '8'
```

- **示例 6** – 对所有用户强制实施 90 天的口令有效期：

```
configure replication server
set password_expiration to '90'
```

用法

- 每个参数都有两个值：配置值和运行值。Replication Server 在重新启动时使用配置值。运行值是 Replication Server 当前使用的值。启动 Replication Server 时，这两个值是相等的。
- 配置的值将存储在 RSSD 的 *rs_config* 系统表中。
- 如果您使用“**set block_size to ‘block_size’ with shutdown**”Replication Server 参数设置了队列块大小，Replication Server 将会自动关闭。新的块大小会在重新启动 Replication Server 后生效。请参见《Replication Server 管理指南第二卷》的“性能调优”中的“增加队列块大小”。
- **varchar_truncation** 在主 Replication Server 或复制 Replication Server 上启用 *varchar* 列截断。如果传入 *varchar* 数据超出在复制定义中指定的列长度，则会出现以下情况：

表 27. varchar_truncation

	varchar_truncation 在主 Replication Server 设置	**varchar_truncation 在复制 Replication Server 上设置**
varchar_ trunction 设置为“on”	Replication Server 按照复制定义中指定的长度截断入站数据。	Replication Server 按照复制定义中指定的长度截断入站数据。

	varchar_truncation 在主 Replication Server 设置	varchar_truncation 在复制 Replication Server 上设置
varchar_truncation 设置为“off”	RepAgent 在 Replication Server 日志中写入消息，Replication Server 忽略超出复制定义中指定列长度的行。	Replication Server 在 Replication Server 日志中写入消息，并且 DSI 关闭。

- 使用 **ha_failover** 可启用 Sybase 故障切换支持。在发生 ASE 服务器故障切换时，从 Replication Server 到 ASE 的所有连接都会失败。Replication Server 将重试连接。如果将 **ha_failover** 设置为 on，则允许新连接故障切换到新 ASE 服务器。
- 使用 ERSSD 配置参数可配置备份时间、目录位置和 RepAgent 名称。

表 28. ERSSD 配置参数

ERSSD 配置参数	值	Default
erssd_backup_start_time	备份开始的时间。 指定为：“hh:mm AM”或“hh:mm PM”（12 小时制），或“hh:mm”（24 小时制）。	缺省值：01:00 AM
erssd_backup_start_date	备份开始的日期。 以“MM/DD/YYYY”的形式指定。	缺省值：当前日期
erssd_backup_interval	数据库和日志的备份间隔。 指定为“nn 小时”或“nn 分钟”或“nn 秒钟”。	缺省值：24 小时
erssd_backup_dir	存储备份文件的位置。 应该是完整的目录路径。如果配置该路径，则会导致立即进行备份。	缺省值：与事务日志镜像的目录相同；初始值在 **rs_init** 中指定。
erssd_ra	配置 Replication Agent 名称以创建从当前节点到另一个 Replication Server 的路由。此服务器名称必须在接口名称中存在。	*erssd_name*_**ra**

Replication Server 参数

- Replication Server 参数指定影响本地 Replication Server 的缺省值。
- Replication Server 参数是静态参数。必须重新启动 Replication Server 才能使这些参数生效。

路由参数

- 路由参数指定所有源自源 Replication Server 的路由的缺省值。
- 可以覆盖使用 **configure replication server** 指定的缺省值，方法是使用 **alter route** 来分别设置各路由的值。

- 必须首先挂起所有源自当前 Replication Server 的路由，然后才能执行 **configure replication server** 命令。更改参数后，必须恢复所有路由以使更改生效。

数据库参数

- 数据库参数指定所有源自源 Replication Server 的连接的缺省值。
- 可以覆盖使用 **configure replication server** 指定的缺省值，方法是使用 **alter connection** 来分别设置各连接的值。
- 必须首先挂起所有源自当前 Replication Server 的连接，然后才能执行 **configure replication server**。更改参数后，必须恢复所有连接以使更改生效。

逻辑数据库参数

- 逻辑数据库参数指定源自源 Replication Server 的逻辑连接的缺省值。
- 可以覆盖使用 **configure replication server** 指定的缺省值，方法是使用 **configure logical connection** 来设置特定逻辑连接的值。
- 逻辑数据库参数是动态参数。这些参数立即生效。

基于网络的安全性参数

- 除 **use_security_services** 和 **use_ssl** 之外，使用 **configure replication server** 配置的安全性参数是动态参数；这些参数立即生效。
- **use_security_services** 和 **use_ssl** 是静态参数。如果更改这些参数的值，必须重新启动 Replication Server 以使更改生效。
- 用 **configure replication server** 设置的基于网络的缺省安全性参数指定所有与当前 Replication Server 相关联的入站和出站路径的值。
- 可以替换使用 **configure replication server** 指定的缺省安全性设置，方法是使用 **alter route** 或 **alter connection** 重置各出站路径的安全性设置值。
- 如果将 **unified_login** 设置为“required”，则只有“sa”用户可以无需认证而登录到 Replication Server。如果安全性机制失败，“sa”用户可以使用口令登录到 Replication Server，并禁用 **unified_login**。
- Replication Server 可以支持多个安全性机制。所支持的每个机制都在 `libtcl.cfg` 文件的 SECURITY 下列出。
- 路由两端必须使用具有相同安全性机制和安全性设置的兼容安全控制层 (SCL) 驱动程序。复制系统管理员负责为每个服务器选择和设置安全性功能。在尝试建立连接之前，Replication Server 不会查询远程服务器的安全性功能。如果路径两端的安全性功能不兼容，网络连接将失败。
- 消息加密进程占用的资源较多，会造成严重的性能下降。大多数情况下，明智的做法是只将某些路径的 **msg_confidentiality** 设置为“required”。或者，选择占用资源较少的功能（如 **msg_integrity**）来确保安全性。

权限

configure replication server 需要“sa”权限。

另请参见

- admin security_property （第 60 页）
- admin security_setting （第 61 页）
- alter connection （第 109 页）
- alter route （第 161 页）
- configure connection （第 180 页）
- configure route （第 200 页）
- create connection （第 214 页）
- create route （第 273 页）
- set proxy （第 331 页）

configure route

更改从当前 Replication Server 到远程 Replication Server 的路由的属性。

注意： configure route 等同于 **alter route** 命令。

语法

有关语法信息，请参见 **alter route** 命令。

用法

有关用法信息，请参见 **alter route** 命令。

connect

将 Replication Server 变为其 RSSD、ID Server、远程 Replication Server 或远程数据服务器的网关。

语法

```
connect [to] [rssd | idserver | srv_name | ds_name.db_name]
```

参数

- **rssd** – 将 Replication Server 变为其 RSSD 的网关。允许网关使用其配置文件中的 *RSSD_primary_user* 和 *RSSD_primary_pw* 条目。**rssd** 是缺省的 **connect to** 选项。
- **idserver** – 将 Replication Server 变为其 ID Server 的网关，但前提是 Replication Server 本身不是 ID Server。允许网关在配置文件中使用 *ID_user* 和 *ID_pw* 条目。
- **srv_name** – 要将网关连接到的远程 Replication Server 的名称。网关使用 RSI 登录到远程服务器，并需要到远程服务器的直接路由。

- **ds_name.db_name** – 要将网关连接到的远程数据服务器和数据库的名称。Replication Server 网关以维护用户的身份登录到远程数据服务器。这样，您便可以执行指定数据库的维护用户允许执行的任务。但是，您无法访问所连接到的数据服务器中定义的其它数据库。

  Replication Server 网关可以直接连接到 Adaptive Server 以及不需要 Enterprise Connect Data Access (ECDA) 的 Sybase® IQ 数据服务器。对于其它数据服务器，Replication Server 网关必须使用 ECDA 连接到 Replication Server 和远程数据服务器。

示例

- **示例 1** – 通过登录到 ost_replinuxvm_02 并发出 **connection to** 命令，创建从 Replication Server ost_replinuxvm_02 到 RSSD ost_replinuxvm_01.emb 的网关连接：

  ```
  isql -Usa -P -S ost_replinuxvm_02
  1> connect to
  2> go
  ```

  ```
  Gateway connection to 'ost_replinuxvm_01.emb' is created.
  ```

  show server 命令将验证该连接：

  ```
  1> show server
  2> go
  ```

  ```
  ost_replinuxvm_01.emb
  ```

- **示例 2** – 从 Replication Server ost_replinuxvm_02 连接到 Replication Server ost_replinuxvm_03：

  ```
  isql -Usa -P -S ost_replinuxvm_02
  1> connect to ost_replinuxvm_03
  2> go
  ```

  show server 命令将验证该连接：

  ```
  1> show server
  2> go
  ```

  ```
  ost_replinuxvm_03
  ```

- **示例 3** – 创建到 Adaptive Server ost_replinuxvm_01.pdb 的网关连接：

  ```
  isql -Usa -P -S ost_replinuxvm_02
  1> connect to ost_replinuxvm_01.pdb1
  2> go
  ```

  ```
  Gateway connection to 'ost_replinuxvm_01.pdb1' is
  created.
  ```

  ```
  1> select db_name()
  2> go
  ```

  ```
  ------------------------------
  pdb1
  (1 row affected)
  ```

用法

- 在首次登录到 Replication Server 时，发出 **connect** 命令需要使用 **sa** 角色。
- 如果发出 **connect** 命令而不指定选项，则会创建到 RSSD 的网关连接。
- 在充当网关时，Replication Server 使用 RSSD 主用户名和口令登录到 RSSD；使用 ID Server 用户名和口令登录到 ID Server；使用远程服务器标识 (RSI) 登录到远程 Replication Server。在访问 Replication Server 本身时，您无需多次提供此信息。
- 在网关中创建的级联连接将保存在连接堆栈中，并将发出第一个 **connect** 命令的 Replication Server 放在堆栈底部。
- Replication Server 无法直接与其自身连接。不过，可以使用级联连接解决该问题。
- 使用 Replication Server 网关时，客户端和服务器必须使用相同的区域设置集，因为 Replication Server 无法执行字符集转换。

权限

将 Replication Server 转变为网关时需要“sa”权限。

另请参见

- disconnect （第 295 页）
- show connection （第 332 页）
- show server （第 333 页）

create alternate connection

添加替代主连接或复制连接，或替代活动连接或备用连接，并设置连接的配置参数。

语法

```
create alternate connection to data_server.database
named conn_server.conn_db
[set error class [to] error_class]
[set function string class [to] function_class]
[set username [to] user]
[set password [to] passwd]
[set database_param [to] 'value' [set database_param [to]
'value']...]
[set security_param [to] 'value' [set security_param [to]
'value']...]
[with {log transfer on | primary only}]
[as {active | standby} for conn_lds.conn_ldb]
```

参数

- **data_server** – 包含要对其添加替代主连接或复制连接的数据库的数据服务器。
- **database** – 要对其添加替代主连接或复制连接的数据库。

- **conn_server.conn_database** – 替代主连接或复制连接的名称。
  - 对于替代复制连接，如果 *conn_server* 不同于 *dataserver*，则 interface 文件中必须有 *conn_server* 的条目。
  - 如果 *conn_server* 与 *dataserver* 相同，则 *conn_db* 必须不同于 *database*。
  - 每个主连接名在复制系统的所有主连接名中必须是唯一的。每个复制连接名在复制系统的所有复制连接名中必须是唯一的。
  - 为了将替代主连接绑定到替代 Replication Agent 路径以创建主复制路径，*conn_server.conn_db* 必须与 Replication Agent 到 Replication Server 的 Replication Agent 路径的名称相匹配，并且 *conn_server* 必须与 *dataserver* 相同
- **error_class** – 处理数据库错误的错误类。
- **function_class** – 数据库中的操作要使用的函数字符串类。
- **user** – 替代复制连接数据库的 Replication Server 维护用户的登录名。Replication Server 使用此登录名来维护复制数据。如果未启用基于网络的安全性，则必须指定用户名。
- **passwd** – 维护用户登录名的口令。除非启用了基于网络的安全性机制，否则必须指定口令。
- **database_param** – 影响与 Replication Server 的数据库连接的参数。可以使用与用于 **alter connection** 或 **create connection** 的参数相同的参数。
- **value** – 包含选项值的字符串。
- **security_param** – 影响基于网络的安全性的参数。可以使用与用于 **alter connection** 或 **create connection** 的参数相同的参数。
- **log transfer on** – 指示 Replication Server 创建与 *dataserver.database* 中指定的数据的替代主连接和替代复制连接，主替代连接和复制替代连接都具有在 *conn_server.conn_db* 中指定的名称
- **仅主数据库环境** – 指示 Replication Server 仅创建与主数据库的替代主连接，其名称在 *conn_server.conn_db* 指定。
- **as {active | standby} for conn_lds.conn_ldb** – 如果已经创建了名为 *conn_lds.conn_ldb* 的替代逻辑连接，则指示 Replication Server 创建到热备份对的活动数据库或备用数据库的连接

示例

- **示例 1** – 创建到 `SALES_DS` 数据服务器中 `pdb` 数据库的名为 `SALES_DS.pdb_conn2` 的替代主连接：

```
create alternate connection to SALES_DS.pdb
named SALES_DS.pdb_conn2
with primary only
go
```

- **示例 2** – 创建到 `FINANCE_DS` 数据服务器中 `rdb` 复制数据库的名为 `FINANCE_DS2.rdb_conn2` 的替代复制连接：

```
create alternate connection to FINANCE_DS.rdb
named FINANCE_DS2.rdb_conn2
go
```

- **示例 3**

  创建到 IQSRVR Sybase IQ 数据服务器中 lqdb 复制数据库的名为 IQSRVR.lqdb_conn2 的替代复制连接，其中主数据库是 Adaptive Server，dbmaint2 是 IQSRVR.lqdb_conn2 的维护用户：

  ```
  create alternate connection to IQSRVR.iqdb
  named IQSRVR.iqdb_conn2
  using profile rs_ase_to_iq; standard
  set username to dbmaint2
  set password to dbmaint2pwd
  go
  ```

  您也可以创建到可用 Sybase IQ Multiplex 节点的替代连接。确保连接名称是唯一的。

  要创建到 IQSRVR Sybase IQ 节点中 iqbd2 数据库的 iqdb2_conn1 替代连接，请发出以下命令：

  ```
  create alternate connection to IQSRVR.iqbd2
  named IQSRVR2.iqdb2_conn1
  using profile rs_ase_to_iq; standard
  set username to dbmaint3
  set password to dbmaint3pwd
  go
  ```

用法

- **set function string class**、**set username** 和 **set password** 是创建替代连接时可对 **alter connection** 和 **create connection** 使用的现有子句。
  - 如果忽略这些子句，替代复制连接将继承您对缺省复制连接设置的值。
  - 如果在与控制缺省连接的（控制器）Replication Server 不同的（当前）Replication Server 上创建替代连接时忽略这些子句，当前 Replication Server 会返回错误。
  - 仅当同一个 Replication Server 同时控制缺省连接和替代连接时，替代复制连接才可以继承缺省连接中的值。
  - 如果不为替代连接设置维护用户，该连接将继承缺省连接的维护用户。替代连接将使用您为替代连接指定的任何新维护用户。
- **set** *database_param* 和 **set** *security_param* 是可用于指定现有可选连接参数的 **alter connection** 和 **create connection** 的现有子句。
  - 为替代连接设置的任何值都将覆盖从缺省连接继承的值或缺省值。
  - 仅当同一个 Replication Server 同时控制替代连接和缺省连接时，替代连接才可以继承缺省连接中的值。
- 在属于同一复制域并管理数据库的 Replication Server 中执行 **create alternate connection**。

- 要为热备份应用程序创建替代逻辑连接，请使用 **create alternate logical connection** 和 **create alternate connection**。
- 使用 **alter connection** 更改连接的属性。如果已更改维护用户的口令，则使用 **alter connection** 输入新的口令。
- 请参见《Replication Server 管理指南第二卷》的“性能调优”中的“Multi-Path Replication”。

权限

create alternate connection 需要“sa”权限。

另请参见

- admin show_connections （第 66 页）
- alter connection （第 109 页）
- create connection （第 214 页）
- create connection using profile （第 219 页）
- drop connection （第 297 页）

create alternate logical connection

向缺省逻辑连接添加替代逻辑连接。Replication Server 使用逻辑连接来管理热备份应用程序。

语法

```
create alternate logical connection to logical_ds.logical_db
named conn_lds.conn_ldb
```

参数

- **logical_ds** – 缺省逻辑连接的逻辑数据服务器的名称。
- **logical_db** – 缺省逻辑连接的逻辑数据库的名称。
- **conn_lds.conn_ldb** – 替代逻辑连接的名称。
  - 如果 *conn_lds* 与 *logical_ds* 相同，则 *conn_ldb* 必须不同于 *logical_db*。
  - 每个替代逻辑连接名 (*conn_lds.conn_ldb*) 在复制系统的所有替代逻辑连接名中必须是唯一的。

示例

- **示例 1** – 创建到 LDS 逻辑数据服务器中 `logicaldb` 逻辑数据库的名为 *lds.conn_logicaldb2* 的替代逻辑连接：

```
create alternate logical connection to LDS.logicaldb
named lds.conn_logicaldb2
```

用法

- 必须先使用 **create logical connection** 创建缺省逻辑连接，然后才能创建替代逻辑连接。请参见《Replication Server 管理指南第二卷》的“管理热备份应用程序”的“Setting Up ASE Warm Standby Databases”（设置 ASE 热备份数据库）中的“任务一：创建逻辑连接”。
- 要创建和配置替代逻辑连接，请参见《Replication Server 管理指南第二卷》的“性能调优”的“Multi-Path Replication”中的“Multiple Replication Paths for Warm Standby Environments”（用于热备份环境的多个复制路径）。
- 可以使用不同的 Replication Server 来控制缺省逻辑连接和替代逻辑连接。活动数据库和备用数据库都必须支持具有多个 Replication Agent。
- 创建替代逻辑连接后，可以使用 **create alternate connection** 创建到活动数据库和备用数据库的替代连接。
- 复制定义和预订使用缺省逻辑连接名称。

权限

create alternate logical connection 需要“sa”权限。

另请参见

- create alternate connection （第 202 页）
- create logical connection （第 252 页）

create applied function replication definition

为要复制的存储过程创建应用函数复制定义和用户定义函数。应用函数由维护用户在复制数据库中应用。

语法

```
create applied function replication definition repdef_name
        with primary at dataserver.database
        [with all functions named 'func_name' |
        [[with primary function named 'func_name']
        [with replicate function named 'func_name']]]
        ([@param_name datatype [, @param_name datatype]…])
        [searchable parameters (@param_name [, @param_name]…)]
        [send standby {all | replication definition} parameters]
```

参数

- **repdef_name** – 应用函数复制定义名称。名称必须符合标识符规则。

- **with primary at** – 指定主数据服务器和主数据库。
- **dataserver** – 包含主数据的数据服务器的名称。如果主数据库是热备份应用程序的一部分，则 *dataserver* 是逻辑数据服务器名称。
- **database** – 包含主数据的数据库的名称。如果主数据库是热备份应用程序的一部分，则 *database* 是逻辑数据库名称。
- **with all functions named** – 指定主数据库和复制数据库中的存储过程名称。
- **'func_name'** – 函数名称。*func_name* 是最多包含 255 个字符的字符串。
- **with primary function named** – 指定主数据库中的存储过程名称。使用 **with primary function named**，可以为主函数指定一个不同于复制定义名称的名称。如果不指定主函数名称，Replication Server 会使用复制定义名称作为主函数的名称。
- **with replicate function named** – 指定要在复制数据库中执行的存储过程的名称。如果不指定复制函数名称，Replication Server 会使用复制定义名称作为复制函数的名称。
- **@param_name** – 函数的参数名。参数名在每个子句中最多只能出现一次。不要求包括参数及其数据类型，但是不管是否包括参数，都必须 包括一对小括号。
- **数据类型** – 函数中参数的数据类型。有关数据类型及其语法的列表，请参见“数据类型”。Adaptive Server 存储过程和函数复制定义不能包含数据类型为 *text*、*unitext*、*rawobject* 和 *image* 的参数。
- **searchable parameters** – 指定可在 **define subscription**、**create subscription** 或 **create article** 的 **where** 子句中使用的参数的列表。如果包括 **searchable parameters** 子句，则必须包括一对小括号。
- **send standby** – 在热备份应用程序中，指定是将此函数中的所有参数都发送到备用数据库 (**send standby all parameters**)，还是只将复制定义中指定的参数发送到备用数据库 (**send standby replication definition parameters**)。缺省值是 **send standby all parameters**。

示例

- **示例 1** – 为同名的函数创建名为 **titles_frep** 的应用函数复制定义。主数据位于 *LDS* 数据服务器的 *pubs2* 数据库中：

```
create applied function replication definition titles_frep
with primary at LDS.pubs2
(@title_id varchar(6), @title varchar(80), @type char(12), @pub_id
char(4),
     @price money, @advance money, @total_sales int)
searchable parameters (@title_id, @title)
```

- **示例 2** – 为同名的函数创建名为 **titles_frep** 的应用函数复制定义。存储过程在复制数据库中命名为 **upd_titles**：

```
create applied function replication definition titles_frep
with primary at LDS.pubs2
with replicate function named 'upd_titles'
(@title_id varchar(6), @title varchar(80), @type char(12), @pub_id
char(4),
```

```
     @price money, @advance money, @total_sales int)
searchable parameters (@title_id, @title)
```

- **示例 3** – 为名为 **upd_titles_prim** 的函数创建名为 **titles_frep** 的应用函数复制定义。存储过程在主数据库中命名为 **upd_titles_prim**，在复制数据库中命名为 **upd_titles**：

```
create applied function replication definition titles_frep
with primary at LDS.pubs2
with primary function named 'upd_titles_prim'
with replicate function named 'upd_titles'
(@title_id varchar(6), @title varchar(80), @type char(12), @pub_id
char(4),
     @price money, @advance money, @total_sales int)
searchable parameters (@title_id, @title)
```

用法

- 使用 **create applied function replication definition** 可描述要复制的存储过程。应用函数复制定义与请求函数复制定义之间的区别是：通过应用函数复制定义复制的函数由维护用户在复制节点上执行，而通过请求函数复制定义复制的函数由在主节点上执行主函数的同一用户在复制节点上执行。有关复制存储过程的概述，请参见《Replication Server 管理指南第一卷》。
- 为主函数创建应用函数复制定义时，请确保此函数中没有同时满足以下两个条件的现有函数复制定义：
  - 函数复制定义是使用 **create function replication definition** 命令创建的
  - 未在 Replication Server 15.0.1 和更低版本中预订函数复制定义，便将函数复制定义用于请求函数复制

  如果满足这两个条件，则会禁用现有请求函数复制定义。有关 Replication Server 15.0.1 和更低版本中的应用函数复制定义的详细信息，请参见《Replication Server 管理指南第二卷》。
- 在对存储主数据的数据库进行管理的 Replication Server 上执行 **create applied function replication definition**。
- 执行 **create applied function replication definition** 之前，请确保满足以下条件：
  - 函数复制定义名称在复制系统中是唯一的。Replication Server 无法始终在您使用 **create applied function replication definition** 时强制执行此要求。
  - Replication Server 和主数据库之间已建立连接。请参见 **create connection**。也可以使用 **rs_init** 来创建连接。有关详细信息，请参见适用于您所用平台的《Replication Server 安装指南》和《Replication Server 配置指南》。
  - 为函数复制定义指定的名称、参数和数据类型必须与所涉及的存储过程的名称、参数和数据类型相匹配。只有在函数复制定义中指定的参数才会得到复制。
- 要更新表，需要与表复制定义关联的复制存储过程，而不需要与函数复制定义关联的复制存储过程。这样，您就可以复制与复制数据不相关的事务。

  有关系统存储过程的详细信息，请参见“RSSD 存储过程”。有关两种复制存储过程类型的详细信息，请参见 **sp_setrepproc**。

- Replication Server 通过复制系统将新的函数复制定义分发到合格的节点。由于普通复制系统会有一定时间的滞后，因此更改不会立即显示在所有合格的节点上。

用户定义的函数和函数字符串

- 创建应用函数复制定义时，Replication Server 会自动创建一个相应的用户定义的函数。同样，在 **rs_sqlserver_function_class** 中，Replication Server 会自动为用户定义的函数创建一个缺省函数字符串。
- 您可以使用 **create function string** 自定义 **rs_sqlserver_function_class** 中和用户定义的函数字符串类中的函数字符串。
- 对于用户定义的每个基本函数字符串类（将在其中使用用户定义的函数）以及从这些类继承的每个派生类，请使用 **create function string** 来创建函数字符串。函数字符串应当使用适合于复制数据服务器的语言来调用存储过程或 RPC。
- 有关函数字符串类、函数字符串和函数的概述，请参见《Replication Server 管理指南第二卷》。

with primary at 子句

使用 **with primary at** 子句可指定主数据服务器和主数据库。主数据库是指包含调用的存储过程的数据库。

with replicate function named 子句

使用 **with replicate function named** 子句可指定要在复制数据库中执行的存储过程的名称。如果创建或更改函数复制定义时不使用 **with replicate function named**，则会将函数作为与函数复制定义同名的存储过程进行传递。在热备份数据库和活动数据库中，存储过程具有相同的名称，因而会忽略 **with replicate function named**。

利用往返复制，一个数据库可以向另一个数据库发送数据更改请求并将数据更改复制回请求数据库中。有关如何使用应用函数复制定义和请求函数复制定义设置往返复制的详细信息，请参见《Replication Server 管理指南第一卷》。

HDS 参数的应用函数复制定义

虽然无法创建可更改参数值数据类型的函数复制定义，但可以使用 HDS 数据类型定义声明应用函数复制定义的参数。声明的参数将受到类级别转换的限制。

有关 HDS 的详细信息，请参见《Replication Server 管理指南第一卷》。

更改函数复制定义

- 使用 **alter applied function replication definition** 可将参数或可搜索参数添加到现有应用函数复制定义中。您也可以为函数指定另一个复制名称。
- 要删除或重命名函数复制定义中的参数，请删除对函数复制定义的所有预订。删除预订后，请删除函数复制定义，然后再重新创建该定义。

预订函数复制定义

要预订应用函数复制定义，请使用带有 **without materialization** 子句的 **create subscription**，或使用 **define subscription** 以及其它涉及批量实现的命令。

函数复制定义和表复制定义

- 使用应用函数复制存储过程时，请为受复制存储过程影响的表创建表复制定义和预订。这样做可以确保影响表的正常事务以及存储过程执行均会得到复制。但是，如果 DML 所在的存储过程被标记为“已复制”，则不会复制该 DML。在这种情况下，即使已经预订表，也要预订该存储过程。
- 如果准备将函数复制定义和表复制定义用于同一个表，可以使用表复制定义的预订来实现表数据。然后，可以使用带有 **without materialization** 子句的 **create subscription** 创建函数复制定义的预订。

创建多个复制定义

- 您可以为同一主函数创建多个应用函数复制定义，并对每一个应用函数复制定义进行自定义，以便不同的复制函数可以对其进行预订。有关详细信息，请参见《Replication Server 管理指南第一卷》。
- 为同一主函数创建的不同应用函数复制定义必须 使用同名的相同参数以及相同的数据类型。
- 如果应用函数复制定义为复制定义和主函数指定的名称不同，则只有 Replication Server 15.1 版或更高版本才可以预订该定义。
- 同一主函数可以拥有应用函数复制定义或请求函数复制定义，但不能同时拥有这两种定义。在创建函数复制定义的主 Replication Server 中，会将使用 **create function replication definition** 命令创建的函数复制定义视为应用函数。
- 在热备份数据库和活动数据库中，存储过程具有相同的名称，因而会忽略 **with replicate function** 子句。如果使用 **send standby replication definition parameters** 子句创建某一个应用函数复制定义，则会将在函数复制定义中指定的参数传递到备用数据库中。否则，将会传递主函数中的所有参数。
- 在 MSA 环境中，如果使用 **send standby** 子句创建的主函数没有函数复制定义，则传递到复制数据库的函数与主函数同名，且具有主函数的所有参数。否则，传递到复制数据库的函数会使用在函数复制定义的 **with replicate function named** 子句中指定的名称，并包括在同一函数复制定义中指定的参数。

权限

create applied function replication definition 需要“create object”权限。

另请参见

- alter function string （第 143 页）
- alter applied function replication definition （第 107 页）
- alter request function replication definition （第 159 页）
- create connection （第 214 页）
- create function string （第 236 页）
- create request function replication definition （第 269 页）
- define subscription （第 290 页）
- drop function replication definition （第 301 页）

- sp_setrepproc （第 491 页）
- rs_send_repserver_cmd （第 546 页）

create article

创建表或函数复制定义的项目，并指定要包含此项目的发布。

语法

```
create article article_name
        for pub_name
with primary at data_server.database
with replication definition {table_rep_def | function_rep_def}
        [where {column_name | @param_name}
                {< | > | >= | <= | = | &} value
        [and {column_name | @param_name}
                {< | > | >= | <= | = | &} value]...
        [or where {column_name | @param_name}
                {< | > | >= | <= | = | &} value
        [and {column_name | @param_name}
                {< | > | >= | <= | = | &} value]...]...]
```

参数

- **article_name** – 项目的名称。此名称必须符合标识符规则，并且在发布中是唯一的。
- **for pub_name** – 包含项目的发布的名称。
- **with primary at data_server.database** – 指定主数据的位置。如果主数据库是热备份应用程序的一部分，则 *data_server.database* 是逻辑数据服务器和数据库的名称。
- **with replication definition table_rep_def** – 指定项目的表复制定义的名称。
- **with replication definition function_rep_def** – 指定项目的函数复制定义的名称。
- **where** – 设置要通过对包含此项目的发布的预订进行复制的列或参数值的标准。如果未包括 **where** 子句，则复制所有行或参数。

  where 子句由一个或多个简单的比较组成，使用以下关系运算符之一对可搜索列或可搜索参数和某个实际值进行比较：**<**、**>**、**<=**、**>=**、**=** 或 **&**。（只有 *rs_address* 列或参数支持 **&** 运算符。）可以使用关键字 **and** 来连接比较。

  where 子句中使用的列或参数名称必须也包括在表复制定义的 **searchable columns** 列表或函数复制定义的 **searchable parameters** 列表中。

  可以在项目中包括多个 **where** 子句，这些子句由关键字 **or** 隔开。

  项目中的 **where** 子句的最大大小为 255 个字符。
- **column_name** – 主表的列名，用于包含表复制定义的项目。
- **@param_name** – 复制存储过程的参数名，用于包含函数复制定义的项目。
- **value** – 指定列或参数的值。有关不同数据类型值的输入格式，请参见“数据类型”。

表达式中使用的列或参数名称必须包括在复制定义的 **searchable columns** 或 **searchable parameters** 列表中。

示例

- **示例 1** – 根据复制定义 *titles_rep*，为发布 *pubs2_pub* 创建名为 *titles_art* 的项目：

```
create article titles_art
 for publication pubs2_pub
 with primary at TOKYO_DS.pubs2
 with replication definition titles_rep
```

- **示例 2** – 如上例所示，为发布 *pubs2_pub* 创建名为 *titles_art* 的项目。此命令包括 **where** 子句，此子句只复制畅销计算机书籍的行，畅销计算机书籍的 *type* 列设置为“popular_comp”：

```
create article titles_art
 for publication pubs2_pub
 with primary at TOKYO_DS.pubs2
 with replication definition titles_rep
 where type = 'popular_comp'
```

- **示例 3** – 如上例所示，为发布 *pubs2_pub* 创建名为 *titles_art* 的项目。此命令包括两个 **where** 子句，它们共同复制畅销计算机书籍和传统菜谱的行：

```
create article titles_art
 for publication pubs2_pub
 with primary at TOKYO_DS.pubs2
 with replication definition titles_rep
 where type = 'popular_comp'
 or where type = 'trad_cook'
```

用法

- 使用 **create article** 指定要使用指定的发布复制数据的复制定义。可选的 **where** 子句有助于确定要复制的数据。
- 在管理存储主数据的数据库的 Replication Server 上执行 **create article**。
- 使用 **create article** 自动使项目的发布失效。在验证发布之前，无法创建新的预订。在刷新预订之前，无法复制新项目的数据。
- 有关使用复制定义、项目和发布的详细信息，请参见《Replication Server 管理指南第一卷》。

  有关预订发布的详细信息，请参见《Replication Server 管理指南第一卷》中的“管理预订”。
- 仅在为发布创建或刷新预订时，Replication Server 才会将与发布及其项目有关的信息分发到复制节点。

使用 **create article** 的要求

- 执行 **create article** 之前，请确保满足以下两个条件：
  - 要为其创建项目的发布已经存在。

- 项目的复制定义已经存在。

将项目添加到新的发布

- 创建发布后，使用 **create article** 创建项目并将它们指派给此发布。项目指定表复制定义（或函数复制定义）以及父发布。根据预订的复制节点的需要，还可以选择包括 **where** 子句。
  在验证发布及为发布创建预订之前，此发布中必须至少包含一个项目。有关详细信息，请参见 **create publication** 命令。

项目和预订

- 创建发布的预订时，Replication Server 将为其中的每个项目都创建一个内部预订。
- 如果包括某个项目的多个 **where** 子句（由 **or** 关键字分隔），则可以消除 Replication Server 的限制，此限制只允许每个预订包含一个 **where** 子句。发布预订不能包括 **where** 子句，而是通过在项目中使用 **where** 子句来代替。

将项目添加到具有预订的发布

- 如果将新项目添加到现有发布，或从现有发布中删除项目，则此发布将失效。虽然现有项目的复制仍不受影响，但要开始新项目的复制，必须执行以下操作：
  - 在完成对发布的更改后，验证此发布，然后
  - 刷新此发布预订。

  有关刷新发布预订的两种方法的详细信息，请参见 **create subscription** 命令和 **define subscription** 命令。另请参见 **validate publication** 命令。

权限

create article 需要“create object”权限。

另请参见

- check publication （第 176 页）
- create applied function replication definition （第 206 页）
- create publication （第 254 页）
- create replication definition （第 258 页）
- create request function replication definition （第 269 页）
- create subscription （第 280 页）
- define subscription （第 290 页）
- drop article （第 296 页）
- drop publication （第 307 页）
- validate publication （第 389 页）

create connection

将数据库添加到复制系统，并设置此连接的配置参数。要创建 Adaptive Server 数据库的连接，请使用 Sybase Central 或 **rs_init**。若要为非 Adaptive Server 数据库创建连接，请参见 **create connection using profile** 命令。

语法

```
create connection to data_server.database
set error class [to] error_class
set function string class [to] function_class
set username [to] user
[set password [to] passwd]
[set replication server error class [to] rs_error_class]
[set database_param [to] 'value' [set database_param [to]
'value']...]
[set security_param [to] 'value' [set security_param [to]
'value']...]
[with {log transfer on, dsi_suspended}]
[as active for logical_ds.logical_db |
as standby for logical_ds.logical_db
[use dump marker]]
```

参数

- **data_server** – 存放要添加到复制系统的数据库的数据服务器。
- **database** – 要添加到复制系统的数据库。
- **error_class** – 处理数据库错误的错误类。
- **function_class** – 数据库中的操作要使用的函数字符串类。
- **user** – 数据库的 Replication Server 维护用户的登录名。Replication Server 使用此登录名来维护复制数据。如果未启用基于网络的安全性，则必须指定用户名。
- **passwd** – 维护用户登录名的口令。除非启用了基于网络的安全性机制，否则必须指定口令。
- **rs_error_class** – 用于处理数据库的 Replication Server 错误的错误类。缺省值为 **rs_repserver_error_class**。
- **database_param** – 影响与 Replication Server 的数据库连接的参数。参数和值的说明详见表 18. 影响数据库连接的参数。
- **value** – 包含选项值的字符串。
- **security_param** – 影响基于网络的安全性的参数。有关可以使用 **create connection** 设置的安全性参数的列表和说明，请参见“影响基于网络的安全性的参数”表。
- **log transfer on** – 指示连接可以是主数据源或复制函数源。如果存在此子句，Replication Server 将创建入站队列，准备接受数据库的 RepAgent 连接。如果忽略此选项，连接无法接受来自 RepAgent 的输入。

- **dsi_suspended** – 在 DSI 线程挂起的情况下启动连接。可稍后恢复 DSI。如果要连接到不支持 Replication Server 连接的非 Sybase 数据服务器，此选项非常有用。
- **as active for** – 指示此连接是与某个逻辑连接的活动数据库的物理连接。
- **as standby for** – 表明此连接是与某个逻辑连接的备用数据库的物理连接。
- **logical_ds** – 逻辑连接的数据服务器名称。
- **logical_db** – 逻辑连接的数据库名称。
- **use dump marker** – 指示 Replication Server 在启用来自活动数据库的事务流中的复制标志，进而接收到第一个 dump marker 后，将事务应用于备用数据库。如果没有此选项，Replication Server 将在启用复制标记后应用接收的事务。

注意： 如果在 MSA 复制中使用跨平台 dump 和 load (XPDL) 功能，则不要对实现使用 **use dump marker** 子句。

示例

- **示例 1** – 在 SYDNEY_DS 数据服务器中创建 *pubs2* 数据库的连接。Replication Server 将使用 *ansi_error* 错误类来处理数据库的错误。使用 *sqlserver_derived_class* 函数字符串类中的函数字符串进行数据处理操作。连接将使用 *pubs2_maint_ps* 登录名和 *pubs2_maint* 口令登录到 *pubs2* 数据库：

```
create connection to SYDNEY_DS.pubs2
 set error class ansi_error
 set function string class sqlserver_derived_class
 set username pubs2_maint
 set password pubs2_maint_pw
```

- **示例 2** – 创建与第一个示例类似的连接。不过，在此示例中，Replication Server 错误类 *tokyo_rs_error* 处理连接的 Replication Server 错误并指定 **with log transfer** 子句。这允许此连接接受来自 RepAgent 的输入。此连接是与包含主数据或将作为复制函数源的数据库的连接：

```
create connection to TOKYO_DS.pubs2
set error class ansi_error
set function string class sqlserver_derived_class
set username pubs2_maint
set password pubs2_maint_pw
set replication server error class tokyo_rs_error
with log transfer on
```

用法

- 使用 **create connection** 将数据库添加到复制系统。通常，使用此命令可添加到非 Sybase 数据库的连接。要创建与 Adaptive Server 数据库的标准连接，可使用 Sybase Central 或 **rs_init**。
- 要创建使用异构数据类型支持 (HDS) 将主数据库的数据类型转换为复制数据库的数据类型的连接，还可以使用 Sybase 提供的脚本，创建连接并安装 HDS。有关说明，请参见适用于您的平台的《Replication Server 配置指南》。

- 在管理数据库的 Replication Server 上执行 **create connection**。
- Replication Server 通过复制系统将数据库连接信息分发到合格的节点。由于通常复制系统会有一定时间的滞后，因此更改不会立即显示在所有这些节点上。
- 必须指定错误类，即便使用缺省错误类：*rs_sqlserver_error_class*。
- 您不必指定 Replication Server 错误类，除非它是新的 Replication Server 错误类。Replication Server 的缺省错误类是 *rs_repserver_error_class*。
- 每个数据库只允许一个连接。在 ID Server 中，强制实施这一限制，将在 *rs_idnames* 系统表中注册每个数据库。创建数据库的连接时，ID Server 必须在可用。
- 使用 **set function string class [to]** *function_class* 可激活非 Sybase 数据服务器的类级别转换。

数据库连接参数

- Replication Server 配置参数存储在 *rs_config* 系统表中。有关 *rs_config* 系统表中的数据库连接参数的详细信息，请参见《Replication Server 管理指南第一卷》。
- 有关配置并行 DSI 线程的详细信息，请参见《Replication Server 管理指南第二卷》。
- 使用 **assign action** 允许对由于特定数据服务器错误而导致失败的事务进行重试。

dump_load 配置参数

- 将 **dump_load** 设置为“on”之前，先创建 **rs_dumpdb** 和 **rs_dumptran** 函数的函数字符串。Replication Server 不会在系统提供的类或从这些类继承的派生类中生成这些函数的函数字符串。

save_interval 配置参数

- 设置 **save_interval** 以将事务保存在 DSI 队列中，以便在从备份中恢复数据库后进行重新同步时使用。如果设置存放复制数据或接收复制函数的数据库的热备份，设置保存间隔也非常有用。可以使用 **sysadmin restore_dsi_saved_segments** 来恢复积压的事务。

错误类和函数类

- 表 29. 错误类和函数类 显示了 Replication Server 为 Replication Server 和数据库连接提供的错误类和函数类。

表 29. 错误类和函数类

类名	说明
rs_repserver_error_class	Replication Server 的错误操作指派。
rs_sqlserver_error_class	Adaptive Server 数据库的错误操作指派。

类名	说明
rs_sqlserver_function_class	Adaptive Server 数据库的函数字符串类。不能参与函数字符串继承。Replication Server 自动生成函数字符串。
rs_default_function_class	Adaptive Server 数据库的函数字符串类。不能修改函数字符串。可以将此类指定为父类，但不能指定为派生类。Replication Server 自动生成函数字符串。
rs_db2_error_class	DB2 数据库的错误类。
rs_db2_function_class	DB2 数据库的函数字符串类。不能修改函数字符串。可以将此类指定为父类，但不能指定为派生类。Replication Server 自动生成函数字符串。
rs_iq_error_class	Sybase IQ 数据库的错误类。
rs_iq_function_class	Sybase IQ Oracle 数据库的函数字符串类。不能修改函数字符串。可以将此类指定为父类，但其派生类不能从父类继承任何类级别转换。Replication Server 自动生成函数字符串。
rs_msss_error_class	Microsoft SQL Server 数据库的错误类。
rs_ms_function_class	Microsoft SQL Server 数据库的函数字符串类。不能修改函数字符串。可以将此类指定为父类，但其派生类不能从父类继承任何类级别转换。Replication Server 自动生成函数字符串。
rs_oracle_error_class	Oracle 数据库的错误类。
rs_oracle_function_class	Oracle 数据库的函数字符串类。不能修改函数字符串。可以将此类指定为父类，但其派生类不能从父类继承任何类级别转换。Replication Server 自动生成函数字符串。
rs_udb_error_class	UDB 数据库的错误类。
rs_udb_function_class	UDB 数据库的函数字符串类。不能修改函数字符串。可以将此类指定为父类，但其派生类不能从父类继承任何类级别转换。Replication Server 自动生成函数字符串。

注意： **rs_dumpdb** 和 **rs_dumptran** 系统函数未进行初始定义，即使 Replication Server 在其中生成缺省函数字符串的函数字符串类也是如此。如果要使用协调的转储，必须创建这些函数的函数字符串。另请注意，无法在备用数据库上执行协调的转储。有关使用函数字符串的详细信息，请参见《Replication Server 管理指南第二卷》。有关 **rs_dumpdb** 和 **rs_dumptran** 函数的详细信息，请参见“Replication Server 系统函数”。

用户名和口令

- 创建连接时指定维护用户的登录名和口令。必须授予维护用户登录名维护数据库中复制数据所有必要的权限。

注意： 如果复制系统中有两个节点具有相同的数据库名称，维护用户登录名必须不同。由 Sybase Central 或 **rs_init** 创建的缺省登录名是 *DB_name*_maint。对系统进行设置时，更改其中一个登录名以使每个名称互不相同。

热备份应用程序

- 要创建热备份应用程序的逻辑连接，请使用 **create logical connection**。
- 在热备份应用程序中，活动数据库和备用数据库的连接必须具有 **log transfer on**。
- 只有在数据库为活动数据库时，才会使用热备份应用程序中此数据库的函数字符串类。Replication Server 对备用数据库使用 *rs_default_function_class*。

更改连接属性

- 使用 **alter connection** 更改连接的属性。
- 如果已更改维护用户的口令，则使用 **alter connection** 输入新的口令。

基于网络的安全性参数

- 连接两端必须使用具有相同安全性机制和安全性功能的兼容安全性控制层 (SCL) 驱动程序。远程服务器必须支持 **set proxy** 或等效的命令。复制系统管理员负责为每个服务器选择和设置安全性功能。在尝试建立连接之前，Replication Server 不会查询远程服务器的安全性功能。如果连接两端的安全性功能不兼容，连接将失败。
- **create connection** 指定从 Replication Server 到目标数据服务器的出站连接的安全性设置。**create connection** 所设置的安全性功能覆盖由 **configure replication server** 设置的安全性功能。
- 如果将 **unified_login** 设置为“required”，只有具有“sa”权限的复制系统管理员才可以在没有凭据的情况下登录到 Replication Server。如果安全性机制失败，复制系统管理员可以使用口令登录到 Replication Server 并禁用 **unified_login**。
- Replication Server 可具有多种安全性机制；所支持的每种机制都在 `libtcl.cfg` 文件中的 SECURITY 下列出。
- 消息加密进程占用的资源较多，会造成严重的性能下降。大多数情况下，明智的做法是只将某些连接的 **msg_confidentiality** 设置为“required”。或者，选择占用资源较少的安全性功能，如 **msg_integrity**。

权限

create connection 需要“sa”权限。

另请参见

- admin show_connection_profiles （第 63 页）
- alter connection （第 109 页）
- create alternate connection （第 202 页）
- create connection using profile （第 219 页）
- configure connection （第 180 页）
- create error class （第 228 页）
- create function string class （第 249 页）
- create logical connection （第 252 页）

- alter route （第 161 页）
- drop connection （第 297 页）
- resume connection （第 321 页）
- rs_classes （第 574 页）
- rs_profdetail （第 605 页）
- rs_profile （第 605 页）
- rs_systext （第 623 页）
- suspend connection （第 334 页）

create connection using profile

create connection using profile 子句使用预定义的信息配置 Replication Server 和非 Adaptive Server 数据库之间的连接；如果需要，还会修改 RSSD 和命名的 *data_server.database*。若要创建到 Adaptive Server 的连接，请参见 **create connection**。

语法

```
create connection to data_server.database
using profile connection_profile;version
set username [to] user
[other_create_connection_options]
[display_only]
```

参数

- **data_server** – 存放要添加到复制系统的数据库的数据服务器。
- **database** – 要添加到复制系统的数据库。
- **connection_profile** – 指示要用于配置连接、修改 RSSD 和生成复制数据库对象的连接配置文件。
- **version** – 指定要使用的连接配置文件版本。
- **user** – 数据库的 Replication Server 维护用户的登录名。Replication Server 使用此登录名来维护复制数据。如果未启用基于网络的安全性，则必须指定用户名。
- **other_create_connection_options** – 可以使用其它 **create connection** 选项设置配置文件中未指定的连接选项（如设置口令）或覆盖配置文件中指定的选项（如指定自定义函数字符串类以覆盖 Replication Server 中提供的函数字符串类）。有关其它 **create connection** 选项的完整列表，请参见 **create connection**。
- **display_only** – 可以使用带有 **using profile** 子句的 **display_only** 显示将要执行的命令以及在上面执行这些命令的服务器的名称。有关使用 **display_only** 的结果，请参见客户端和 Replication Server 日志。

示例

- **示例 1** – 创建 Oracle 复制数据库的连接：

```
create connection to oracle.instance
using profile rs_ase_to_oracle;standard
set username to ora_maint
set password to ora_maint_pwd
```

- **示例 2** – 创建 Microsoft SQL Server 复制数据库（同时也是主数据库）的连接。在本示例中，以下命令使用 *my_msss_error_class* 错误类替换连接配置文件提供的任何错误类设置：

```
create connection to msss_server.msss_db
using profile rs_ase_to_msss
set username to msss_maint;standard
set password to msss_maint_pwd
set error class to my_msss_error_class
with log transfer on
```

- **示例 3** – 使用特定版本的配置文件 v9_1 创建 DB2 复制数据库。在本示例中，以下命令使用新值 (16384) 覆盖连接配置文件提供的命令批处理大小：

```
create connection to db2.subsys
using profile rs_ase_to_db2;v9_1
set username to db2_maint
set password to db2_maint_pwd
set dsi_cmd_batch_size to '16384'
```

- **示例 4** – 使用 **display_only** 选项显示您使用特定配置文件将要执行的命令。这些命令和命令输出显示在屏幕上，并且还会将其写入到 Replication Server 日志中：

```
create connection to oracle.instance
using profile rs_ase_to_oracle;standard
set username to ora_maint
set password to ora_maint_pwd
display_only

go
```

```
Display only using Connection Profile rs_ase_to_oracle;standard.

Command(s) intended for: prs01
create connection to oracle.instance
   set error class to rs_oracle_error_class
   set function string class to rs_oracle_function_class
   set username to ora_maint
   set password to ********
   set batch to off

Command(s) intended for 'edsprs01.edbprs01':
delete from rs_translation where classid = 0x0000000001000007 and
                                 source_dtid = 0x000000000000000c

Command(s) intended for 'edsprs01.edbprs01':
insert rs_translation (prsid, classid, type, source_dtid,
target_dtid,
                       target_length, target_status, rowtype)
values (0, 0x0000000001000007, 'D', 0x000000000000000c,
0x0000000000010200,
        19, 0, 0)
```

```
Command(s) intended for 'edsprs01.edbprs01':
delete from rs_translation where classid = 0x0000000001000007  and
                                source_dtid = 0x000000000000000d

Command(s) intended for 'edsprs01.edbprs01':
insert rs_translation (prsid, classid, type, source_dtid,
target_dtid,
                       target_length, target_status, rowtype)
values (0, 0x0000000001000007, 'D', 0x000000000000000d,
0x0000000000010200,
        19, 0, 0)

Command(s) intended for 'edsprs01.edbprs01':
delete from rs_translation where classid = 0x0000000001000007 and
                                source_dtid = 0x0000000000000001

Command(s) intended for 'edsprs01.edbprs01':
insert rs_translation (prsid, classid, type, source_dtid,
target_dtid,
                       target_length, target_status, rowtype)
values (0, 0x0000000001000007, 'D', 0x0000000000000001,
0x0000000000010202,
        0, 0, 0)
Command(s) intended for 'edsprs01.edbprs01':
delete from rs_translation where classid = 0x0000000001000007 and
                                source_dtid = 0x0000000000000013

Command(s) intended for 'edsprs01.edbprs01':
insert rs_translation (prsid, classid, type, source_dtid,
target_dtid,
                       target_length, target_status, rowtype)
values (0, 0x0000000001000007, 'D', 0x0000000000000013,
0x0000000000010202,
        0, 0, 0)

Command(s) intended for 'edsprs01.edbprs01':
delete from rs_translation where classid = 0x0000000001000007 and
                                source_dtid = 0x000000000000000E

Command(s) intended for 'edsprs01.edbprs01':
insert rs_translation (prsid, classid, type, source_dtid,
target_dtid,
                       target_length, target_status, rowtype)
values (0, 0x0000000001000007, 'D', 0x000000000000000E,
0x0000000000010205,
        136, 0, 0)

Command(s) intended for 'edsprs01.edbprs01':
delete from rs_translation where classid = 0x0000000001000007 and
                                source_dtid = 0x000000000000000F

Command(s) intended for 'edsprs01.edbprs01':
insert rs_translation (prsid, classid, type, source_dtid,
target_dtid,
                       target_length, target_status, rowtype)
values (0, 0x0000000001000007, 'D', 0x000000000000000f,
```

```
0x0000000000010205,
        136, 0, 0)

Command(s) intended for 'edsprs01.edbprs01':
delete from rs_translation where classid = 0x0000000001000007 and
                                 source_dtid = 0x000000000000001b

Command(s) intended for 'edsprs01.edbprs01':
insert rs_translation (prsid, classid, type, source_dtid,
target_dtid,
                       target_length, target_status, rowtype)
values (0, 0x0000000001000007, 'D', 0x000000000000001b,
0x0000000000010201,
        9, 0, 0)

Command(s) intended for 'edsprs01.edbprs01':
delete from rs_translation where classid = 0x0000000001000007  and
                                 source_dtid = 0x000000000000001c

Command(s) intended for 'edsprs01.edbprs01':
insert rs_translation (prsid, classid, type, source_dtid,
target_dtid,
                       target_length, target_status, rowtype)
values (0, 0x0000000001000007, 'D', 0x000000000000001c,
0x0000000000010200,
        19, 0, 0)

Command(s) intended for 'oracle.instance':
drop table rs_info

Command(s) intended for 'oracle.instance':
commit

Command(s) intended for 'oracle.instance':
create table rs_info (rskey varchar2 (20), rsval varchar2 (20))

Command(s) intended for 'oracle.instance':
commit

Command(s) intended for 'oracle.instance':
insert into rs_info values ('charset_name', 'iso_1')

Command(s) intended for 'oracle.instance':
insert into rs_info values ('sortorder_name', 'bin_iso_1')

Command(s) intended for 'oracle.instance':
commit

Command(s) intended for 'oracle.instance':
drop public synonym rs_lastcommit

Command(s) intended for 'oracle.instance':
commit

Command(s) intended for 'oracle.instance':
drop table rs_lastcommit
```

```
Command(s) intended for 'oracle.instance':
commit

Command(s) intended for 'oracle.instance':
create table rs_lastcommit(origin  number(8),origin_qid char(72),
                        secondary_qid char(72),origin_time  date,
                           dest_commit_time date)

Command(s) intended for 'oracle.instance':
commit

Command(s) intended for 'oracle.instance':
grant all on rs_lastcommit to public

Command(s) intended for 'oracle.instance':
commit

Command(s) intended for 'oracle.instance':
create public synonym rs_lastcommit for rs_lastcommit

Command(s) intended for 'oracle.instance':
commit

Command(s) intended for 'oracle.instance':
CREATE OR REPLACE PROCEDURE
     RS_UPDATE_SEQUENCE(SequenceName VARCHAR2, SequenceValue
NUMBER,
                        Increment NUMBER)
    AS CurrentID NUMBER; LastID NUMBER; SeqCursor INTEGER; SQLStmt
        VARCHAR2(1024);
     Result NUMBER;
     BEGIN
     SQLStmt := 'SELECT ' || SequenceName || '.NEXTVAL FROM DUAL';
     SeqCursor := DBMS_SQL.OPEN_CURSOR;
     DBMS_SQL.PARSE(SeqCursor,SQLStmt,DBMS_SQL.NATIVE);
     DBMS_SQL.DEFINE_COLUMN(SeqCursor, 1, LastID);
     Result := DBMS_SQL.EXECUTE_AND_FETCH(SeqCursor);
     DBMS_SQL.COLUMN_VALUE(SeqCursor,1,CurrentID);
     LOOP
       IF ( Increment < 0 ) THEN EXIT WHEN CurrentID <=
SequenceValue;
         EXIT WHEN CurrentID > LastID;
       ELSE EXIT WHEN CurrentID >= SequenceValue;
         EXIT WHEN CurrentID < LastID;
       END IF;
       LastID := CurrentID;
       Result := DBMS_SQL.EXECUTE_AND_FETCH(SeqCursor);
       DBMS_SQL.COLUMN_VALUE(SeqCursor,1,CurrentID);
     END
     LOOP;
       DBMS_SQL.CLOSE_CURSOR(SeqCursor);
     END;

Command(s) intended for 'oracle.instance':
grant all on RS_UPDATE_SEQUENCE to public
```

```
Command(s) intended for 'oracle.instance':
DROP sequence rs_ticket_seq

Command(s) intended for 'oracle.instance':
CREATE sequence rs_ticket_seq

Command(s) intended for 'oracle.instance':
Drop table rs_ticket_history

Command(s) intended for 'oracle.instance':
CREATE TABLE rs_ticket_history(cnt numeric(8,0, h1 varchar(10,
   h2 varchar(10), h3 varchar(10), h4 varchar(50), pdb
varchar(30),
   prs varchar(30), rrs varchar(30), rdb varchar(30), pdb_t date,
   exec_t date, dist_t date, rsi_t date, dsi_t date,
  rdb_t date default current_date, exec_b int, rsi_b int, dsi_tnx
int,
   dsi_cmd int, ticket varchar(1024))

Command(s) intended for 'oracle.instance':
create unique index rs_ticket_idx on rs_ticket_history(cnt)

Command(s) intended for 'oracle.instance':
create or replace trigger rs_ticket_tri
   before insert on rs_ticket_history
   for each row
   begin
     if :new.cnt is null then
       select rs_ticket_seq.nextval into :new.cnt  from dual;
     end if;
   end rs_ticket_tri;Command(s) intended for 'oracle.instance':
grant all on rs_ticket_history to public

Command(s) intended for 'oracle.instance':
commit
```

用法

- 连接配置文件指定了函数字符串类和错误类。连接配置文件还可以指定其它连接选项，例如，是否应对命令进行批处理以及要使用的命令分隔符。除了连接设置以外，连接配置文件还可以指定要在复制数据库中创建的 RSSD 和对象（如 *rs_lastcommit* 表）中安装的类级别转换。
- 在使用连接配置文件创建连接时，将刷新系统表服务 (STS) 高速缓存，因此，您无需重新启动 Replication Server。
- 应始终紧靠 **using profile** 子句后面指定 **set username** 子句。

另请参见

- admin show_connection_profiles （第 63 页）
- create connection （第 214 页）

create database replication definition

创建用于复制数据库或数据库对象的复制定义。

语法

```
create database replication definition db_repdef
           with primary at server_name.db
           [[not] replicate DDL]
           [[not] replicate setname setcont]
           [[not] replicate setname setcont]
           [[not] replicate setname setcont]
           [[not] replicate setname setcont]
            [[not] replicate {SQLDML | DML_options} [in table_list]]
```

```
setname  ::= {tables | functions | transactions | system procedures}
```

```
setcont  ::= [[in] ([owner1.]name1[, [owner2.]name2 [, ... ]])]
```

注意： *setname* 中的 functions 一词指用户定义的存储过程或用户定义的函数。

参数

- **db_repdef** – 数据库复制定义的名称。
- **server_name.db** – 主服务器/数据库组合的名称。例如：*TOKYO.dbase*。
- **[not] replicate DDL** – 指示 Replication Server 是否将 DDL 发送到预订数据库。如果未包含“replicate DDL”，或者此子句包含“not”，则不会将 DDL 发送到复制数据库。
- **[not] replicate setname setcont** – 指定是否将 *setname* 类别中指定的对象发送到复制数据库。对于表、函数、事务和系统过程，*setname* 类别最多分别包含一个子句。

  如果省略系统过程 *setname* 或包含 **not** 选项，则 Replication Server 不会复制系统过程。

  如果省略表、函数或事务 *setname* 或包含 **not** 选项，Replication Server 将复制 *setname* 类别的所有对象。
- **[not] replicate {SQLDML | DML_options} [in table_list]** – 通知 Replication Server 是否将 SQL 语句复制到 *in table_list* 中定义的表。
- **SQLDML** – 以下 DML 操作：
  - U – **update**
  - D – **delete**
  - I – **insert select**
  - S – **select into**
- **DML_options** – 以下 DML 操作的任意组合：

  - U – **update**
  - D – **delete**
  - I – **insert select**
  - S – **select into**

  如果将数据库复制模式设置为任何 **UDIS** 组合，RepAgent 将会发送各个日志记录以及 Replication Server 生成 SQL 语句所需的其它信息。
- **owner** – 表的所有者或执行事务的用户。Replication Server 不会为函数或系统过程处理所有者信息。

  可以用由单引号括起的空格或星号替换 *owner*。

  - 空格 (' ') – 表示没有所有者。
  - 星号 (*) – 表示全部所有者。因此，例如，**.publisher* 表示所有名为 *publisher* 的表，而与其所有者无关。
- **name** – 表、函数、事务或系统过程的名称。

  可以用由单引号括起的空格或星号替换 *name*。

  - 空格 (' ') – 表示没有名称。例如，*maintuser.*’ ‘表示所有未命名的维护用户事务。
  - 星号 (*) – 表示所有名称。例如，*robert.** 表示 *robert* 拥有的所有表（或事务）。

示例

- **示例 1** – 创建数据库复制定义 *rep_1B*。此数据库复制定义指定只复制表 *employee* 和 *employee_address*:

```
create database replication definition rep_1B
  with primary at PDS.pdb
  replicate tables in (employee, employee_address)
```

- **示例 2** – 创建数据库复制定义 **rep_2**。在此示例中，将会复制数据库 *my_db*，复制 DDL，但不会复制系统过程：

```
create database replication definition rep_2
  with primary at dsA.my_db
  replicate DDL

  not replicate system procedures
```

- **示例 3** – 从 *pdb1* 数据库的所有表中复制 **insert**、**update**、**delete** 和 **select into** 命令。将复制所有事务和函数，但不会复制 DDL 和系统过程：

```
create database replication definition rep_3
   with primary at ds3.pdb1
   replicate SQLDML
```

  此示例具有与上一示例相同的结果：

```
create database replication definition rep_3
   with primary at ds3.pdb1
   replicate 'UDSI'
```

- **示例 4** – 滤除所有表的 **select into** 语句。第二个子句 **not replicate 'U' in (T)** 会过滤 *T* 表上的更新：

```
create database replication definition dbrepdef
       with primary at ds1.pdb1
       not replicate  'S'
       not replicate  'U'  in (T)
go
```

- **示例 5** – 使用 replicate 'UD' 子句在所有表上启用 **update** 和 **delete** 语句：

```
create database replication definition dbrepdef_UD
       with primary at ds2.pdb1
       replicate 'UD'
go
```

- **示例 6** – 您可以在同一定义中多次使用多个子句指定表。但是，在每个定义中，您只能分别使用一次 **U**、**D**、**I** 和 **S**：

```
create database replication definition dbrepdef
       with primary at ds2.pdb1
       replicate tables in (tb1,tb2)
       replicate 'U' in (tb1)
       replicate 'I' in (tb1,tb2)
go
```

- **示例 7** – 一个复制定义，用于复制数据库中除 *T* 表以外的所有其它表的所有用户存储过程、系统过程和 DML。对于 *T* 表，该复制定义复制除 **delete** 命令以外的所有其它命令：

```
create database replication definition repdef_7
   with primary at ds3.pdb1
   replicate functions
   replicate system procedures
   replicate 'IUS'      /* replicate 'IUS' DML for all tables,
including */
                       /* table 'T' */
   not replicate 'D' in (T)  /* not replicate 'D' DML for table T,
but */
                           /* replicate 'D' for all other tables
*/
```

用法

- 使用 **create database replication definition** 可从主数据库复制所有内容（包括某些例外情况）或者只复制一部分表、函数、事务和系统过程。
- **create database replication definition** 可单独使用，或者与表和函数复制定义结合使用。
- 如果只使用数据库复制定义（即，不使用表或函数复制定义），则 Replication Server 无法转换数据。但是，它可以复制最少量的列。此数据复制行为与缺省热备份的数据复制行为相似。

若要在不使用表级别复制定义的情况下使用数据库复制定义复制加密列，必须使用 INIT_VECTOR NULL 和 PAD NULL 为加密列定义加密密钥。如果数据库中的表包括加密列，而加密密钥是利用随机填充（缺省设置）或初始化矢量在这些列中创建的，要确保数据库的一致性，表级别复制定义是不可或缺的。

- 数据库复制定义是全局对象。它们将被复制到具有来自定义 Replication Server 的路由的所有 Replication Server。
- 数据库复制定义不影响请求函数复制。
- 如果存在表和函数预订，则不实现表和函数过滤器。
- Replication Server 不会为函数和系统过程处理所有者信息。

所有者信息

- Replication Server 始终使用数据库复制定义中提供的所有者信息。
- 如果表标记为 **sp_reptostandby**，则 Replication Server 不使用表复制定义中提供的所有者信息。
- 只有在使用带有 **owner_on** 子句的 **sp_setreptable** 标记表后，Replication Server 才会使用表复制定义中提供的所有者信息。

SQL 语句复制

- 若要在 MSA 环境中复制 SQL 语句，必须在复制定义中包含 **replicate SQLDML** 子句。
- 可以在 **create database replication definition** 中使用多个 replicate 子句。但对于 **alter database replication definition**，您只能使用一个子句。
- 如果在复制定义中未指定过滤器，则缺省过滤器为 **not replicate** 子句。若要更改 SQLDML 过滤器，请应用 **alter database replication definition**。您可以在 **replicate** 子句中指定一个或多个 SQLDML 过滤器。
- 如果为表定义了带有 **send standby** 子句的表复制定义，表复制定义的 SQL 复制设置将覆盖该表的数据库复制定义中定义的设置。

另请参见

- alter database replication definition （第 135 页）
- drop database replication definition （第 298 页）

create error class

创建错误类。

语法

```
create [replication server] error class error_class
    [set template to template_error_class]
```

参数

- **replication server** – 指示新错误类是 Replication Server 错误类，而不是数据服务器错误类。
- **error_class** – 新错误类的名称。此名称在复制系统中必须是唯一的，且必须符合标识符的规则。

  注意： Replication Server 错误类和数据服务器错误类不能使用相同的名称。
- **set template to template_error_class** – 可以使用此子句根据其它错误类来创建某个错误类。 **create error class** 将模板错误类中的错误操作复制到新错误类中。

示例

- **示例 1** – 此示例创建名为 *pubs2_db_err_class* 的新错误类：

```
create error class pubs2_db_err_class
```

- **示例 2** – 根据 **rs_oracle_error_class** 创建 **my_error_class** 错误类：

```
create error class my_error_class set template to
rs_oracle_error_class
```

- **示例 3** – 创建一个名为 **pubs2_rs_err_class** 的新 Replication Server 错误类：

```
create replication server error class
   pubs2_rs_err_class
```

- **示例 4** – 根据缺省 Replication Server 错误类 **rs_repserver_error_class** 创建 Replication Server 错误类 **my_rs_err_class**：

```
create replication server error class my_rs_err_class
set template to rs_repserver_error_class
```

用法

- 使用 **create error class** 可创建错误类。错误类是用于对数据库错误操作指派进行分组的名称。
- 此命令具有以下要求：
  - 必须存在创建错误类的 Replication Server 到管理要使用错误类的数据服务器的 Replication Server 之间的路由。
  - *rs_sqlserver_error_class* 是为 Adaptive Server 数据库提供的缺省错误类；而 *rs_repserver_error_class* 是为 Replication Server 提供的缺省错误类。最初，这两个错误类没有主节点。必须先在主节点上创建这些错误类，然后才能更改缺省错误操作。
  - 使用 **create error class** 后，请使用 **rs_init_erroractions** 存储过程来初始化错误类。
- 使用 **create connection** 或 **alter connection** 将错误类与数据库相关联。每个数据库可以有一个错误类。一个错误类可以与多个数据库相关联。

- Replication Server 通过复制系统将新的错误类分发到合格的节点。通常由于复制系统会有一定时间的滞后，因此更改不会立即显示在所有这些节点上。

指派错误操作

- 使用 **assign action** 来更改 Replication Server 对特定数据服务器错误的响应。在创建错误类的 Replication Server 上指派操作。

删除错误类

- 使用 **drop error class** 可删除错误类以及与其相关联的所有操作。

非 Adaptive Server 错误类

- 可以使用 **create connection** 和 **alter connection** 命令，将非 Adaptive Server 错误类指派给非 Adaptive Server 复制数据库上的特定连接。
- 当 Replication Server 建立到非 ASE 复制服务器的连接时，Replication Server 会验证是否对连接启用从非 ASE 复制服务器返回本机错误代码的选项。如果未启用此选项，Replication Server 会记录警告消息，表明连接有效但错误操作映射可能不正确。

  请参见 Replication Server Options 文档中的“**ReturnNativeError**”以在 Enterprise Connect™ Data Access (ECDA) Option for ODBC 中为复制服务器设置此选项。
- 有关非 Adaptive Server 错误类列表，请参见“错误类和函数类”表。有关非 Adaptive Server 复制错误类的详细信息，请参见《Replication Server 管理指南第一卷》。

权限

create error class 需要“sa”权限。

另请参见

- alter connection （第 109 页）
- alter error class （第 138 页）
- assign action （第 172 页）
- create connection （第 214 页）
- drop error class （第 299 页）
- move primary （第 318 页）
- rs_init_erroractions （第 546 页）

create function

创建用户定义的函数。

注意： 创建函数复制定义时，将自动创建用户定义的函数。有关详细信息，请参见 **create applied function replication definition** 和 **create request function replication definition**。

如果应用程序使用与表复制定义相关联的异步过程传输，则可能需要创建用户定义的函数。有关详细信息，请参见《Replication Server 管理指南第二卷》。

语法

```
create function replication_definition.function
([@param_name datatype [, @param_name datatype]...])
```

参数

- **replication_definition** – 函数的复制定义的名称。可以为同一表的所有复制定义只创建一个用户定义的函数。如果同一个表有多个复制定义，可以指定其中任何一个定义的名称。但是，每个复制定义对此用户定义的函数都有其各自的函数字符串。
- **function** – 函数名。此名称对于复制定义必须是唯一的，且必须符合标识符的规则。“Replication Server 系统函数”中列出的系统函数名称以及以“rs_”开头的所有函数名称均已保留。
- **@param_name** – 用户定义的函数的参数名称。每个参数名称前必须带有 @ 符号，且必须符合标识符规则。执行函数时提供每个参数的值。
- **数据类型** – 参数的数据类型。某些数据类型要求数据类型名称后面有一个用小括号括起来的长度。有关数据类型及其语法的说明，请参见“数据类型”。数据类型不能是 *text*、*unitext*、*rawobject* 或 *image*。

示例

- **示例 1** – 为 *publishers_rep* 复制定义创建一个名为 *newpublishers* 的用户定义函数，此函数具有四个参数：

```
create function publishers_rep.newpublishers
 (@pub_id char(4), @pub_name varchar(40),
 @city varchar(20), @state char(2))
```

用法

- 使用 **create function** 创建用户定义的函数。
- 在创建复制定义的 Replication Server 上执行 **create function**。

- 用户定义的函数可用于异步过程传递。有关异步过程的详细信息，请参见《Replication Server 管理指南第二卷》。
- 必须将列出的参数用小括号 **()** 括起来，即使要定义的函数没有参数也应如此。
- 对于系统提供的三个函数字符串类中的每个类（将在其中使用用户定义的函数）以及从这些类继承的每个派生类，Replication Server 将为用户定义的函数生成缺省函数字符串。
- 可以使用 **create function string** 自定义 *rs_sqlserver_function_class* 和用户创建的函数字符串类中的函数字符串。
- 对于用户创建的每个基本函数字符串类（将在其中使用用户定义的函数）以及从这些类继承的每个派生类，必须使用 **create function string** 来创建函数字符串。函数字符串应当使用适合于复制数据服务器的语言来调用存储过程或 RPC。
- 有关函数字符串类、函数字符串和函数的概述，请参见《Replication Server 管理指南第二卷》。
- Replication Server 通过复制系统将新的用户定义函数分发到合格的节点。通常由于复制系统会有一定时间的滞后，因此更改不会立即显示在所有这些节点上。
- 为复制定义创建用户定义函数的同时，也为主表中的所有复制定义创建了此函数。

权限

create function 需要“create object”权限。

另请参见

- create applied function replication definition （第 206 页）
- create function string （第 236 页）
- create request function replication definition （第 269 页）
- drop function （第 300 页）

create function replication definition

为要复制的存储过程创建函数复制定义和用户定义的函数。

注意： **create function replication definition** 和 **alter function replication definition** 是将来不再支持的命令。Sybase 建议您改用下列命令：

- **create applied function replication definition** 和 **alter applied function replication definition**。
- **create request function replication definition** 和 **alter request function replication definition**。

语法

```
create function replication definition
        function_rep_def
```

```
with primary at data_server.database
[deliver as 'proc_name']
        ([@param_name datatype [, @param_name datatype]...])
        [searchable parameters (@param_name
         [, @param_name]...)]
        [send standby {all | replication definition}
        parameters]
```

参数

- **function_rep_def** – 函数复制定义的名称。此名称必须符合标识符规则。
- **with primary at** – 指定包含主数据的数据服务器和数据库。
- **data_server** – 包含主数据的数据服务器的名称。如果主数据库是热备份应用程序的一部分，则 *data_server* 是逻辑数据服务器名称。
- **database** – 包含主数据的数据库的名称。如果主数据库是热备份应用程序的一部分，则 *database* 是逻辑数据库名称。
- **deliver as** – 指定要在传递复制函数的数据库上执行的存储过程的名称。*proc_name* 是一个字符串，最多可包含 200 个字符。如果不使用此子句，函数将作为与函数复制定义同名的存储过程传递。
- **@param_name** – 函数的参数名。参数名在每个子句中最多只能出现一次。不要求包括参数及其数据类型，但不管是否包括参数，此子句必须包括小括号 **( )**。
- **数据类型** – 函数中参数的数据类型。有关数据类型及其语法的列表，请参见“数据类型”。Adaptive Server 存储过程和函数复制定义不能包含数据类型为 *text*、*unitext*、*rawobject* 和 *image* 的参数。
- **searchable parameters** – 指定可在 **define subscription**、**create subscription** 或 **create article** 的 **where** 子句中使用的参数的列表。如果包括此子句，则必须包括小括号 **( )**。
- **send standby** – 在热备份应用程序中，指定是将此函数中的所有参数都发送到备用数据库 (**send standby all parameters**)，还是只将复制定义中指定的参数发送到备用数据库 (**send standby replication definition parameters**)。缺省值是 **send standby all parameters**。

示例

- **示例 1** – 为同名的函数和存储过程创建名为 *titles_frep* 的函数复制定义。主数据库位于 LDS 数据服务器的 *pubs2* 数据库中。请将此类函数复制定义用于应用函数：

```
create function replication definition titles_frep
 with primary at LDS.pubs2
 (@title_id varchar(6), @title varchar(80),
  @type char(12), @pub_id char(4),
  @price money, @advance money,
  @total_sales int)
  searchable parameters (@title_id, @title)
```

- **示例 2** – 如上例所示，为同名的函数和存储过程创建名为 *titles_frep* 的函数复制定义。在这种情况下，要在目标数据库中调用的存储过程名为 *upd_titles*。请将此类函数复制定义用于请求函数：

```
create function replication definition titles_frep
 with primary at LDS.pubs2
 deliver as 'upd_titles'
 (@title_id varchar(6), @title varchar(80),
  @type char(12), @pub_id char(4),
  @price money, @advance money,
  @total_sales int)
  searchable parameters (@title_id, @title)
```

用法

- 使用 **create function replication definition** 来说明要复制的存储过程。有关复制存储过程的概述，请参见《Replication Server 管理指南第一卷》。
- 在管理存储主数据的数据库的 Replication Server 上执行 **create function replication definition**。
- 每个复制存储过程只能创建一个函数复制定义。
- 执行此命令之前，请确保满足以下条件：
  - 函数复制定义名称在复制系统中是唯一的。使用 **create function replication definition** 时，Replication Server 不能始终检查是否满足此要求。
  - Replication Server 与存储主数据的数据库之间已建立连接。有关详细信息，请参见 **create connection**。也可以使用 **rs_init** 来创建连接。请查阅适用于您的平台的 Replication Server 安装和配置指南。
  - 为函数复制定义指定的名称、参数和数据类型必须与涉及到的存储过程的名称、参数和数据类型相匹配。您可以只指定那些在复制过程中要用到的参数。
- 要更新表，需要与表复制定义关联的复制存储过程，而不需要与函数复制定义关联的复制存储过程。这样，您就可以复制与复制数据不相关的事务。有关系统存储过程的详细信息，请参见“RSSD 存储过程”。

  有关两种复制存储过程类型的详细信息，请参见 **sp_setrepproc**。
- Replication Server 通过复制系统将新的函数复制定义分发到合格的节点。通常由于复制系统会有一定时间的滞后，因此更改不会立即显示在所有这些节点上。

用户定义的函数和函数字符串

- 创建函数复制定义时，Replication Server 会自动创建一个相应的用户定义的函数。
- 对于系统提供的函数字符串类（将在其中使用与此函数复制定义关联的用户定义函数）以及从这些类继承的每个派生类，Replication Server 将为用户定义的函数生成缺省函数字符串。
- 可以使用 **create function string** 自定义 *rs_sqlserver_function_class* 和用户创建的函数字符串类中的函数字符串。

- 对于用户创建的每个基本函数字符串类（将在其中使用用户定义的函数）以及从这些类继承的每个派生类，必须使用 **create function string** 来创建函数字符串。函数字符串应当使用适合于复制数据服务器的语言来调用存储过程或 RPC。
- 有关函数字符串类、函数字符串和函数的概述，请参见《Replication Server 管理指南第二卷》。

with primary at 子句

- 使用 **with primary at** 子句来指定包含主数据的数据服务器和数据库。它不必是包含被调用存储过程的数据库。

  对于应用函数（主到复制的函数复制）和请求函数（复制到主的函数复制），在管理主数据的 Replication Server 上创建函数复制定义，并使用 **with primary at** 子句指定主数据库。

deliver as 子句

- 使用可选的 **deliver as** 子句来指定要在目标数据库（在此数据库中传递复制函数）上执行的存储过程的名称。如果在创建或更改函数复制定义时未使用此子句，函数将作为与函数复制定义同名的存储过程传递。

  在热备份数据库和活动数据库中，存储过程具有相同的名称，因而会忽略 **deliver as** 子句。

  通常会在请求函数传递中使用 **deliver as** 子句；也就是在将函数从复制 Replication Server 复制到主 Replication Server 时。这样，复制函数的名称将不同于所执行的存储过程的名称。

  此方法用于进行“往返”存储过程复制。在此复制中，作为请求函数目标的主 Replication Server 将执行一个应用函数，而源复制 Replication Server 又预订此函数。

  有关详细信息，请参见《Replication Server 管理指南第一卷》。

HDS 参数的函数复制定义

- 虽然无法创建更改参数值数据类型的函数复制定义，但是可以使用 HDS 数据类型定义来声明应用函数复制定义的参数。这样的参数将受到类级别转换的限制。有关 HDS 的详细信息，请参见《Replication Server 管理指南第一卷》。
- Replication Server 不对请求函数的参数值进行转换。但请注意，在函数字符串映射过程中，Replication Server 使用为已声明数据类型的参数值定义的分隔符来生成 SQL。

更改函数复制定义

- 使用 **alter function replication definition** 可将参数或可搜索参数添加到现有函数复制定义中。还可以指定新的存储过程名称，以供在目标数据库传递复制函数时使用。
- 如果需要删除或重命名函数复制定义中的参数，则必须删除函数复制定义的所有预订（仅限于应用函数）。然后删除函数复制定义，并重新创建此定义。

预订函数复制定义

- 要预订函数复制定义，请使用带有 **without materialization** 子句的 **create subscription**，或使用 **define subscription** 以及其它涉及批量实现的命令。

函数复制定义和表复制定义

- 通过应用函数复制存储过程时，建议为要受到复制存储过程影响的相同的表创建表复制定义和预订。这样做可以确保影响表的所有正常事务以及存储过程执行均会得到复制。
  对于标记为“已复制”的存储过程中的 DML，不会通过表复制进行复制，因而您必须预订该存储过程，即使已预订表也是如此。
- 如果准备将两种复制定义用于同一个表，可以使用表复制定义的预订来实现表数据。然后，可以使用带有 **without materialization** 子句的 **create subscription** 创建函数复制定义的预订。

权限

create function replication definition 需要“create object”权限。

另请参见

- alter function replication definition （第 141 页）
- alter function string （第 143 页）
- create connection （第 214 页）
- create function string （第 236 页）
- define subscription （第 290 页）
- drop function replication definition （第 301 页）
- sp_setrepproc （第 491 页）

create function string

将函数字符串添加到函数字符串类中。Replication Server 使用函数字符串来生成数据服务器的指令。

语法

```
create function string
        {replication_definition |
        [owner.] table |
        stored_procedure} .function[;function_string]
        for { [function_class] function_class |
        [database] data_server.database}
        [with overwrite]
        [scan 'input_template']
        [output
        {language 'lang_output_template' |
        rpc 'execute procedure
        [@param_name=]{constant |?variable!mod?}
```

```
                [, [@param_name=]
                        {constant |?variable!mod?}]...' |
        writetext [use primary log | with log | no log] |
        none}]
```

参数

- **replication_definition** – 函数操作的复制定义的名称。只用于具有复制定义范围的函数。

  函数具有函数字符串类作用域、复制定义作用域或目标作用域。

  引导事务控制的函数具有函数字符串类作用域。用户定义的函数以及修改数据和针对复制定义而创建的函数具有复制定义作用域。

  针对备用或复制表或存储过程而创建的函数字符串是目标作用域函数字符串。
- **[owner.]table** – 指定函数字符串的目标表和表所有者。
- **存储过程** – 指定函数字符串的目标存储过程
- **函数** – 函数名。系统函数名称必须按照“Replication Server 系统函数”中的说明提供。用户定义函数的名称必须与现有的用户定义函数相匹配。
- **function_string** – 自定义 **rs_get_textptr**、**rs_textptr_init** 和 **rs_writetext** 函数时，必须提供函数字符串名称；对于其它函数，则是可选的。对于 **rs_get_textptr**、**rs_textptr_init** 和 **rs_writetext**，复制定义中的每个 *text*、*unitext* 或 *image* 列都需要一个函数字符串。所提供的函数字符串名称必须是：

  - 复制定义的 *text*、*unitext* 或 *image* 列名。
  - 能够符合标识符规则。
  - 在函数范围内是唯一的。

  Replication Server 还使用函数字符串名称来生成错误消息。
- **function_class** – 为复制定义作用域函数字符串指定函数字符串与之关联的函数类。
- **data_server.database** – 指定您要在其中为目标表或存储过程创建目标作用域函数字符串的备用数据库或复制数据库。

  请参见《Replication Server 管理指南第二卷》中的“创建函数字符串”。
- **with overwrite** – 如果函数字符串已经存在，则如同使用 **alter function string** 一样，此选项将删除并重新创建函数字符串。只能将 **with overwrite** 选项与 create function string 一起使用。
- **scan** – 在输入模板之前。
- **input_template** – 用单引号字符引起来的字符串，Replication Server 对其进行扫描，以将 **rs_select** 或 **rs_select_with_lock** 函数字符串与 **create subscription** 命令中的 **where** 子句相关联。输入模板字符串是以 SQL **select** 语句的形式编写的，并使用用户定义的变量而非预订的 **where** 子句中的实际值。
- **输出** – 在输出模板之前。

- **language** – 指示 Replication Server 使用客户端/服务器接口语言接口向数据服务器提交输出模板命令。
- **lang_output_template** – 用单引号字符引起来的字符串，包含数据服务器指令。语言输出模板字符串可以包含嵌入变量，Replication Server 在将字符串发送到数据服务器之前，会使用运行时值替换这些变量。
- **rpc** – 输出模板，通知 Replication Server 使用客户端/服务器接口远程过程调用 (RPC) 接口。Replication Server 解释字符串并构建远程过程调用以发送到数据服务器。

  RPC 输出模板中显示以下关键字和选项：

  procedure – 要执行的远程过程调用的名称。它可以是 Adaptive Server 存储过程、Open Server 网关 RPC 处理程序处理的过程或 Open Server 网关中注册的过程。有关在网关程序中处理 RPC 的信息，请参见《Open Server Server-Library/C 参考手册》。

  @*param_name* – 过程参数的名称，由过程定义。如果使用 @*param_name* = *value* 形式，可以按照任意顺序提供参数。如果省略参数名，则必须按照远程过程中定义的顺序提供参数值。

  constant – 一个实际值，数据类型是将此值指派给的参数的数据类型。

  ?*variable*!*mod*? – *variable* 是运行时值的占位符。它可以是列名、系统定义的变量名、用户定义函数中的参数名或输入模板中定义的变量名。变量引用的值必须与将其指派给的参数的数据类型相同。有关系统定义变量的列表，请参见“系统定义变量”。

  变量名的 *mod* 部分标识变量所表示的数据类型。所有变量都需要变量修饰符，变量修饰符必须是以下各项之一：

表 30. 函数字符串变量修饰符

修饰符	说明
new、*new_raw*	对要插入或更新的行中某一列的新值的引用
old、*old_raw*	对要更新或删除的行中某一列的现有值的引用
user、*user_raw*	对 **rs_select** 或 **rs_select_with_lock** 函数字符串的输入模板中定义的变量的引用。
sys、*sys_raw*	对系统定义变量的引用
param、*param_raw*	对函数参数的引用

修饰符	说明
text_status	对 *text*、*unitext* 或 *image* 数据的 *text_status* 值的引用。可能的值有： • 0x000 – 文本字段包含 NULL 值，且尚未初始化文本指针。 • 0x0002 – 已初始化文本指针。 • 0x0004 – 后面为实际文本数据。 • 0x0008 – 后面无文本数据，因为未复制文本数据。 • 0x0010 – 文本数据未被复制，但包含 NULL 值。

注意： 用户定义的函数的函数字符串不可以使用 new 或 old 修饰符。

- **writetext** – 指示 Replication Server 使用 Client-Library™ 函数 **ct_send_data** 来更新 *text*、*unitext* 或 *image* 列的值。此选项仅适用于 **rs_writetext** 函数。

  以下选项会在 **writetext** 输出模板中显示，用来指定复制数据库中 *text*、*unitext* 或 *image* 列的记录行为：

  use primary log – 在复制数据库中记录数据（如果在主数据库中指定了记录选项）。

  with log – 在复制数据库事务日志中记录数据。

  no log – 不会将数据记录到复制数据库事务日志中。
- **none** – 应用于所有函数，并可灵活地标识 Replication Server 可在复制数据库上避免执行哪些函数字符串：
  - 对于 **rs_writetext** 函数 – 指示 Replication Server 不要复制 *text*、*unitext* 或 *image* 列的值。
  - 对于非 **rs_writetext** 函数 – 指示 Replication Server 不要在复制数据库上执行命令。

示例

- **示例 1** – 创建 **rs_begin** 函数的函数字符串：

```
create function string rs_begin
 for sqlserver2_function_class
 output language
 'begin transaction'
```

- **示例 2** – 创建 **rs_commit** 函数的函数字符串，包含两个以分号分隔的命令。此函数字符串执行 Adaptive Server 存储过程，以更新 *rs_lastcommit* 系统表并随后提交事务：

```
create function string rs_commit
 for sqlserver2_function_class
 output language
 'execute sqlrs_update_lastcommit
   @origin = ?rs_origin!sys?,
   @origin_qid = ?rs_origin_qid!sys?,
```

```
   @secondary_qid = ?rs_secondary_qid!sys?;
   commit transaction'
```

- **示例 3** – 示例 3 和 4 为 *titles* 表创建复制定义，并为 *sqlserver2_function_class* 创建一个 **rs_insert** 函数字符串。此函数字符串将数据插入复制数据库的 *titles_rs* 表，而不是插入 *titles* 表中：

```
create replication definition titles_rep
 with primary at LDS.pubs2
 (title_id varchar(6), title varchar(80),
  type char(12), pub_id char(4), advance money,
  total_sales int, notes varchar(200),
  pubdate datetime, contract bit, price money)
 primary key (title_id)
 searchable columns (price)
```

- **示例 4** – 示例 3 和 4 为 *titles* 表创建复制定义，并为 *sqlserver2_function_class* 创建一个 **rs_insert** 函数字符串。此函数字符串将数据插入复制数据库的 *titles_rs* 表，而不是插入 *titles* 表中：

```
create function string titles_rep.rs_insert
 for sqlserver2_function_class
 output language
 'insert titles_rs values (?title_id!new?,
   ?title!new?, ?type!new?, ?pub_id!new?,
   ?advance!new?, ?total_sales!new?, ?notes!new?,
   ?pubdate!new?, ?contract!new?, ?price!new?)'
```

- **示例 5** – 示例 5 和 6 创建一个用户定义的函数 **update_titles**，并为 *sqlserver2_function_class* 创建相应的函数字符串。此函数字符串执行一个名为 **update_titles** 的 Adaptive Server 存储过程：

```
create function titles_rep.update_titles
 (@title_id varchar(6), title varchar(80),
  @price money)
```

- **示例 6** – 示例 5 和 6 创建一个用户定义的函数 **update_titles**，并为 *sqlserver2_function_class* 创建相应的函数字符串。此函数字符串执行一个名为 **update_titles** 的 Adaptive Server 存储过程：

```
create function string titles_rep.update_titles
 for sqlserver2_function_class
 output rpc
 'execute update_titles
   @title_id = ?title_id!param?,
   @title = ?title!param?,
   @price = ?price!param?'
```

- **示例 7** – 示例 7 中的 **rs_select** 函数字符串用于实现请求 *title_id* 列为指定值的行的预订。与示例 8 相似，**scan** 子句指定的输入模板将这两个函数字符串区分开：

```
create function string
   titles_rep.rs_select;title_id_select
 for sqlserver2_function_class
 scan 'select * from titles
   where title_id = ?title_id!user?'
```

```
output language
'select * from titles
  where title_id = ?title_id!user?'
```

- **示例 8** – 示例 8 中的 **rs_select** 函数字符串是 RPC 函数字符串的示例。它用于实现一些预订以请求 *price* 列值在指定范围内的行：

```
create function string
   titles_rep.rs_select;price_range_select
for sqlserver2_function_class
scan 'select * from titles
  where price > ?price_min!user?
  and price < ?price_max!user?'
output rpc
  'execute titles_price_select
  ?price_min!user?, ?price_max!user?'
```

- **示例 9** – 为数据库 NY_DS.rdb1 的 **upd_datetime** 存储过程创建目标作用域函数字符串：

```
create function string upd_datetime.upd_datetime
     for database NY_DS.rdb1
     with overwrite
          output language
          'update datetime set
          row_num = ?row_num!param?,
          datecol = ?datecol!param?,
          timecol = ?timecol!param?,
          ndatecol = ?ndatecol!param?,
          ntimecol = ?ntimecol!param?,
          comment = ?comment!param?
          where
          row_num = ?row_num!param?'
```

- **示例 10** – 为 NY_DS.rdb1 的 dbo.datetime 表创建目标作用域函数字符串：

```
create function string dbo.datetime.rs_insert
     for database NY_DS.rdb1
     with overwrite
     output language
     'insert datetime values (
             ?row_num!new? ,
             ?datecol!new? ,
             ?timecol!new? ,
             ?ndatecol!new? ,
             ?ntimecol!new? ,
             ?comment!new?)
       update fn_monitor set insert_count = insert_count + 1'
```

- **示例 11** – 为 dbo.tbl1.unitext_fd1 列创建 rs_writetext 自定义函数字符串：

```
create function string dbo.tbl1.rs_writetext; unitext_fld1 for
NY_DS.rdb1
         output RPC
        'exec update_repl_unitext
                @p_key       = ?p_key!new?,
```

```
                @unitext_fld       = ?unitext_fld1!new?,
                @last_chunk         = ?rs_last_text_chunk!sys?'
```

- **示例 12** – 为 dbo.tbl1 表创建目标作用域函数字符串：

```
create function string dbo.tbl1.rs_datarow_for_writetext
for NY_DS.rdb1
        output RPC
        'exec update_txtimg_stat
                @p_key       = ?p_key!new?,
                @txtfld_stat = ?unitext_fld1!text_status?'
```

- **示例 13** – 为 NY_DS 数据服务器的 rdb1 目标数据库中的 dbo.authors 表创建 **rs_insert** 自定义函数字符串：

```
create function string dbo.authors.rs_insert
     for database NY_DS.rdb1
     output language
     'insert authors values (
           ?au_id!new? ,
           ?au_lname!new? ,
           ?au_fname!new? ,
           ?phone!new? ,
           ?address!new? ,
           ?city!new? ,
           ?state!new? ,
           "00000" ,
           ?contract!new?)
      update fn_monitor set insert_count = insert_count + 1'
```

- **示例 14** – 为 NY_DS 数据服务器的 rdb1 目标数据库中的 **upd_bits** 存储过程创建自定义函数字符串：存储过程的函数具有与存储过程相同的名称：

```
create function string upd_bits.upd_bits
     for database NY_DS.rdb1
     with overwrite
     output language
     'exec upd_bits
           @firstbit = ?firstbit!param?,
           @secondbit = ?secondbit!param?,
     @commit = ?comment!param?'
```

用法

- 使用 **create function string** 将函数字符串添加到函数字符串类中。函数字符串包含 Replication Server 将函数转换为数据库命令所需的特定数据库指令。
- 有关函数、函数字符串和函数字符串类的概述，请参见《Replication Server 管理指南第二卷》。
- 在控制目标数据库（备用数据库或复制数据库）的 Replication Server 中为目标作用域函数字符串执行 **create function string**。
- 只能将 **with overwrite** 选项与 **create function string** 一起使用。
- 复制定义作用域函数字符串与函数类关联，而目标作用域函数字符串与目标数据库关联。

- 如果目标表没有所有者信息，并且您在命令的函数字符串中未指定 **function class** 和 **database** 选项，Replication Server 只有在检查 **for** 关键字后面的字符串格式是表示函数类还是数据库后，才会知道函数字符串是用于复制定义还是用于表。对于:
  - 复制定义作用域函数字符串 – 例如 **rs_sqlserver_function_class**，**for** 关键字后面的字符串格式不包含数据库名称
  - 目标作用域函数字符串 – 例如 `NY_DS.rdb1`，**for** 关键字后面的字符串格式包含数据服务器和数据库的名称
- 您只能对备用数据库或复制数据库，而不能对为多个复制路径配置的连接创建目标作用域函数字符串。
- 在热备份环境中，受影响的数据库是物理数据库。如果想要为逻辑数据库定义目标作用域函数字符串，则必须对活动数据库和备用数据库均发出函数字符串命令。
- 对于备用存储过程或复制存储过程的目标作用域函数字符串，函数名与存储过程名相同。
- 对于备用表或复制表的目标作用域函数字符串，有效函数为：**rs_insert**、**rs_update**、**rs_delete**、**rs_truncate**、**rs_writetext**、**rs_datarow_for_writetext**、**rs_textptr_init** 和 **rs_get_textptr**。
- 仅当对象没有复制定义或对象未使用该对象的所有复制定义时，Replication Server 才使用目标作用域函数字符串。

  请参见《Replication Server 管理指南第一卷》中的“热备份和多节点可用性环境”
- 在函数字符串类的主节点上为具有类范围的函数创建或更改函数字符串。有关函数字符串类的主节点的详细信息，请参见 **create function string class**。
- 在创建复制定义的节点上创建或更改具有复制定义范围的函数（包括用户定义函数）的函数字符串。每个复制定义都有其自己的函数字符串集。
- Replication Server 通过复制系统将新的函数字符串分发到合格的节点。通常由于复制系统会有一定时间的滞后，因此更改不会立即显示在所有这些节点上。
- 有些函数字符串是动态生成的；它们不存储在 RSSD 中。

函数字符串和函数字符串类

- 对于系统提供的每个函数字符串类（将在其中使用某个函数）以及从这些类继承的每个派生类，Replication Server 将为该函数生成缺省函数字符串。对于系统函数和用户定义的函数都是如此。（未提供 **rs_dumpdb** 和 **rs_dumptran** 函数的缺省函数字符串。只有在使用协调的转储时，才需要创建它们。）
- 使用 **alter function string** 自定义 *rs_sqlserver_function_class* 中的函数字符串。请使用 **create function string** 自定义用户创建的函数字符串类中的函数字符串。
- 对于用户创建的每个基本函数字符串类（将在其中使用函数）以及要在其中覆盖继承的函数字符串的每个派生类，必须使用 **create function string** 来创建函数字符串。
- 如果省略 **output** 子句，则指示 Replication Server 按照为 *rs_sqlserver_function_class* 或 *rs_default_function_class* 函数字符串类生成函数字符串的相同方式生成函数字符串。
- 用户定义函数的缺省函数字符串是对一个存储过程的调用，此存储过程的名称为此函数的名称，参数为此函数的参数。存储过程作为语言命令执行，而不是作为 RPC 执行。

注意： ExpressConnect for Oracle 不支持对文本和图像处理使用自定义函数字符串。

请参见《Replication Server 异构复制指南》中的“ExpressConnect 设置”和“函数字符串、错误类和用户定义的数据类型”。

函数字符串和 replicate minimal columns

- 如果已经为复制定义指定了 **replicate minimal columns**，则通常无法为 **rs_update**、**rs_delete**、**rs_get_textptr**、**rs_textptr_init** 或 **rs_datarow_for_writetext** 系统函数创建非缺省函数字符串。
  但是，如果在函数字符串中使用 *rs_default_fs* 系统变量，则可以为 **rs_update** 和 **rs_delete** 函数创建非缺省函数字符串。此变量表示缺省函数字符串行为。可以添加其它命令来扩展函数字符串行为。
- 有关 **replicate minimal columns** 选项的详细信息，请参见 **create replication definiton**。

输入模板和输出模板

- 根据函数的不同，函数字符串可以有输入模板和输出模板。Replication Server 将变量值替换到模板中，并将结果传递到数据服务器进行处理。
- 对输入模板和输出模板有以下要求：
  - 它们的大小限制为 64K。用运行时值替换函数字符串输入模板中的嵌入变量的结果不能超过 64K。
  - 输入模板和语言或 RPC 输出模板由两个单引号字符 (') 分隔。
  - 输入模板和输出模板中的变量名由问号 (?) 分隔。
  - 变量名及其修饰符由感叹号 (!) 分隔。
- 创建函数字符串时：
  - 使用两个连续的单引号字符 ('') 可以表示字符或日期/时间数据类型的数据中或两边的一个实际单引号字符，如以下字符串中的“Berkeley”所示：

    ```
    'insert authors
    (city, au_id, au_lname, au_fname)
    values (''Berkeley'', ?au_id!new?,
    ?au_lname!new?,
    ?au_fname!new?)'
    ```
  - 使用两个连续问号 (??) 可以表示字符数据类型的数据中的一个问号。
  - 使用两个连续分号 (;;) 可以表示字符数据类型的数据中的一个分号。

输入模板

- 输入模板仅用于 **rs_select** 和 **rs_select_with_lock** 函数，这两个函数在非批量预订实现和 **with purge** 预订取消实现期间使用。Replication Server 将预订的 where 子句与输入模板相匹配，以查找要使用的函数字符串。
- 对输入模板具有以下要求：
  - 它们仅包含用户定义的变量，这些变量的值来自 where 子句中的常量。在函数字符串的输出模板中，也可以引用用户定义的变量。

- 如果省略 *input_template*，则它可以匹配所有 **select** 命令。这样，您就可以创建一个缺省函数字符串，在函数字符串类中没有其它函数字符串具有匹配 **select** 命令的 *input_template* 时，将会执行此缺省函数字符串。

输出模板

- 输出模板确定发送到复制数据服务器的命令的格式。大多数输出模板可以使用以下格式之一：语言 RPC 或 none。**rs_writetext** 函数字符串的输出模板可以使用 RPC 格式或其它格式（**writetext** 或 none）。有关这些格式的说明，请参见《Replication Server 管理指南第二卷》。
- Replication Server 将函数字符串输出模板映射到数据服务器命令时，将使用 Adaptive Server 期望的格式来设置变量格式。对于那些不是以 *_raw* 结尾的修饰符（常用修饰符），Replication Server 将修改数据类型，如下所示：
  - 在字符和日期/时间值中出现的单引号字符后添加另一个单引号字符，以对单引号字符的特殊含义进行转义。
  - 在字符和日期/时间值两侧添加单引号字符（如果缺少）。
  - 在货币数据类型的值前面添加适当的货币符号（美式英语中的美元符号）。
  - 在数据类型为二进制的值前面添加“0x”前缀。
  - 在字符值中现有的反斜杠 (\) 和换行符的实例之间添加一对反斜杠和换行符。Adaptive Server 将后面跟换行符的反斜杠当作连续字符，因此，将删除添加的字符对，以保持原始字符不变。

    对于以 *_raw* 结尾的修饰符，Replication Server 不会按照上述方式修改数据类型。

    函数字符串变量格式表总结了对于不以 *_raw* 结尾的修饰符，Replication Server 设置每种数据类型格式的方法：

表 31. 设置函数字符串变量的格式

数据类型	设置实际值的格式
bigint、int、smallint、tinyint、rs_address	整数
unsigned bigint、unsigned int、unsigned smallint、unsigned tinyint	无符号的整数
decimal、numeric、identity	精确十进制数
float 和 real	十进制数
char、varchar	用单引号字符引起来 在所有单引号字符后添加一个单引号字符 在反斜杠和换行符之间添加一对反斜杠和换行符
unichar 和 univarchar	Unicode

数据类型	设置实际值的格式
money 和 smallmoney	添加适当的货币符号（美式英语中的美元符号）
date、time、datetime、smalldatetime	用单引号字符引起来 在所有单引号字符后添加一个单引号字符
binary、timestamp、varbinary	添加“0x”前缀
bit	1 或 0

- 对输出模板有以下要求：
  - 用运行时值替换函数字符串输出模板中的嵌入变量的结果不能超过 64K。
  - 可以将多条命令放在语言函数字符串输出模板中，命令之间用分号 (;) 分隔。如果数据库配置为支持命令批处理（这是缺省配置），Replication Server 在将函数字符串以单个批处理的形式发送到数据服务器之前，将使用此连接的 DSI 命令分隔符来替换分号。此分隔符在 **alter connection** 命令的 **dsi_cmd_separator** 选项中定义。

    要表示一个不会被解释为命令分隔符的分号，可使用两个连续的分号 (;;)。

    如果数据库连接未配置为支持批处理，则 Replication Server 一次一条地将函数字符串中的命令发送到数据服务器。要启用或禁用数据库的批处理，可使用 alter connection。

“Replication Server 系统定义的变量”表列出了可在函数字符串输出模板中使用的系统定义变量。对于这些变量，请使用 *sys* 或 *sys_raw* 修饰符。

表 32. Replication Server 系统定义的变量

系统变量	数据类型	说明
rs_default_fs	text	为函数生成的缺省函数字符串文本
rs_deliver_as_name	varchar(200)	用于复制函数的执行，要在目标调用的过程的名称
rs_destination_db	varchar(30)	发送事务的数据库的名称
rs_destination_ds	varchar(30)	发送事务的数据服务器的名称
rs_destination_ldb	varchar(30)	发送事务的逻辑数据库的名称
rs_destination_lds	varchar(30)	发送事务的逻辑数据服务器的名称
rs_destination_ptype	char(1)	发送事务的数据库的物理连接类型（“A”表示活动，“S”表示备用）
rs_destination_user	varchar(30)	将在目标执行事务的用户

系统变量	数据类型	说明
rs_dump_dbname	varchar(30)	数据库或事务转储源自的数据库的名称
rs_dump_label	varchar(30)	数据库或事务转储的标签信息。对于 Adaptive Server，此变量保存代表转储开始时间的 datetime 值。
rs_dump_status	int(4)	转储状态指示符： • 0 – 表示 dump transaction 命令不包含 **with standby_access** 参数 • 1 – 表示 dump transaction 命令包含 **with standby_access** 参数
rs_dump_timestamp	varbinary(16)	数据库或事务转储的时间戳
rs_lorigin	int(4)	事务的源逻辑数据库的 ID
rs_isolation_level	varchar(30)	数据库连接的事务隔离级别。
rs_origin	int(4)	事务的源数据库的 ID
rs_origin_begin_time	datetime	在源数据库应用命令的时间 **注意：** 当 ASE 仍在处理用户数据库恢复时，如果执行 **select getdate()**，**select getdate()** 返回的值可能与 rs_origin_begin_time 值不同。
rs_origin_commit_time	datetime	在源数据库提交事务的时间 **注意：** 当 ASE 仍在处理用户数据库恢复时，如果执行 **select getdate()**，**select getdate()** 返回的值可能与 rs_origin_begin_time 值不同。
rs_origin_db	varchar(30)	源数据库的名称
rs_origin_ds	varchar(30)	源数据服务器的名称
rs_origin_ldb	varchar(30)	热备份应用程序的逻辑数据库的名称
rs_origin_lds	varchar(30)	热备份应用程序的逻辑数据服务器的名称
rs_origin_qid	varbinary(36)	事务中第一个命令的源队列 ID
rs_origin_user	varchar(30)	在源执行事务的用户
rs_origin_xact_id	binary(120)	系统指派的唯一事务 ID
rs_origin_xact_name	varchar(30)	用户指派的源事务名称

系统变量	数据类型	说明
rs_repl_objowner	varchar	复制对象的所有者
rs_secondary_qid	varbinary(36)	预订实现或取消实现队列中事务的队列 ID
rs_last_text_chunk	int(4)	如果此值为 0，则不是 text 数据的最后一个大块。如果此值为 1，则是 text 数据的最后一个大块。
rs_writetext_log	int(4)	如果此值为 0，则 rs_writetext 尚未完成在主数据库事务日志中记录 text、unitext 和 image 数据的操作。如果此值为 1，则 rs_writetext 已完成在主数据库事务日志中记录 text、unitext 和 image 数据的操作。

从 DSI 队列中读取较大事务的提交语句之前，如果未使用并行 DSI 来处理这些事务，则 *rs_origin_commit_time* 系统变量值包含在主节点上提交事务组中最后一个事务的时间。

从 DSI 队列中读取较大事务的提交语句之前，如果使用并行 DSI 来处理这些事务，当 DSI 线程开始处理这些事务中的某个事务时，*rs_origin_commit_time* 系统变量的值将被设置为 *rs_origin_begin_time* 系统变量的值。

读取事务的提交语句时，*rs_origin_commit_time* 的值将被设置为实际提交时间。因此，如果将配置参数 **dsi_num_large_xact_threads** 设置为大于零的值，则 *rs_origin_commit_time* 的值对除 rs_commit 以外的所有系统函数都不可靠。

系统变量和 NULL 值

- 以下系统变量可以具有 NULL 值：
  - *rs_origin_ds*
  - *rs_origin_db*
  - *rs_origin_user*
  - *rs_origin_xact_name*
  - *rs_destination_db*
  - *rs_destination_user*
  - *rs_dump_dbname*
  - *rs_dump_label*

  当系统变量没有值时，Replication Server 将字“NULL”映射到函数字符串模板中。这可能会导致某些生成的语句中出现语法错误。例如，如果 *rs_origin_xact_name* 的值为 null 值，则生成以下命令：

  ```
  begin transaction NULL
  ```

  为避免发生这种错误，应使用如下所示的输出模板创建函数字符串：

  ```
  'begin transaction t_?rs_origin_xact_name!sys_raw?'
  ```

如果 *rs_origin_xact_name* 系统变量为空，则事务名称将为“t_NULL”。

替换函数字符串

- 要替换函数字符串，可使用 **alter function string**，或结合使用 **create function string** 和 **overwrite**。这两种方法都会在单个事务中执行 **drop function string** 和 **create function string**，从而避免由于临时缺失函数字符串而发生错误。

权限

create function string 需要“create object”权限。

另请参见

- alter function string （第 143 页）
- configure connection （第 180 页）
- create connection （第 214 页）
- create function string class （第 249 页）
- create subscription （第 280 页）
- define subscription （第 290 页）
- drop function string （第 302 页）

create function string class

创建函数字符串类。

语法

```
create function string class function_class
    [set parent to parent_class]
```

参数

- **function_class** – 要创建的函数字符串类的名称。此名称必须符合标识符规则。函数字符串类的名称具有全局名称空间，因此，在复制系统中必须是唯一的。
- **set parent to** – 为新的派生类指定父类。
- **parent_class** – 被指定为新派生类的父类的现有函数字符串类的名称。**rs_sqlserver_function_class** 不能用作父类。

示例

- **示例 1** – 创建一个名为 *sqlserver_derived_class* 的派生函数字符串类，此类将从系统提供的类 *rs_default_function_class* 继承函数字符串：

```
create function string class
 sqlserver_derived_class
 set parent to rs_default_function_class
```

- **示例 2** – 创建一个名为 *sqlserver2_function_class* 的函数字符串类。此类将成为基类，不会继承函数字符串。但是，可以将此类指定为派生类的父类：

```
create function string class sqlserver2_function_class
```

用法

- 使用 **create function string class** 可创建函数字符串类。函数字符串类用于对数据库的函数字符串进行分组。函数字符串类及其成员函数字符串与数据库相关联。使用 **create connection** 或 **alter connection** 命令可以建立这种关联。
- 将 **create function string class** 发送到的 Replication Server 将成为新创建的函数字符串类的主 Replication Server。
- 使用 **set parent to** 子句创建新的派生类时，可以指定一个父类，新类将从此类中继承函数字符串。省略此子句可以创建一个新的基类，基类不会从父类继承函数字符串。
- 有关函数字符串类、函数字符串和函数的概述，请参见《Replication Server 管理指南第二卷》。
- 执行此命令之前，请确保新函数字符串类的名称在复制系统中是唯一的。Replication Server 不会检测所有的名称冲突。
- Replication Server 通过复制系统将新的函数字符串类分发到合格的节点。通常由于复制系统会有一定时间的滞后，因此更改不会立即显示在所有这些节点上。
- 要修改 *rs_sqlserver_function_class* 类中的函数字符串，必须首先选择一个 Replication Server 作为此类的主节点。然后在此节点上执行 **create function string class**，创建 *rs_sqlserver_function_class*。
- 作为任何函数字符串类的主节点的 Replication Server 必须具有到将使用此类的所有其它 Replication Server 的路由。
- 派生类的主节点和其父类的主节点相同。派生类必须在其父类的主节点上创建。但是，如果父类是系统提供的类 *rs_default_function_class* 或 *rs_db2_function_class*，则派生类的主节点就是创建此派生类的 Replication Server。

系统提供的函数字符串类

- Replication Server 提供了三个可供您使用的函数字符串类：
  - *rs_sqlserver_function_class* – 此类包含生成的缺省 Adaptive Server 函数字符串。*rs_sqlserver_function_class* 中的缺省函数字符串与 *rs_default_function_class* 中的缺省函数字符串相同。缺省情况下，此类指派给通过使用 **rs_init** 添加到复制系统中的 Adaptive Server 数据库。可以为此类自定义函数字符串。*rs_sqlserver_function_class* 不能用作父类或派生类。
  - *rs_default_function_class* – 此类包含生成的缺省 Adaptive Server 函数字符串。*rs_sqlserver_function_class* 中的缺省函数字符串与 *rs_default_function_class* 中的

缺省函数字符串相同。您无法为此类自定义函数字符串。此类可以用作父类，但不能成为派生类。

- *rs_db2_function_class* – 此类包含生成的缺省DB2 专用函数字符串。虽然此类是 *rs_default_function_class* 的派生类（其中带有为 DB2 自定义的内容），但不能为此类自定义函数字符串。*rs_db2_function_class* 可以用作父类，但不能作为派生类。

函数字符串继承的优点

- 使用从系统提供的类 *rs_default_function_class* 或 *rs_db2_function_class* 继承（无论直接还是间接）的派生类时，可以只自定义需要自定义的函数字符串，并继承所有其它字符串，即使是对于新表或新的函数复制定义也是如此。
  如果使用不从系统提供的类继承函数字符串的类，则必须自己在父类或派生类中创建所有的函数字符串，而且，只要创建新表或新的函数复制定义，就需要添加新的函数字符串。
- 升级到 Replication Server 的更高版本后，从系统提供的类 *rs_default_function_class* 或 *rs_db2_function_class* 继承（无论直接还是间接）的派生类将继承所有新系统函数的函数字符串定义。

将函数字符串添加到函数字符串类中

- 创建了不从父类继承函数字符串的函数字符串类后，应为具有函数字符串类范围的系统函数添加函数字符串。然后为具有复制定义范围的系统函数和用户定义函数添加函数字符串，这些函数将被复制到使用新函数字符串类的数据库中。
- 要在函数字符串类中创建或自定义函数字符串，请使用 **create function string**。不能在 *rs_default_function_class* 或 *rs_db2_function_class* 类中创建函数字符串。

权限

create function string class 需要“sa”权限。

另请参见

- alter connection （第 109 页）
- alter function string class （第 145 页）
- create connection （第 214 页）
- create function （第 231 页）
- create function string class （第 249 页）
- move primary （第 318 页）

create logical connection

创建逻辑连接。Replication Server 使用逻辑连接来管理热备份应用程序。

语法

```
create logical connection to data_server.database
[set logical_database_param [to] 'value'
[set logical_database_param [to] 'value' ]...]
```

参数

- **data_server** – 数据服务器的名称。此数据服务器不必是真实的数据服务器。
- **database** – 数据库的名称。此数据库不必是真实的数据库。
- **logical_database_param** – 影响逻辑连接的配置参数的名称。表 19. 影响逻辑连接的配置参数介绍了可使用 **create logical connection** 设置的参数。

示例

- **示例 1** – 创建名为 *LDS.logical_pubs2* 的逻辑连接：

  ```
  create logical connection to LDS.logical_pubs2
  ```

- **示例 2** – 为现有连接创建逻辑连接。例如，如果数据库 *TOKYO_DS.pubs2* 已经存在，并且将作为热备份应用程序中的活动数据库，则应当输入以下命令：

  ```
  create logical connection to TOKYO_DS.pubs2
  ```

用法

- **create logical connection** 创建用于热备份应用程序的逻辑连接。有关设置和管理热备份应用的信息，请参见《Replication Server 管理指南第二卷》。
- 逻辑连接用于符号 *data_server.database* 规格。数据服务器和数据库不一定具有真实性；Replication Server 会将它们映射到当前的活动数据库。
- 如果要为现有连接创建逻辑连接，则 *data_server.database* 必须引用现有连接的数据服务器和数据库名称。否则，建议使用不同于活动和备用数据库名称的逻辑名称。
- 复制定义和预订使用逻辑连接名称。
- 创建逻辑连接后，使用 **rs_init** 为逻辑连接添加实际的活动和备用数据库。

权限

create logical connection 需要“sa”权限。

另请参见

- alter logical connection （第 146 页）
- configure connection （第 180 页）
- configure logical connection （第 180 页）
- drop connection （第 297 页）
- drop logical connection （第 305 页）
- switch active （第 338 页）
- create alternate logical connection （第 205 页）

create partition

使 Replication Server 可以使用某一分区。分区既可以是一个磁盘分区，也可以是一个操作系统文件。

语法

```
create partition logical_name
on 'physical_name' with size size
[starting at vstart]
```

参数

- **logical_name** – 分区的名称。名称必须符合标识符规则。**drop partition** 和 **alter partition** 命令中也会使用此名称。
- **physical_name** – 分区的完整名称。此名称必须用单引号引起来。
- **size** – 分区的大小（以 MB 为单位）。可能的最大大小为 1TB。
- **starting at vstart** – 指定与分区起始处的偏移量 (*vstart*)，以 MB 为单位。

示例

- **示例 1** – 在名为 `/dev/rsd0a` 的设备上添加一个名为 *P1* 的 20MB 分区：

  ```
  create partition P1 on '/dev/rsd0a' with size 20
  ```

- **示例 2** – 在名为 `/dev/rsd0a` 的设备上添加一个名为 *P1* 的 20MB 分区。不过，因为指定了 1MB 的偏移量，所以可供 Replication Server 使用的总可用分区空间为 19MB：

  ```
  create partition P1 on '/dev/rsd0a' with size 20
   starting at 1
  ```

用法

- Replication Server 将分区用于稳定消息队列。消息队列在将数据发送到其目标之前保存数据。
- 增加分区中的可用磁盘空间可使 Replication Server 支持更多的路由和数据库连接，并在更长时间的故障期间能够继续对消息排队。
- 分区的最大大小为 1TB，大约为 1,000,000MB。
- 不能装入磁盘分区以供操作系统使用或用于任何其它用途，例如交换空间或 Adaptive Server 磁盘设备。
- 将整个分区分配给 Replication Server。如果将分区的一部分分配给 Replication Server，则无法将其余部分用于任何其它用途。如果使用 **starting at** *vstart* 子句，Replication Server 可用的分区空间就是总分区大小减去偏移量大小后剩余的部分。
- **starting at** *vstart* 子句使分区起始处的可用空间用于磁盘镜像信息。
- 可以在分区中使用操作系统文件。但是，操作系统会缓冲文件 I/O，这样您可能在发生故障后无法完全恢复稳定队列。因此，您应当仅在测试环境下在分区中使用操作系统文件，除非操作系统不支持物理磁盘分区。
- 如果使用操作系统文件，您必须在执行 **create partition** 之前创建此文件。在 UNIX 平台上，可以使用 **touch** 命令创建此文件。此文件的长度可以为 0 字节；**create partition** 命令可将此文件扩展到指定的 *size*。
- “sybase”用户应当拥有磁盘分区或操作系统文件，并且必须具有对磁盘分区或操作系统文件的读写权限，非“sybase”用户不应当具有对分区的读或写权限。

权限

create partition 需要“sa”权限。

另请参见

- admin disk_space （第 50 页）
- drop partition （第 306 页）
- alter partition （第 149 页）

create publication

为表或存储过程创建发布，以将其作为一组复制到一个或多个预订复制数据库。

语法

```
create publication pub_name
    with primary at data_server.database
```

参数

- **pub_name** – 发布的名称。此名称必须符合标识符规则，而且对于指定的主数据服务器和数据库必须是唯一的。
- **with primary at data_server.database** – 指定主数据的位置。如果主数据库是热备份应用程序的一部分，则 *data_server.database* 是逻辑数据服务器和数据库的名称。

示例

- **示例 1** – 创建名为 *pubs2_pub* 的发布，您可以使用此发布为 *pubs2* 数据库中的多个表和存储过程复制数据。

```
create publication pubs2_pub
 with primary at TOKYO_DS.pubs2
```

用法

- 使用 **create publication** 可创建发布，发布对象可以简化对数据库中多个表和存储过程的复制过程。首先创建发布，然后添加项目（用于指定复制定义），最后再为此发布创建一个预订。
- 应在管理存储主数据的数据库的 Replication Server 上执行 **create publication**。
- 有关使用复制定义、项目和发布的详细信息，请参见《Replication Server 管理指南第一卷》。

  有关预订发布的详细信息，请参见《Replication Server 管理指南第一卷》中的“管理预订”。
- 仅在为发布创建或刷新预订时，Replication Server 才会将有关新发布的信息分发到复制节点。

使用 **create publication** 的要求

- 执行 **create publication** 之前，应确保满足以下条件：
  - 输入的发布名称对于主数据服务器和数据库是唯一的。
  - Replication Server 与存储主表或存储过程的数据库之间有连接。

为预订准备发布

- 创建发布后，使用 **create article** 创建项目并将它们指派给此发布。项目指定表复制定义或函数复制定义，而且根据预订复制节点的需要，还可以包括 **where** 子句。有关详细信息，请参见 create article。
- 由于一个复制表不能预订同一主对象的两个或多个复制定义，因此对于相同的主表和相同的复制表，一个发布不能包含两个或多个具有不同复制定义的项目。
- 指派了所有项目后，必须首先使用 **validate publication** 验证此发布，复制节点才能对其进行预订。在验证发布的过程中，将检验发布是否至少包含一个项目，然后将此发布标记为可以预订。有关详细信息，请参见 validate publication。
- 要检查发布状态，请使用 **check publication**。此命令显示发布包含的项目数，并指出此发布是否有效。有关详细信息，请参见 check publication。

预订发布

- 如果发布有效，可以创建对此发布的预订，以便开始复制到复制数据库。支持所有形式的预订实现。有关详细信息，请参见 create subscription 或 define subscription。
- 创建发布预订时，Replication Server 为此发布包含的每个项目分别创建一个基本预订。每个项目预订使用父发布预订的名称。
- 对发布的预订不能包括 **where** 子句。但是，您可以通过在发布所包含的每个项目中加入一个或多个 **where** 子句，自定义到复制节点的复制。

表复制定义的项目

- 如果发布仅包含表复制定义的项目，则可以使用 **create subscription** 通过原子实现或非原子实现来预订此发布。有关详细信息，请参见 create subscription。
- 也可以对发布预订使用批量实现：
  - 如果复制数据库中已经有数据，请使用带有 **without materialization** 子句的 **create subscription**。
  - 如果必须手动传送预订数据，请使用 **define subscription** 和其它批量实现命令。有关详细信息，请参见 define subscription。

函数复制定义的项目

- 如果发布仅包含函数复制定义的项目，请对此发布预订使用批量实现：
  - 如果复制数据库中已经有数据，请使用带有 **without materialization** 子句的 **create subscription**。有关详细信息，请参见 create subscription。
  - 如果必须手动传送预订数据，请使用 **define subscription**、**activate subscription** 和 **validate subscription** 命令以通过批量实现来预订此发布。有关详细信息，请参见 define subscription。

表复制定义和函数复制定义的项目

- 如果发布中既包含表复制定义的项目，又包含函数复制定义的项目，虽然各种类型的复制定义要求不同的实现方法，但仍可以使用相同的预订命令。
  要创建预订，首先为需要批量实现的组件预订（例如函数复制定义的组件预订），将数据传送到复制数据库。然后使用 **create subscription** 预订此发布：
  - 对表复制定义的项目的预订是通过基本或非基本实现来实现的，除非使用 **without materialization** 子句。
  - 对函数复制定义项目的预订是使用 without materialization 实现的。
    如果函数复制定义的存储过程所操作的表还有一个表复制定义，则无需另外传送数据。

刷新发布预订

- 如果将新项目添加到现有发布，或从现有发布中删除项目，则此发布将失效。虽然现有项目的复制仍不受影响，但要开始新项目的复制或创建新发布预订，必须执行以下操作：

  - 在完成对发布的更改后，验证此发布，然后
  - 刷新此发布预订。
- 要为基本或非原子实现刷新发布预订，请执行以下操作：
  - 使用 **create subscription** 重新创建预订。有关详细信息，请参见 **create subscription**。
- 要为批量实现刷新发布预订，请执行以下操作：
  - 如果复制数据库中已经有数据，请使用带有 **without materialization** 子句的 **create subscription**。
  - 使用 **define subscription**、**activate subscription** 和 **validate subscription** 重新创建预订，并根据需要手动传送预订数据。有关详细信息，请参见 **define subscription**。

删除预订、项目和发布

- 可以删除对发布的预订，对于表复制定义的项目的组件预订，还可以选择清除其预订数据。请参见 **drop subscription**。
- 如果没有预订，则可以删除发布所包含的项目，并且可以选择删除相关联的复制定义（如果未在别处使用）。删除项目后，此发布将失效。请参见 **drop article**。
- 只能删除没有预订的发布。删除发布时，它所包含的项目也将被删除。还可以选择删除发布项目的所有复制定义（如果它们未在别处使用）。有关详细信息，请参见 **drop publication**。

热备份应用程序中的发布

- 在热备份应用程序中，也可以使用发布所包含的项目来指定用于将数据复制到备用数据库的复制定义。

权限

create publication 需要“create object”权限。

另请参见

- check publication （第 176 页）
- create article （第 211 页）
- create applied function replication definition （第 206 页）
- create replication definition （第 258 页）
- create request function replication definition （第 269 页）
- create subscription （第 280 页）
- define subscription （第 290 页）
- drop article （第 296 页）
- drop publication （第 307 页）
- drop subscription （第 312 页）
- validate subscription （第 390 页）

create replication definition

为要复制的表创建复制定义。

语法

```
create replication definition replication_definition
with primary at data_server.database
[with all tables named [table_owner.] 'table_name' [quoted] |
[with primary table named [table_owner.]'table_name']
 with replicate table named [table_owner.]'table_name'] [quoted]]
(column_name [as replicate_column_name] [datatype [null | not null]
        [datatype [null |not null ]
[map to published_datatype]] [quoted]
[, column_name [as replicate_column_name]
        [map to published_datatype]] [quoted]...)
        [references [table_owner.]table_name [(column_name)]]
primary key (column_name [, column_name]...)
[searchable columns (column_name [, column_name]...)]
[send standby [{all | replication definition} columns]]
[replicate {minimal | all} columns]
[replicate {SQLDML [ 'off' ] | 'options' }]
[replicate_if_changed (column_name [, column_name]...)]
[always_replicate (column_name [, column_name]...)]
[with dynamic sql | without dynamic sql]
```

参数

- **replication_definition** – 复制定义，它必须符合标识符的规则。复制定义的名称假定为主表和复制表的名称，除非您指定了表名。
- **with primary at data_server.database** – 指定主数据的位置。如果主数据库是热备份应用程序的一部分，则 *data_server.database* 是逻辑数据服务器和数据库的名称。
- **with all tables named** – 指定主数据库和复制数据库中的表名。*table_name* 是最多含 200 个字符的字符串。*table_owner* 是可选的，表示表的所有者。如果表的实际所有者与复制定义中指定的所有者不相符，数据服务器操作可能会失败。
- **quoted** – 可以使用 **quoted** 参数，将要创建的表或列名指定为带引号的标识符。对于复制需要具有引号的每个对象使用 quoted 子句。
- **指定主表名称** – 指定主数据库中的表名。*table_name* 是最多为 200 个字符的字符串。*table_owner* 是可选的，表示表的所有者。如果表的实际所有者与复制定义中指定的所有者不相符，数据服务器操作可能会失败。

  如果指定了主表的名称，但未指定复制表的名称，则假定复制定义的名称为复制表的名称。
- **with replicate table named** – 指定复制数据库中的表名。*table_name* 是最多为 200 个字符的字符串。*table_owner* 是可选的，表示表的所有者。如果表的实际所有者与复制定义中指定的所有者不相符，数据服务器操作可能会失败。

如果指定了复制表的名称，但未指定主表的名称，则复制定义的名称将假定为主表的名称。

- **column_name** – 主表中的列名。同一列名在每个子句中只能使用一次。

  每一列和每个数据类型必须用小括号 **( )** 括起来。
- **as replicate_column_name** – 指定复制表中的列名，主列中的数据将被复制到此列中。如果源列和目标列的名称不同，则需要使用此子句。
- **数据类型** – 主表中的列的数据类型。有关数据类型和语法的列表，请参见“数据类型”。

  指定列级数据类型转换时，用作 *declared_datatype*。声明的数据类型必须是 Replication Server 的本机数据类型，或者是主数据类型的数据类型定义。

  对于为同一个表创建的不同复制定义，列数据类型必须相同，但发布的数据类型可以不同。有关详细信息，请参见《Replication Server 管理指南第一卷》。

  如果为同一表创建的复制定义中已经有此列，则可以不指定数据类型。
- **null 或 not null** – 仅适用于 *text*、*unitext*、*image* 或 *rawobject* 列。指定复制表中是否允许空值。缺省值为 **not null**，表示复制表不接受空值。

  对于同一主表的所有复制定义，每个 *text*、*unitext*、*image* 和 *rawobject* 列的空值状态必须相匹配，并且必须与实际表中的设置相匹配。如果同一主表的现有复制定义有 *text*、*unitext*、*image* 和 *rawobject* 列，则可以选择是否指定空值状态。

  一旦将某一列包括在表的复制定义中，就不能再更改此列的此项设置。要更改此值，必须删除并重新创建包括此列的所有复制定义。
- **map to published_datatype** – 指定在列级别数据类型转换之后，但在进行类级别转换并提交到复制数据库之前，某一列的数据类型。
- **references table owner.table name column name** – 指定在主数据库中具有参照约束的表的表名。*table_name* 是最多为 200 个字符的字符串。*table_owner* 是可选的，表示表的所有者。*column name* 是可选的。如果表的实际所有者与复制定义中指定的所有者不相符，数据服务器操作可能会失败。
- **primary key column_name** – 指定组成此表主键的列。同一列名在每个子句中只能使用一次。

  不能将 *text*、*unitext*、*image*、*rawobject*、*rawobject in row* 或 *rs_address* 列作为主键的一部分。
- **searchable columns column_name** – 指定可以在 **create subscription**、**define subscription** 或 **create article** 的 **where** 子句中使用的列。同一列名在每个子句中只能使用一次。

  不能将 *text*、*unitext*、*image*、*rawobject*、*rawobject in row* 或加密列指定为可搜索列。
- **send standby** – 指定在将数据复制到热备份应用程序的备用数据库的过程中如何使用复制定义。有关使用此子句及其选项的详细信息，请参见“复制定义和热备份应用程序”。

- **replicate minimal columns 或 replicate all columns** – 发送每个事务的所有复制定义列，或仅发送在复制数据库中执行更新或删除操作时所需的那些复制定义列。缺省设置是 **replicate all columns**。

  注意: 如果复制定义具有 `[replicate {minimal | all} columns]` 子句，则 `[replicate {minimal | all} columns]` 子句必须始终位于 `[replicate {SQLDML ['off'] | 'options'}]` 子句前面。

- **replicate SQLDML [‘off’]** – 打开或关闭指定的 DML 操作的 SQL 复制。
- **replicate ‘options’** – 复制以下 DML 操作的任意组合:
  - U – **update**
  - D – **delete**
  - I – **insert select**
- **replicate_if_changed** – 仅当 *text*、*unitext*、*image* 或 *rawobject* 列的数据发生更改时，才复制这些列。
- **always_replicate** – 始终复制 *text*、*unitext*、*image* 和 *rawobject* 列。
- **with dynamic sql** – 指定在命令符合条件并且有足够的高速缓存空间时 DSI 将动态 SQL 应用到表中。这是缺省值。

  有关命令必须满足哪些条件才能应用动态 SQL 的信息，请参见《Replication Server 管理指南第二卷》。
- **without dynamic sql** – 指定 DSI 不能使用动态 SQL 命令。

示例

- **示例 1** – 为 *authors* 表创建名为 *authors_rep* 的复制定义。*authors* 表的主副本在 LDS 数据服务器的 *pubs2* 数据库中。此表所有副本的名称均为 *authors*。仅复制删除和更新操作所需的最少数量的列:

```
create replication definition authors_rep
  with primary at LDS.pubs2
  with all tables named 'authors'
    (au_id varchar(11), au_lname varchar(40),
     au_fname varchar(20), phone char(12),
     address varchar(12), city varchar(20),
     state char(2), country varchar(12), postalcode
     char(10))
  primary key (au_id)
  searchable columns (au_id, au_lname)
  replicate minimal columns
```

- **示例 2** – 为 *pubs2* 数据库中属于“emily”的 *blurbs* 表创建名为 *blurbs_rep* 的复制定义。列中的数据更改时，将复制数据类型为 *text* 并接受空值的 *copy* 列中的数据:

```
create replication definition blurbs_rep
  with primary at TOKYO_DS.pubs2
  with all tables named emily.'blurbs'
    (au_id char(12), copy text null)
```

```
  primary key (au_id)
  replicate_if_changed (copy)
```

- **示例 3** – 对于 *pubs2* 数据库中的主表 *publishers*，至少已经存在一个复制定义时，此命令将创建另一个名为 *pubs_copy_rep* 的复制定义。此复制定义可以由名为 *pubs_copy* 且属于“joe”的复制表预订。对于名称也是 *pubs_copy*，但不属于“joe”的复制表，预订可能会失败：

```
create replication definition pubs_copy_rep
  with primary at TOKYO_DS.pubs2
  with primary table named 'publishers'
  with replicate table named joe.'pubs_copy'
    (pub_id, pub_name as pub_name_set)
  primary key (pub_id)
```

  主表中 *pub_name* 列的数据将复制到复制表的 *pub_name_set* 列中，它们的数据类型必须相同。无需为现有复制定义中的列指定数据类型。在此示例中，由于主表中的 *city* 和 *state* 列对于复制表 *pubs_copy* 不是必需的，因此复制定义中未包括这两列。

- **示例 4** – 创建一个复制定义，以将所有修改的 *authors* 表列复制到备用数据库。此定义还会复制到 MSA，但只复制修改的 *au_id* 和 *au_lname* 列值。*au_id* 是在 *authors* 表中执行更新和删除操作时使用的键：

```
create replication definition authors_rep
  with primary at LDS.pubs2
  with all tables named 'authors'
     (au_id varchar(11), au_lname varchar(40))
  primary key (au_id)
  send standby
  replicate minimal columns
```

- **示例 5** – 创建 *foo* 表，其中的 *foo_col1* 列为带引号的标识符：

```
create replication definition repdef
   with primary at primaryDS.primaryDB
   with all tables named “foo”
   (“foo_col1” int quoted, “foo_col2” int)
   primary key (“foo_col1”)
```

- **示例 6** – 创建一个表复制定义，用于复制 **update** 和 **delete** 语句：

```
create replication definition repdef1
       with primary at ds3.pdb1
       with all tables named 'tb1'
         (id_col int, str_col char(40))
       primary key (id_col)
       replicate all columns
       replicate ‘UD’
go
```

用法

- 在管理存储主表的数据库的 Replication Server 上执行此命令。

- 使用 **rs_helprep** 可以确定哪些复制定义可用于 Replication Server 12.0 版和更高版本。有关详细信息，请参见 **rs_helprep**。
- 有关定义和维护复制表的概述以及使用复制定义、项目和发布的信息，请参见《Replication Server 管理指南第一卷》。
- 执行 **create replication definition** 命令之前，请确保满足以下条件：
  - 输入的复制定义名称在复制系统中的所有复制定义（表或函数）中是唯一的。输入 **create replication definition** 时，Replication Server 不能始终检查是否满足此要求。
  - 存在 Replication Server 到存储主表的数据库的连接。有关详细信息，请参见 **create connection**。也可以使用 **rs_init** 来创建数据库连接。请参见适用于您的平台的 Replication Server 安装和配置指南。
  - 如果使用 Replication Server 的多个版本（例如，版本 12.0 和版本 11.0.x），并且为同一主表创建了多个复制定义，则应当检查复制系统中所有与混合版本相关的问题（例如，同一表的列名在两个版本中是否不同）。有关详细信息，请参见“创建多个复制定义”。
- Replication Server 通过复制系统将新的复制定义分发到合格的节点。由于通常复制系统会有一定时间的滞后，因此更改不会立即显示在所有这些节点上。
- 为主数据库创建的复制定义应用于控制复制定义的 Replication Server 和主数据库之间的所有主连接（缺省连接和替代连接）。因此，必须先删除主数据库的所有复制定义才能删除到主数据库的最后一个主连接逻辑连接。

  使用 1570 版本的系统时，您只能对数据库创建复制定义和发布。为 **create replication definition** 命令的 **with primary at** 子句指定的名称必须是主数据库名称。

复制状态

- *text*、*unitext*、*image* 和 *rawobject* 列的复制状态在 Adaptive Server 数据库和复制定义中必须相同。
- 使用 **alter replication definition** 可以更改复制状态。
- 对于为同一主表创建的所有复制定义，复制状态必须一致。
  - 如果使用 **alter replication definition** 来更改复制状态，则同一主表的其它复制定义的复制状态也将更改。
  - 如果此列已经在同一主表的另一个复制定义中列出，则无需指定复制状态。

创建多个复制定义

- 可以为同一主表创建多个复制定义并对每个复制定义进行自定义，从而使特性不同于主表和其他复制表的复制表也可以预订此主表。

  除描述主表以外，每个复制定义还可以为复制表指定一些内容，例如，更少的列、不同的列名或不同的表名。符合指定特性的复制表可以预订复制定义。即使复制表和主表匹配，也可以使用多个复制定义。

  利用此功能，您可以在数据库要求不同的情况下，为常规复制创建一个复制定义，为备用复制创建另一个复制定义。有关详细信息，请参见《Replication Server 管理指南第一卷》。

- 对于每个主表，一个复制表只能预订一个复制定义，但可以多次预订同一复制定义。
- 对于 *text*、*unitext* 和 *image* 列，为同一主表创建的不同复制定义必须使用相同的列数据类型和相同的空值状态。
- 如果使用内部复制定义将某个表复制到备用或 MSA 连接，并为连接启用了动态 SQL，则该表的任何新复制定义应定义与主数据库中列顺序相一致的列顺序。否则，这可能会使现有预准备语句无效，并且可能需要重新启动备用或 MSA 连接。

函数和函数字符串

- Replication Server 为复制定义创建 **rs_insert**、**rs_delete**、**rs_update**、**rs_truncate**、**rs_select** 和 **rs_select_with_lock** 函数。如果复制定义包含 *text*、*unitext*、*image* 或 *rawobject* 数据，Replication Server 还创建 **rs_datarow_for_writetext**、**rs_get_textptr**、**rs_textptr_init** 和 **rs_writetext** 函数。
- Replication Server 会为系统提供的函数字符串类以及从这些类继承的派生类中的这些函数生成缺省的函数字符串。某些函数字符串可能是动态生成的，因此，它们从不存在于 RSSD 中。对于其它函数字符串类，必须创建所有函数字符串。
- 对于每个函数字符串类，相同表的每个复制定义都有一组各自的系统函数字符串。
- 创建、删除或更改用户定义的函数时，将为相同主表的所有复制定义创建、删除或更改这些函数字符串。
- 虽然同一主表的不同复制定义共享相同的用户定义的函数，但是每个用户定义的函数都有其自己的函数字符串。使用与表复制定义相关联的方法复制存储过程时，可以通过 **create function** 来创建用户定义的函数。

指定列和数据类型

- 指定要复制的列和数据类型时，应遵循以下原则：
  - 具有用户定义的数据类型的列，在复制定义中必须使用基本数据类型来定义。
  - 对于主表的所有复制定义，*text*、*unitext*、*image* 或 *rawobject* 列的复制状态（即 **replicate_if_changed**、**always_replicate**）必须相同。如果使用 **alter replication definition** 更改 *text*、*unitext*、*image* 或 *rawobject* 列的复制状态，则对于同一主表的其它复制定义，此列的复制状态也将更改。

    对于属于同一表的某个复制定义的一部分的 *text*、*unitext*、*image* 或 *rawobject* 列，无需指定其复制状态。
  - 在 *numeric* 数据类型声明中省略长度和精度。Replication Server 在处理数据类型为 *numeric* 的值时，不会影响精度。

    注意： 如果使用 **map to** 选项将较大的 **varchar** 转换为每列字符较少的 **varchar**，请确保复制的任何数据不超过要复制到目标列的字符长度。

    例如，您可以将 **varchar(100)** 映射为 **varchar(25)** 列，只要复制的项目不超过 **varchar(25)** 的限制。否则将会出现错误消息。
  - 如果复制定义列的列表包含 IDENTITY 列并且复制表位于 Adaptive Server 中，则维护用户在复制数据库中必须是此表的所有者（或者必须是“dbo”或别名为“dbo”），才能使用 Transact-SQL 的 **identity_insert** 选项。

一个主表（带有一个或多个复制定义）只能包含一个 IDENTITY 列。但是，您可以使用 **map to** 选项在一个或多个复制定义中将多个列发布为 *identity* 数据类型。

- 如果复制定义列的列表包含 *timestamp* 列并且复制表位于 Adaptive Server 中，则维护用户在复制数据库中必须是此表的所有者（或者必须是“dbo”或别名为“dbo”）。

  一个主表（带有一个或多个复制定义）只能包含一个 *timestamp* 列。但是，您可以使用 **map to** 选项在一个或多个复制定义中将多个列发布为 *timestamp* 数据类型。
- *rs_address* 数据类型提供了一种唯一预订解析方法。将 *rs_address* 数据类型（基于基本的 *int* 数据类型）的位图与预订的 **where** 子句中的位屏蔽进行比较，以确定是否应复制某一行。要使用这种预订解析方法，必须首先创建列数据类型为 *int* 的表。创建复制定义时，将这些列加入到列的列表中，但将其数据类型声明为 *rs_address*，而不是 *int*。

  有关详细信息，请参见 **create subscriptions**。另请参见《Replication Server 管理指南第一卷》以了解使用 *rs_address* 数据类型的详细信息。

为列级别转换指定列和数据类型

- 不能将 *text*、*unitext*、*image* 或 *rawobject* 数据类型作为基本数据类型或数据类型定义使用，也不能将其作为列级别转换或类级别转换的源或目标使用。
- *declared_datatype* 取决于传递给 Replication Server 的值的数据类型：
  - 如果 Replication Agent 传递的是 Replication Server 本机数据类型，则 *declared_datatype* 为此本机数据类型。
  - 如果 Replication Agent 传递的是任何其它数据类型，则 *declared_datatype* 必须是主数据库中原始数据类型的数据类型定义。
- *published_datatype* 是在列级别转换后，但尚未进行任何类级别转换前值的数据类型。*published_datatype* 必须是 Replication Server 本机数据类型或目标数据库中数据类型的数据类型定义。
- 在多个复制定义中声明的列所使用的 *declared_datatype* 在每个复制定义中必须相同。但 *published_datatype* 可以不同。

使用 **replicate minimal columns** 选项

- 使用 **replicate minimal columns** 选项可以提高 DSI 性能、减少消息开销并缩小队列大小。这还有助于避免由于为实际上并未发生更改的列设置触发器而引发的应用问题。

  注意： 如果复制定义具有 **replicate all columns** 并且 **replicate minimal columns** 连接属性设置为“on”，连接将复制最少列。如果想要将所有列复制到目标数据库，则将 DSI 连接的 **replicate minimal columns** 值设置为“off”。

  有关此选项的工作方式的详细信息，请参见《Replication Server 管理指南第二卷》。
- 下列要求适用于复制最少的列：

- 通常，**replicate minimal columns** 只能用于使用 **rs_update** 和 **rs_delete** 函数的缺省函数字符串的复制定义。如果指定 **replicate minimal columns**，则可以在函数字符串中使用 *rs_default_fs* 系统变量为复制定义创建非缺省的 **rs_update** 和 **rs_delete** 函数字符串。有关详细信息，请参见 **create function string**。
- 使用 **replicate minimal columns** 选项时，不能同时使用自动更正功能。如果在设置 **replicate minimal columns** 之前指定 **set autocorrection on**，将为每个删除或更新操作记录一条信息性消息。如果首先指定 **replicate minimal columns**，则不能为复制定义指定 **set autocorrection on**。
- 如果已经为复制定义指定了 **replicate minimal columns**，则不能使用非原子实现（**create subscription** 命令和 **without holdlock** 选项）为其创建预订，也不能使用模拟非原子实现的批量实现选项。有关详细信息，请参见《Replication Server 管理指南第二卷》。

复制 `text`、`unitext`、`image` 或 `rawobject` 数据类型

- 复制定义的主键必须包括可唯一标识表中的单个行的列。
- 使用 **always_replicate** 和 **replicate_if_changed** 子句可以指定 *text*、*unitext*、*image* 和 *rawobject* 列的复制状态。也可以在 Adaptive Server 系统过程 **sp_setreptable** 和/或 **sp_setrepcol** 或者 **sp_reptostandby** 中设置此状态。复制状态在 Adaptive Server 系统过程和主表的复制定义中必须相同。如果不一致，RepAgent 可能会关闭。有关设置状态和解决不一致问题（如果发生）的信息，请参见《Replication Server 管理指南第一卷》。有关将 *text*、*unitext*、*image* 和 *rawobject* 数据复制到热备份应用程序的信息，请参见“复制定义和热备份应用程序”。
- 当仅将表标记为 **sp_setreptable** 时，必须将复制定义的复制状态指定为 **always_replicate**，因为 **sp_setreptable** 缺省复制状态是 **always_replicate**。可以将表的复制状态更改为 **replicate_if_changed**，方法是将表的复制定义的复制状态更改为 **replicate_if_changed**，并对表中的每一列进行标记，将 **sp_setrepcol** 复制状态设置为 **replicate_if_changed**。
- 以下是复制 *text*、*unitext*、*image* 或 *rawobject* 数据类型时应满足的要求：
  - 如果 *text*、*unitext*、*image* 或 *rawobject* 列出现在 **replicate_if_changed** 列的列表中，尝试为复制定义启用自动更正功能将会引发错误。自动更正要求所有 *text*、*unitext*、*image* 和 *rawobject* 列出现在复制定义的 **always_replicate** 列表中。
  - 如果在主表中执行更新操作时未更改状态为 **replicate_if_changed** 的 *text*、*unitext*、*image* 或 *rawobject* 列，而更新操作导致行迁移到预订中，则在复制表中插入的行中的 *text*、*unitext*、*image* 或 *rawobject* 数据将会丢失。当行迁移到预订中，但 *text*、*unitext*、*image* 或 *rawobject* 数据丢失时，Replication Server 将在错误日志中显示一条警告消息。在这种情况下，运行 **rs_subcmp** 可调和复制表和主表中的数据。

复制定义和热备份应用程序

- Replication Server 不要求使用复制定义来维护热备份应用程序中的备用数据库。使用复制定义可以提高复制到备用数据库操作的性能。可以只为实现此目的而为逻辑数据库中的每个表创建复制定义。

- 使用带有任意选项的 **send standby**，可以使用复制定义将表的事务复制到备用数据库。复制定义的主键列和 **replicate minimal columns** 设置用于将数据复制到备用数据库。此方法的选项包括：
  - 使用 **send standby** 或 **send standby all columns** 可将表中的所有列复制到备用数据库。
  - 使用 **send standby replication definition columns** 可只将复制定义的列复制到备用数据库。
- 在 **alter replication definition** 中使用 **send standby off** 表示在复制到备用数据库的操作中不会使用此表的任何一个复制定义。

  如果主表的所有复制定义都未标记为由备用数据库使用，则所有列都将被复制到备用数据库，此表所有复制定义的所有主键将合并在一起用于主键，并且复制最少列数。逻辑连接的 **replicate_minimal_columns** 设置确定是发送最少数量的列还是所有列以进行更新和删除。有关详细信息，请参见 alter logical connection 和 alter replication definition。
- 有关在复制到备用数据库的操作中使用复制定义所带来的性能优化的详细信息，请参见《Replication Server 管理指南第二卷》。
- 对于具有多个复制定义的主表，如果某个复制定义已被标记为由备用数据库使用，则使用 **send standby** 创建或更改的另一个复制定义将取消第一个复制定义的标记。
- 如果仅将某个数据库标记为 **sp_reptostandby**，则必须将复制定义的复制状态指定为 **replicate_if_changed**，因为 **sp_reptostandby** 缺省复制状态是 **replicate_if_changed**。如果仅将数据库标记为 **sp_reptostandby**，则不能更改 *text*、*unitext*、*image* 和 *rawobject* 列的复制状态。
- 如果将某个数据库标记为 **sp_reptostandby**，而将此数据库中的某个表标记为 **sp_setreptable**，则必须将复制定义的复制状态指定为 **always_replicate**，因为缺省复制状态是 **always_replicate**。可以将表的复制状态更改为 **replicate_if_changed**，方法是将表的复制定义的复制状态更改为 **replicate_if_changed**，并对表中的每一列进行标记，将 **sp_setrepcol** 复制状态设置为 **replicate_if_changed**。

更改复制定义

- 使用 **alter replication definition** 可以添加更多列或更多可搜索列，并对现有复制定义的设置进行其它更改。有关详细信息，请参见 alter replication definition。
- 如果需要删除或重命名现有复制定义中的主列，则必须删除对此复制定义的所有预订，删除此复制定义，然后再重新创建，并重新创建预订。

复制存储过程

- 要启用存储过程的复制，请使用 **create applied function replication definition** 或 **create request function replication definition**。有关复制存储过程的概述，请参见《Replication Server 管理指南第一卷》。

复制计算列

- **create replication definition** 支持复制已实现的计算列。您需要使用复制定义中的基本数据类型来定义实现计算列。

- 实现计算列将其值存储在表页中，这一点与常规列相同。每次插入或更新其基列后，将对其进行重新求值。不会在查询中对其进行重新求值。
- 另一种类型的计算列称为虚拟计算列或非实现计算列。这种计算列的值不存储在表或索引中。如果在查询中引用该列，并且在插入或更新操作后没有执行任何操作，此时才会对其进行求值。

  不支持复制虚拟计算列，因此不应将其包括在复制定义中。

有关复制计算列的详细信息，请参见《Replication Server 管理指南第一卷》。

使用带引号的标识符

- 对于复制需要具有引号的每个对象使用 quoted 子句。如果使用 **quoted** 参数标记标识符，并且将预订复制定义的复制服务器的 **dsi_quoted_identifier** 设置为 on，复制服务器则会以带引号的标识符形式接收标记的标识符。如果 **dsi_quoted_identifier** 为 off，则会忽略标记，复制服务器不会收到带引号的标识符。
- 如果复制到热备份数据库和复制定义预订者，并且将主表名称标记为带引号的标识符但未标记复制表名称（或相反），则 Replication Server 将主表名称和复制表名称均作为带引号的标识符发送。
- 不支持标识符中嵌入的双引号字符。
- 在长度、特殊字符以及支持的保留字方面，Adaptive Server、SQL Anywhere、Microsoft SQL Server、Universal Database (UDB) 以及 Oracle 等数据服务器处理带引号的标识符的方式并不相同。在异构环境中，您必须确保所复制的带引号标识符在主数据服务器和复制数据服务器上均有效。
- 要成功复制带引号的标识符，连接到复制数据服务器的主 Replication Server 和 Replication Server 版本必须为 15.2 和更高版本。不过，路由中的中间 Replication Server 可以为较低的版本。

复制 SQL 语句

- 包含 **send standby** 子句的表复制定义可以指定 **replicate 'I'** 语句。只能在热备份或 MSA 环境中将 **insert select** 语句作为 SQL 复制语句进行复制。不包含 **send standby** 子句的表复制定义无法复制 **insert select** 语句。
- 缺省情况下，热备份应用程序不会复制支持 SQL 语句复制的 DML 命令。要使用 SQL 复制，您可以：
  - 使用 **replicate SQLDML** 和 **send standby** 子句创建表复制定义。
  - 将 **WS_SQLDML_REPLICATION** 参数设置为 on。缺省值为 **UDIS**。不过，**WS_SQLDML_REPLICATION** 的优先级比 SQL 复制的表复制定义低。如果表的表复制定义包含 **send standby** 子句，该子句将确定是否复制 DML 语句，而无论 **WS_SQLDML_REPLICATION** 参数设置是什么。
- SQL 语句复制无法执行自动更正。如果数据服务器接口 (DSI) 在 SQL 语句复制过程中遇到 DML 命令，并且将 auto-correction 设置为 on，则会在缺省情况下挂起 DSI 并停止复制。请将 **assign action** 命令与错误号 5193 一起使用，以指定 Replication Server 如何处理这种错误。

  在验证表级预订后，Replication Server 才会复制 SQLDML。

- 在下列情况下，不支持 SQL 语句复制：
  - 复制数据库的表模式与主数据库不同。
  - Replication Server 必须执行数据或模式转换。
  - 预订包含 **where** 子句。
  - 更新包含一个或多个 *text* 或 *image* 列。

处理具有参照约束的表

对于带有 **reference** 子句的 **alter replication definition** 和 **create replication definition**，Replication Server 会：

- 将 **reference** 子句视为列属性。每个列只能引用一个表。
- 不处理您在 **reference** 子句中的 *column_name* 参数中提供的列名。
- 不允许具有循环引用的参照约束。例如，原始被引用表不能具有对原始引用表的参照约束。

在复制处理过程中，RTL 装载：

- 对您在复制定义中指定的引用表之前的被引用表执行的插入操作。
- 对您在复制定义中指定的表之后的被引用表执行的删除操作。

在一些情况下，对两个表执行的更新操作因冲突而失败。为防止 RTL 重试复制处理并因而降低性能，您可以：

- 通过将 **dsi_command_convert** 设置为“u2di”（这将更新转换为删除和插入），停止复制更新。
- 关闭 **dsi_compile_enable** 以避免编译受影响的表。

RTL 无法编译因而无法标出具有自定义函数字符串的表和具有对它无法编译的现有表的参照约束的表。通过标记出这些表，RTL 避免了由于参照约束错误引发的事务重试，从而优化了复制处理。

权限

create replication definition 需要“create object”权限。

另请参见

- alter function string （第 143 页）
- alter replication definition （第 152 页）
- configure logical connection （第 180 页）
- create connection （第 214 页）
- create applied function replication definition （第 206 页）
- create request function replication definition （第 269 页）
- create function string （第 236 页）
- create subscription （第 280 页）
- drop replication definition （第 308 页）

- rs_set_quoted_identifier （第 430 页）
- set （第 328 页）
- sp_setrepcol （第 483 页）
- sp_setreptable （第 493 页）
- rs_send_repserver_cmd （第 546 页）

create request function replication definition

为要复制的存储过程创建请求函数复制定义和用户定义的函数。请求函数由在主数据库中执行存储过程的同一用户在复制数据库中应用。

语法

```
create request function replication definition repdef_name
        with primary at dataserver.database
        with primary function named 'func_name'
        with replicate function named 'func_name'
        ([@param_name datatype [, @param_name datatype]…])
        [searchable parameters (@param_name [, @param_name]…)]
         [send standby {all | replication definition} parameters]
```

参数

- **repdef_name** – 函数复制定义名称。名称必须符合标识符规则。
- **with primary at** – 指定包含主数据的数据服务器和数据库。
- **dataserver** – 包含主数据的数据服务器的名称。如果主数据库是热备份应用程序的一部分，则 *dataserver* 是逻辑数据服务器名称。
- **database** – 包含主数据的数据库的名称。如果主数据库是热备份应用程序的一部分，则 *database* 是逻辑数据库名称。
- **with primary function named** – 指定主数据库中的存储过程名称。如果不指定主函数名称，Replication Server 会使用复制定义名称作为主函数的名称。主函数名称不得与在 **with replicate function named** 子句中指定的复制函数名称相同。
- **'func_name'** – 函数的名称，最多包含 255 个字符。
- **with replicate function named** – 指定要在复制数据库中执行的存储过程的名称。如果不指定复制函数名称，Replication Server 会使用复制定义名称作为复制函数的名称。复制函数名称不得与在 **with primary function named** 子句中指定的主函数名称相同。

注意： 主存储过程是指最初由客户端调用的存储过程，而复制存储过程是指从主数据库复制的存储过程，该存储过程由复制 Replication Server 调用。

此请求函数行为与 Replication Server 15.0.1 及更低版本中的请求函数行为不同。有关 Replication Server 15.0.1 和更低版本中的请求函数行为的详细信息，请参见《Replication Server 管理指南第二卷》。

- **@param_name** – 函数的参数名。参数名在每个子句中最多只能出现一次。不要求包括参数及其数据类型，但是不管是否包括参数，都必须 包括一对小括号。
- **数据类型** – 函数中参数的数据类型。有关数据类型及其语法的列表，请参见“数据类型”。Adaptive Server 存储过程和函数复制定义不能包含数据类型为 *text*、*unitext*、*rawobject* 和 *image* 的参数。
- **searchable parameters** – 指定可在 **define subscription**、**create subscription** 或 **create article** 的 **where** 子句中使用的参数的列表。如果包括此子句，则必须包括小括号 **( )**。
- **send standby** – 在热备份应用程序中，指定是将此函数中的所有参数都发送到备用数据库 (**send standby all parameters**)，还是只将复制定义中指定的参数发送到备用数据库 (**send standby replication definition parameters**)。缺省值是 **send standby all parameters**。

示例

- **示例 1** – 为名为 **upd_titles_prim** 的函数创建一个名为 **titles_frep** 的请求函数复制定义。要在目标数据库中调用的存储过程名为 **upd_titles:**

```
create request function replication definition titles_frep
with primary at LDS.pubs2
with primary function named 'upd_titles_prim'
with replicate function named 'upd_titles'
(@title_id varchar(6), @title varchar(80), @type char(12), @pub_id
char(4),
     @price money, @advance money, @total_sales int)
searchable parameters (@title_id, @title)
```

用法

- 使用 **create request function replication definition** 可描述要复制的存储过程。应用函数复制定义与请求函数复制定义之间的区别是：通过应用函数复制定义复制的函数由维护用户在复制节点上执行，而通过请求函数复制定义复制的函数由在主节点上执行主函数的同一用户在复制节点上执行。有关复制存储过程的概述，请参见《Replication Server 管理指南第一卷》。
- 为主函数创建请求函数复制定义时，请确保此函数中没有同时满足以下两个条件的现有函数复制定义：
  - 函数复制定义是使用 **create function replication definition** 命令创建的
  - 未在 Replication Server 15.0.1 和更低版本中预订函数复制定义，便将函数复制定义用于请求函数复制

  如果满足这两个条件，则会禁用现有请求函数复制定义。有关 Replication Server 15.0.1 和更低版本中的请求函数复制定义的详细信息，请参见《Replication Server 管理指南第二卷》。
- 在对存储主存储过程的数据库进行管理的 Replication Server 上执行 **create request function replication definition** 命令。
- 执行 **create request function replication definition** 之前，请确保满足以下条件：

  - 函数复制定义名称在复制系统中是唯一的。使用 **create request function replication definition** 时，Replication Server 无法始终强制执行此要求。
  - Replication Server 与存储主数据的数据库之间已建立连接。请参见 **create connection**。您还可以使用 **rs_init** 创建连接；请参见适用于您所用平台的《Replication Server 安装指南》和《Replication Server 配置指南》。
  - 为函数复制定义指定的名称、参数和数据类型必须与所涉及的存储过程的名称、参数和数据类型相匹配。只有在函数复制定义中指定的参数才会得到复制。
- Replication Server 通过复制系统将新的函数复制定义分发到合格的节点。通常由于复制系统会有一定时间的滞后，因此更改不会立即显示在所有这些节点上。

用户定义的函数和函数字符串

- 创建请求函数复制定义时，Replication Server 会自动创建一个相应的用户定义的函数。同样，在 **rs_sqlserver_function_class** 中，Replication Server 会自动为用户定义的函数创建一个缺省函数字符串。
- 您可以使用 **create function string** 自定义 **rs_sqlserver_function_class** 中和用户定义的函数字符串类中的函数字符串。
- 对于用户创建的每个基本函数字符串类（将在其中使用用户定义的函数）以及从这些类继承的每个派生类，必须使用 **create function string** 来创建函数字符串。函数字符串应当使用适合于复制数据服务器的语言来调用存储过程或 RPC。
- 有关函数字符串类、函数字符串和函数的概述，请参见《Replication Server 管理指南第二卷》。

with primary at 子句

使用 **with primary at** 子句可指定主数据服务器和主数据库。主数据库是指包含调用的主存储过程的数据库。

with replicate function named 子句

使用 **with replicate function named** 子句可指定要在目标数据库（在此数据库中传递复制函数）中执行的存储过程的名称。如果创建或更改函数复制定义时不使用 **with replicate function named**，则会将函数作为与函数复制定义同名的存储过程进行传递。在热备份数据库和活动数据库中，存储过程具有相同的名称，因而会忽略 **with replicate function named**。

利用往返复制，一个数据库可以向另一个数据库发送数据更改请求并将数据更改复制回请求数据库中。有关如何使用应用函数复制定义和请求函数复制定义设置往返复制的详细信息，请参见《Replication Server 管理指南第一卷》。

HDS 参数的请求函数复制定义

虽然无法创建可更改参数值数据类型的函数复制定义，但可以使用 HDS 数据类型定义声明请求函数复制定义的参数。声明的参数将受到类级别转换的限制。

有关 HDS 的详细信息，请参见《Replication Server 管理指南第一卷》。

更改函数复制定义

- 使用 **alter request function replication definition** 可将参数或可搜索参数添加到现有请求函数复制定义中。您也可以为函数指定另一个复制名称。
- 要删除或重命名函数复制定义中的参数，请删除对函数复制定义的所有预订。删除预订后，请删除函数复制定义，然后再重新创建该定义。

预订函数复制定义

要预订请求函数复制定义，请使用带有 **without materialization** 子句的 **create subscription**，或使用 **define subscription** 以及其它涉及批量实现的命令。

创建多个复制定义

- 您可以为同一主函数创建多个请求函数复制定义，并对每一个请求函数复制定义进行自定义，以便不同的复制函数可以对其进行预订。有关详细信息，请参见《Replication Server 管理指南第一卷》。
- 为同一主函数创建的不同请求函数复制定义必须 使用具有相同名称的相同参数以及相同的数据类型。
- 请求函数复制定义只能由 Replication Server 15.1 版预订。
- 同一主函数可以拥有应用函数复制定义或请求函数复制定义，但不能同时拥有这两种定义。在创建函数的主 Replication Server 中，会将使用 **create function replication definition** 命令创建的函数复制定义视为应用函数。
- 在热备份数据库和活动数据库中，存储过程具有相同的名称，因而会忽略 **with replicate function named** 子句。如果使用 **send standby replication definition parameters** 子句创建某一个请求函数复制定义，则会将在函数复制定义中指定的参数传递到备用数据库中。否则，将会传递主函数中的所有参数。
- 在 MSA 环境中，如果使用 **send standby** 子句创建的主函数没有函数复制定义，则传递到复制数据库的函数与主函数同名，且具有主函数的所有参数。否则，传递到复制数据库的函数会使用在该函数复制定义的 **with replicate function named** 子句中指定的名称，并包括在同一函数复制定义中指定的参数。

权限

create request function replication definition 需要“create object”权限。

另请参见

- alter applied function replication definition （第 107 页）
- alter function string （第 143 页）
- alter request function replication definition （第 159 页）
- create applied function replication definition （第 206 页）
- create connection （第 214 页）
- create function string （第 236 页）
- define subscription （第 290 页）
- drop function replication definition （第 301 页）
- sp_setrepproc （第 491 页）

- rs_send_repserver_cmd （第 546 页）

create route

指定用于当前 Replication Server 到远程 Replication Server 的连接的路由。

语法

```
create route to dest_replication_server {
 set next site [to] thru_replication_server  |
 with primary at dataserver.database |
 [set username [to] user]
 [set password [to] passwd]
 [set route_param to 'value'
 [set route_param to 'value']... ]
 [set security_param to 'value'
 [set security_param to 'value']... ]}
```

参数

- **dest_replication_server** – 目标 Replication Server。
- **thru_replication_server** – 中间 Replication Server，通过它向目标 Replication Server 传递消息。创建间接路由时，应指定此选项。
- **with primary** – 指定您要为其创建专用路由的主数据库连接。

  注意： 只有当两个 Replication Server 之间具有直接路由时，才能在这两个 Replication Server 之间创建专用路由。如果 Replication Server 之间只有间接路由，则无法创建专用路由。
- **user** – 登录到目标 Replication Server 时使用的 Replication Server 登录名。这是 RSI 用户线程使用的登录名。如果未输入用户名，Replication Server 将使用在启动 Replication Server 时通过 **-S** 标志输入的主体用户名。
- **passwd** – 与登录名一起使用的口令。如果未输入口令，Replication Server 将使用 null 值。
- **route_param** – 影响路由的参数。有关参数和值的列表，请参见“影响路由的配置参数”。
- **value** – 包含参数值的字符串。
- **security_param** – 指定安全性参数的名称。有关可以使用 **create route** 设置的安全性参数的列表和说明，请参见表 33. 影响基于网络的安全性的参数。

表 33. 影响基于网络的安全性的参数

security_param	*值*
msg_confidentiality	指示 Replication Server 是否发送和接收加密数据。如果设置为“required”，将对出站数据进行加密。如果设置为“not required”，则无论入站数据是否加密，Replication Server 均会接受。 缺省值：not_required
msg_integrity	指示是否检查数据的篡改情况。 缺省值：not_required
msg_origin_check	指示是否应当检验数据源。 缺省值：not_required
msg_replay_detection	指示是否应当检查数据以确保其未被读取或截取。 缺省值：not_required
msg_sequence_check	指示是否应当检查数据截取情况。 缺省值：not_required
mutual_auth	要求远程服务器在建立连接之前提供身份证明。 缺省值：not_required
security_mechanism	为路径启用的第三方安全性机制的名称。 缺省值：在 `libtcl.cfg` 的 SECURITY 部分列出的第一种机制
unified_login	指示 Replication Server 如何尝试登录到远程数据服务器并接受入站登录。值为： • required – 始终尝试使用认证登录到远程服务器。 • not_required – 始终尝试使用口令登录到远程服务器。 缺省值：not_required
use_security_services	指示 Replication Server 是否使用安全服务。如果 **use_security_services** 为“off”，则不使用任何安全性功能。 **注意：** 此参数只能由 **configure replication server** 设置。

示例

- **示例 1** – 此命令是在 TOKYO_RS Replication Server 上输入的，它创建从 TOKYO_RS 到 SYDNEY_RS Replication Server 的直接路由。TOKYO_RS 可以通过此路由，使用登录名“sydney_rsi”和口令“sydney_rsi_ps”登录到 SYDNEY_RS：

```
create route to SYDNEY_RS
 set username sydney_rsi
 set password sydney_rsi_ps
```

- **示例 2** – 此命令是在 TOKYO_RS 上输入的，它创建从 TOKYO_RS 经由中间 Replication Server (MANILA_RS) 到 SYDNEY_RS 的间接路由。从 TOKYO_RS 到 MANILA_RS 以及从 MANILA_RS 到 SYDNEY_RS 的直接路由必须已经存在：

```
create route to SYDNEY_RS
 set next site MANILA_RS
```

- **示例 3** – 此命令创建与第一个示例中类似的直接路由。但是，如果启用了基于网络的安全性，TOKYO_RS 必须使用认证登录到 SYDNEY_RS：

```
create route to SYDNEY_RS
 set unified_login 'required'
```

- **示例 4** – 要在 `NY_DS.pdb1` 主连接的 RS_NY 主 Replication Server 和 RS_LON 复制 Replication Server 之间创建专用路由，请在 RS_NY 上输入：

```
create route to RS_LON
 with primary at NY_DS.pdb1
 go
```

  在为特定连接创建专用路由后，从该连接到目标 Replication Server 的所有事务都将遵循该专用路由。

用法

- 使用 **create route** 创建当前 Replication Server 到远程 Replication Server 的直接或间接路由。
- 创建路由之前，应确定好整体路由方案。有关创建和管理路由的信息，请参见《Replication Server 管理指南第一卷》。
- Replication Server 不支持的路由方案是：路由从同一源 Replication Server 分支，然后在同一中间或目标 Replication Server 会聚。
- Replication Server 通过复制系统将有关新路由的信息分发到合格的节点。通常由于复制系统会有一定时间的滞后，因此更改不会立即显示在所有这些节点上。
- 如果 Replication Server 配置了嵌入 RSSD (ERSSD)，则可以创建一个路由，条件是两个 Replication Server 均为 15.0 或更高版本。如果所创建的路由是源自于当前节点的第一个路由，则会启动日志传送并自动启动 Replication Agent：
  若要更改 Replication Agent 名称，请输入：

```
configure replication server
set erssd_ra to 'value'
```

直接路由

- 在 **create route** 中指定 RSI 用户名和口令并省略 **next site** 子句，可以建立从当前 Replication Server 到目标 Replication Server 的直接路由。
- 创建直接路由之前，在目标 Replication Server 上创建登录名和口令。您可以使用 **rs_init** 来设置这些属性；缺省用户名为“*RS_name*_rsi”，缺省口令为“*RS_name*_rsi_ps”。

如果创建路由时使用的用户名和口令在目标 Replication Server 上不存在，请在此目标 Replication Server 上添加或更改用户和口令。

间接路由

- 在 **create route** 中包括 **next site** 子句，可为 Replication Server 消息建立间接路由。例如，来自纽约且目标为所有欧洲节点的消息可以通过伦敦节点，沿间接路由发送。使用间接路由可减少通过路由的某个部分的消息量。
- 创建间接路由之前，必须首先创建从源 Replication Server 到中间 Replication Server 以及从中间 Replication Server 到目标 Replication Server 的直接路由。
- 一个路由可以有任意数量的中间 Replication Server。但是，每增加一个中间 Replication Server，将延长主节点和复制节点之间的滞后时间，因此应限制中间节点的数量。

专用路由

只有在以下情况下才能创建专用路由：

- 存在从主 Replication Server 到目标 Replication Server 的共享路由，并且该共享路由是直接路由。如果 Replication Server 之间只有间接路由，则无法创建专用路由。
- 共享路由有效且未挂起。
- 共享路由的路由版本为 1570 或更高版本。

请参见《Replication Server 管理指南第二卷》的“性能调优”的“Multi-Path Replication”中的“专用路由”。

路由和 RSSD 表

- 创建直接路由时指定的 RSI 用户名和口令将添加到目标 Replication Server 的 RSSD 中的 *rs_users* 系统表。此用户名和口令还会添加到源 Replication Server 的 RSSD 中的 *rs_maintusers* 系统表。
- 创建路由时，源 Replication Server 向目标 Replication Server 发送源 RSSD 的主用户的登录名和口令。目标 Replication Server 使用此登录来创建对源 Replication Server 上某些 RSSD 系统表的预订。主用户登录名通常命名为“*source_RSSD_name*_prim”，并且存储在目标 Replication Server 上的 *rs_users* 系统表中。

基于网络的安全性参数

- 路由两端必须使用具有相同安全性机制和安全性功能的兼容安全控制层 (SCL) 驱动程序。复制系统管理员负责为每个服务器选择和设置安全性功能。在尝试建立连接之前，Replication Server 不会查询远程服务器的安全性功能。
- **create route** 所指定的基于网络的安全性设置将影响当前 Replication Server 登录到目标 Replication Server 的方式，以及如何实现消息的安全传输。
- 如果将 **unified_login** 设置为“required”，则只有“sa”用户可以无需认证而登录到 Replication Server。如果安全性机制失败，“sa”用户可以使用口令登录到 Replication Server，并禁用 **unified_login**。

- Replication Server 可具有多种安全性机制；所支持的每种机制都在 `libtcl.cfg` 文件中的 SECURITY 下列出。
- 消息加密进程占用的资源较多，会造成严重的性能下降。大多数情况下，明智的做法是只将某些路由的 **msg_confidentiality** 设置为“on”。或者，选择占用资源较少的安全性功能，如 **msg_integrity**。

权限

create route 需要“sa”权限。

另请参见

- alter connection （第 109 页）
- alter route （第 161 页）
- configure replication server （第 181 页）
- create connection （第 214 页）
- drop connection （第 297 页）
- drop route （第 309 页）

create schedule

创建日程表以便在指定的时间执行 shell 命令。

语法

```
create schedule sched_name as ‘sched_time’
[set {on | off}] for exec ‘command’
```

参数

- **sched_name** – 所提供日程表的名称，该名称：
  - 必须符合标识符规则
  - 必须是唯一的
  - 长度可在 1 到 30 个字节之间
- **sched_time** – 执行 ‘*command*’ 的时间和日期。采用限制的 UNIX cron 格式提供日期和时间，用一个空格分隔 time 和 date 参数：

  ```
  [mm] [HH] [DOM] [MON] [DOW]
  ```

  time 和 date 参数为：

参数	说明	值
mm	超过小时的分钟数	0 – 59.使用“*”包括所有合法值。

参数	说明	值
HH	以 24 小时格式表示的小时	0 – 23。使用“*”包括所有合法值。
DOM	月份中的某一天	1 – 31。使用“*”包括所有日期。
MON	一年中的月份	1 – 12。使用“*”包括所有月份。
DOW	星期几	0 – 6（0=星期日）。使用“*”包括所有星期几。

  - 使用星号“*”指定所有有效值。例如，“17 20 * * *”表示晚上 8:17 的每日日程表。
  - 使用逗号“,”分隔非某个范围部分的值。例如，“17 20 1,15 * *”表示每个月第 1 天和第 15 天的晚上 8:17，其中 1 和 15 是 *DOM* 参数的值。
  - 在两个值之间使用连字符“-”指定某个范围的值，含这两个值。例如，“17 20 * * 1-5”表示从星期一到星期五的下午 8:17，其中“1-5”是 *DOW* 参数值的范围。
  - 对于 *DOM*、*MON* 或 *DOW* 参数，您可以同时使用 *DOM* 和 *DOW* 参数来指定日期。Replication Server 遵循您在该字符串中指定的所有日程表。例如，“0 12 16 * 1”表示每个星期一的中午 12:00 和每月 16 日的中午 12:00。
- **set [on | off}** – 创建日程表后启用或禁用该日程表。缺省情况下，启用日程表。
- **command** – shell 命令，例如要在指定日程表中执行的脚本、可执行程序或批处理文件。用单引号括起来。

  Shell 命令：

  - 必须位于 `$SYBASE/$SYBASE_REP/sched`（对于 UNIX）或 `%SYBASE%\%SYBASE_REP%\sched`（对于 Windows）
  - 可以包括在 shell 命令内用空格分隔的参数。
  - 在 Windows 中，**create schedule** 执行 shell 命令或批处理文件内最后一个参数指定的命令。还必须在 **create schedule** 命令行中包括指向某个文件的 **stdout**。

  在 shell 命令名中：

  - 只能使用 ASCII 字母数字字符，例如 A – Z、a – z 和 0 – 9
  - 可以使用“.”、“-”和“_”字符
  - 不能使用“\”和“/”字符
  - 如果要在 Windows 上执行，则必须包括 `.bat` 后缀。例如，在 Windows 上，名称应为 `suspend_conn.bat`，在 UNIX 上，名称应为 `suspend_conn.sh`。

示例

- **示例 1** – 在 Windows 中创建“schedule1”，用于在每个星期一的中午 12:00 和每个月 16 日的中午 12:00 挂起到 SYDNEY_DS 数据服务器中 *pubs2* 数据库的连接：

1. 创建一个文本文件（例如 `sql.txt`），其中包含要调度的实际 Replication Server 命令行。例如，`sql.txt` 可以包含

   ```
   suspend connection to SYDNEY_DS.pubs2
   go
   ```

2. 在 Windows 中创建一个批处理文件（例如 `suspend_conn.bat`），其中包含 **isql** 和用于运行 `sql.txt` 中的命令行的相关参数。例如，`suspend_conn.bat` 可以包含：

   `%SYBASE%\OCS-15_0\bin\isql.exe` **-Usa -P -S** SYDNEY_DS **-I** `%SYBASE%\sql.ini` **-i** `%SYBASE%\REP-15_5\sched\sql.txt`

3. 创建日程表“schedule1”：

   ```
   create schedule schedule1 as ‘0 12 16 * 1’ for exec
   ‘test.bat > c:\temp\test.out’
   go
   ```

- **示例 2** – 创建“schedule2”，用于在 UNIX 中执行 `suspend_conn.sh` 脚本，以便在每天下午 8:17 挂起到 SYDNEY_DS 数据服务器中 *pubs2* 数据库的连接：

  ```
  create schedule schedule2 as ‘17 20 * * *’ for exec
  ‘suspend_conn.sh’
  ```

- **示例 3** – 创建“schedule2”，用于执行 `resume_conn.sh` 脚本，以便在每天上午 7:15 恢复到 SYDNEY_DS 数据服务器中 *pubs2* 数据库的连接：

  ```
  create schedule schedule2 as '15 7 * * *' for exec
  ‘resume_conn.sh’
  ```

用法

- 安排要执行复制任务（例如，在复制数据库冻结并且未从主数据库接收数据时，报告复制数据库的特定状态）的时间。
- 您可以将复制调度为仅在特定晚上时间发生，以便第二天的处理不会更改复制数据库，并且可以报告前一天在复制数据库中冻结的数据。您可以通过创建在一天的特定时间挂起和重新开始与复制数据库的连接的日程表来实现此目的。

权限

create schedule 需要“sa”权限。

另请参见

- admin schedule （第 59 页）
- alter schedule （第 168 页）
- drop schedule （第 311 页）

create subscription

创建和初始化预订，并实现预订数据。预订的对象可以是数据库复制定义、表复制定义、函数复制定义或发布。

语法

```
create subscription sub_name
for {table_repdef | func_repdef | {publication pub |
     database replication definition db_repdef}
     with primary at server_name.db}
with replicate at data_server.database
[where {column_name | @param_name}
     {< | > | >= | <= | = | &} value
[and {column_name | @param_name}
     {< | > | >= | <= | = | &} value]...]
[without holdlock | incrementally | without materialization]
[subscribe to truncate table]
[for new articles]
```

参数

- **sub_name** – 预订的名称，此名称必须符合标识符的命名规则。预订的名称对于复制定义（如果适用）、复制数据服务器和数据库必须是唯一的。
- **for table_rep_def** – 指定所预订的表复制定义。
- **for function_rep_def** – 指定所预订的函数复制定义的名称。
- **for publication pub_name** – 指定预订的发布。
- **for database replication definition db_repdef** – 指定所预订的数据库复制定义。
- **with primary at data_server.database** – 指定主数据的位置。如果主数据库是使用逻辑连接的热备份应用程序的一部分，则 *data_server.database* 是逻辑数据服务器和数据库的名称。如果配置的是多路径复制系统，则也可以在该子句中指定替代主连接名。
- **with replicate at data_server.database** – 指定复制数据的位置。如果复制数据库是热备份应用程序的一部分，则 *data_server.database* 是逻辑数据服务器和数据库的名称。如果配置的是多路径复制系统，则也可以在该子句中指定替代复制连接名。
- **where** – 设置要通过预订复制的列或参数值的标准。如果省略 **where** 子句，将复制所有行或参数。

  在对表复制定义或函数复制定义的预订中可以使用 **where** 子句。但是，在对数据库或发布的预订中不能使用 **where** 子句。

  where 子句由一个或多个简单比较组成，使用以下关系运算符之一对复制定义中的可搜索列或可搜索参数和某个实际值进行比较：**<**、**>**、**<=**、**>=**、**=** 或 **&**。（只有 *rs_address* 列或参数支持 **&** 运算符。）可以使用关键字 **and** 来连接比较。

  表达式中使用的列或参数名称必须包括在表复制定义的 **searchable columns** 列表或函数复制定义的 **searchable parameters** 列表中。

在预订表达式中不能对 Java 列进行求值。因此，不能在 **where** 子句中包括类型为 *rawobject* 或 *rawobject in row* 的 Java 列。

预订中的 **where** 子句的最大大小为 255 个字符。

注意： 无法将少于 7 个字节的二进制值转换为整数。解决方法包括用零将二进制值填充到 8 个字节，或者使用整数值代替二进制值。

- **column_name** – 主表中的列名，用于对表复制定义的预订。
- **@param_name** – 复制存储过程中的参数名，用于对函数复制定义的预订。
- **value** – 指定列或参数的值。有关不同数据类型值的输入格式，请参见“数据类型”。

  表达式中使用的列或参数名称必须包括在复制定义的 **searchable columns** 或 **searchable parameters** 列表中。
- **without holdlock** – 在不锁定的情况下从主数据库中选择数据，用于非原子实现。在复制数据库中将以每个事务插入 1000 行为增量进行应用。有关详细信息，请参见“非原子实现”。
- **incrementally** – 初始化预订，并以每个事务插入 1000 行为增量应用预订数据。在主数据库上使用锁定，用于原子实现。
- **without materialization** – 不实现预订的数据。如果主数据库没有活动，而且复制数据库中已经存在数据，请使用此选项。或者，如果已经挂起主数据库中的活动，并且已经手动将数据转移到复制数据库，请使用此选项。数据库预订必须包括此选项。
- **subscribe to truncate table** – 对于表复制定义、数据库复制定义或发布的预订，允许将 **truncate table** 命令复制到预订复制数据库。

  对于特定的数据库，此选项的设置必须与将数据复制到相同复制表中的任何现有预订的选项设置相同。否则，新预订将被拒绝。
- **for new articles** – 刷新现有的预订。指示 Replication Server 检查对发布的预订，然后创建对未预订项目的预订。

示例

- **示例 1** – 创建名为 *titles_sub* 的预订。它指定将 *titles* 表中带有类型为“business”的列的行复制到名为 SYDNEY_DS 的数据服务器的 *pubs2* 数据库中的 *titles* 表中：

```
create subscription titles_sub
 for titles_rep
 with replicate at SYDNEY_DS.pubs2
 where type = 'business'
```

- **示例 2** – 创建名为 *titles_sub* 的预订，此预订包括 *titles* 表中价格大于或等于 $10.00 的行：

```
create subscription titles_sub
 for titles_rep
 with replicate at SYDNEY_DS.pubs2
 where price >= $10.00
```

- **示例 3** – 为函数复制定义 *myproc_rep* 创建名为 *myproc_sub* 的预订。要使用此命令创建对函数复制定义的预订，数据必须已经存在于复制数据库上，并且您必须使用 **without materialization** 子句：

```
create subscription myproc_sub
 for myproc_rep
 with replicate at SYDNEY_DS.pubs2
 without materialization
```

- **示例 4** – 为发布 *pubs2_pub* 创建名为 *pubs2_sub* 的预订：

```
create subscription pubs2_sub
 for publication pubs2_pub
 with primary at TOKYO_DS.pubs2
 with replicate at SYDNEY_DS.pubs2
```

- **示例 5** – 为数据库复制定义 *pubs2_rep* 创建数据库预订 *pubs2_sub*：

```
create subscription pubs2_sub
 for database replication definition pubs2_rep
   with primary at NEWYORK_DS.pubs2
   with replicate at TOKYO_DS.pubs2
 without materialization
 subscribe to truncate table
```

- **示例 6**

  为 NY_DS.rdb_conn2 替代复制连接上的 **repdef_conn2** 复制定义创建 **sub_conn2** 预订：

```
create subscription sub_conn2 for repdef_conn2
with replicate at NY_DS.rdb_conn2
without materialization
go
```

- **示例 7**

  对到 LON_DS 主数据服务器的 LON_DS.pdb_conn2 替代主连接上的 **repdef_conn2** 复制定义创建 **sub_conn2** 预订，其中 NY_DS.rdb 是缺省复制连接：

```
create subscription sub_conn2 for repdef_conn2
with primary at LON_DS.pdb_conn2
with replicate at NY_DS.rdb
without materialization
go
```

用法

- 要预订函数或数据库复制定义，请使用带有 **without materialization** 子句的 **create subscription** 命令，或使用 **define subscription** 和其它批量实现命令。
- 在存储复制数据的数据库所在的 Replication Server 上执行 **create subscription**。
- 有关预订及其在复制中所起作用的详细信息，请参见《Replication Server 管理指南第一卷》。

- 如果需要更改预订的复制定义，必须删除此预订，然后再重新创建，并指定要预订的复制定义的名称。
- 可以为同一主表或数据库创建多个复制定义。而同一复制表或数据库不能预订多个复制定义，但可以多次预订同一复制定义。
- 如果要实现 *text*、*unitext*、*image* 或 *rawobject* 数据，只有在数据行的大小小于 32K 时，才能使用自动实现。否则，必须使用批量实现。
- 对于多路径复制，由于主数据库和 Replication Server 之间的所有主连接共享所有复制定义，因此必须在预订中指定哪个主连接是数据源，哪个复制连接是复制目标。请参见《Replication Server 管理指南第二卷》的“性能调优”中的“Multi-Path Replication”。

对于数据库复制定义的预订

- 创建数据库预订时，不能使用 **where** 子句限制数据预订。所有数据都将被预订。
- 对于数据库预订，只能使用不实现或批量实现方法。通过 **define subscription** 可使用转储和装载或其它批量实现方法。通过 **create subscription** 可使用不实现方法。
- 不能从同一源预订多个数据库复制定义。
- 如果复制 Replication Server 版本比主 Replication Server 低，则无法在复制 Replication Server 中为主 Replication Server 控制的主数据库创建数据库预订。
- 若要成功为数据库预订所预订的主数据库创建表复制定义，复制 Replication Server 版本必须比表复制定义版本高或与之相同。

预订发布

- 如果发布有效，可以创建对此发布的预订，以便开始复制到复制数据库。支持所有形式的预订实现。
- 创建发布预订时，Replication Server 为此发布包含的每个项目分别创建一个基本预订。每个项目预订使用父发布预订的名称。
  - 使用基本或非原子实现时，将按照项目添加到发布的顺序，每次实现一个项目的预订。
  - 使用带有 **without materialization** 子句的 **create subscription** 时，将同时激活并验证所有项目的预订。
- 对发布的预订不能包括 **where** 子句。但是，您可以通过在发布所包含的每个项目中加入一个或多个 **where** 子句，自定义到复制节点的复制。

指定需要进行 HDS 转换的列

- 创建包含 **where** 子句的预订时，确保 **where** 子句比较中的值是已声明的数据类型格式。
- 在 **where** 子句中指定要进行类级别转换或列级转换的列的预订不能自动取消实现。必须使用批量实现或不实现方法。

复制 Truncate Table

- 创建第一个预订时，可以选择包括或不包括 **subscribe to truncate table** 选项。随后每个复制到同一表中的预订必须以第一个预订为范例。否则，尝试创建此预订时，将被拒绝。
- 通过执行 **sysadmin apply_truncate_table**，可以更改特定复制表当前的“subscribe to truncate table”状态

执行 **create subscription** 的要求

- 除了下面列出的权限外，确保在执行 **create subscription** 之前满足这些要求。
  对于表复制定义的预订：
  - 存在要复制的主表的复制定义，并且已经使用 **sp_setreptable** 将此表标记为要复制。
  - 如果预订使用 **sp_reptostandby** 标记的表，则必须使用 **rep_as_standby** 配置参数配置主数据库连接，使用 **send_warm_standby_exacts** 配置 RepAgent。
  - 复制定义中引用的表必须同时存在于主数据库和复制数据库中。每个表均包含复制定义中定义的列和数据类型。
    此表对于创建预订的用户和维护预订的用户是可见的。实现此目的最简单的方法就是让数据库所有者创建此表。

  对于函数复制定义的预订：
  - 存在要复制的存储过程的复制定义，并且已经使用 **sp_setrepproc** 将此存储过程标记为要复制。
  - 函数复制定义中引用的存储过程必须同时存在于主数据库和复制数据库中。每个存储过程均包含函数复制定义中定义的参数和数据类型。

  对于发布的预订：
- 存在一个发布，其中包含要复制的主表或存储过程的项目。这些项目指定的复制定义满足上述要求。
- 此发布有效。

热备份应用程序的要求

- 在热备份应用程序中创建预订时，有以下要求：
  - 如果目标数据库是热备份应用程序的一部分，则此表同时存在于活动数据库和备用数据库中。两个表必须都使用 **sp_setreptable** 或 **sp_reptostandby** 标记为要复制。
  - 对于逻辑主数据库，在 Replication Server 添加备用数据库的过程中，不能创建预订。

同名表的要求

- 如果主 Adaptive Server 数据库包含一个复制表和另一个与其同名的表，则第二个（非复制）表的所有者必须使用自定义的 **rs_select** 或 **rs_select_with_lock** 函数字符串，才可以创建对此复制表的预订。例如：
  - 如果存在名为 *db.dbo.table1* 的主表的复制定义，并且
  - 数据库用户“jane”拥有名为 *db.jane.table1* 的表，则

- Jane 不能使用缺省的函数字符串创建对 *db.dbo.table1* 的复制定义的预订。

原子实现

- 使用此命令实现预订的缺省方法是原子实现。原子实现锁定主表，并在一次基本操作中通过网络复制预订数据。
- 在原子实现期间，直到主数据库中的 select 事务完成后，复制数据库中才出现行。如果预订指定大量的行，则 select 事务运行的时间可能会较长，并导致复制节点上出现延迟。

使用原子实现的要求

- 如果要使用预订实现的基本方法，需要满足以下条件：
  - 您或数据库所有者必须拥有主表，或者您必须在主数据库上使用用户定义的函数字符串进行 **select** 操作。
  - 数据库所有者或维护用户必须拥有复制表，或者您必须在复制数据库上使用用户定义的函数字符串进行 **select** 操作。如果复制表的所有者不同于主表的所有者，则必须使用不同的函数字符串类来创建唯一的函数字符串。

使用 without holdlock 或 incrementally 选项

- **without holdlock** 或 **incrementally** 选项是预订实现的缺省基本方法的替代方法。当指定这些选项时，Replication Server 将分批应用行，而在复制数据库中，数据也将分批出现。

  其结果是，在实现过程中，对复制数据库的查询可能会返回不完整的预订数据。这种临时情况将在 **check subscription** 指示此预订有效时结束。

incrementally 选项

- **incrementally** 选项是原子实现的另一种表现方式。对于数量较多的预订，使用此选项可以避免复制数据库出现运行时间较长的事务。预订数据不是在复制数据库上基本应用的，因此，这些数据可用；但是，在实现结束并且预订生效之前，这些数据是不完整的。
- 使用 **incrementally** 时，将在使用 **with a holdlock** 的情况下执行 **select** 操作，以保持与主数据库的连续一致性。复制表将经过之前在主数据上出现的状态。

  在所有情况下，当实现结束，并且 **check subscription** 指示预订有效时，复制数据将与主数据库一致。

非原子实现

- **without holdlock** 选项使用非原子实现。如果指定此选项，将在不锁定的情况下，从主数据库中选择实现行。如果主数据库中的行是在选择后更新的，则会出现不一致现象。要消除不一致现象，可能在使用 **without holdlock** 时使用 **set autocorrection on**。
- 如果数据已经存在于复制数据库上，则可以使用基本或非基本实现，而不使用批量实现。

使用非原子实现的要求

- 如果要使用预订实现的非基本方法，需要满足以下条件：
  - 如果通过从主数据库分发应用函数来更新数据，或者使用交换函数来更新数据，请不要使用 **without holdlock**。例如，如果存储过程通过增加某一列以前的值来更新行，在实现结束时此值可能不正确。
  - 如果为复制定义设置了 **replicate minimal columns**，则不能使用 **without holdlock** 创建新预订。
  - 对于非基本预订，如果在执行 **switch active** 时实现非基本预订，则此预订将被标记为“SUSPECT”。

注意： 如果您对原子或非原子实现方法使用 **create subscription** 并且复制定义中有带引号的标识符，则必须更改主连接以便允许使用带引号的标识符。

不实现

without materialization 子句指定不实现方法。在预订数据已经存在于复制数据库的情况下，此子句提供了一种创建预订的简便方法。

不实现的要求

- 预订数据必须已经存在于复制数据库中。
- 主数据库和复制数据库必须处于同步状态。
- 主数据库上的活动必须停止，使得 Replication Server 稳定队列中没有其他更新操作。

使用 **rs_address** 数据类型

- 可以预订列或参数使用特殊数据类型 *rs_address* 的复制定义。此数据类型提供了一种唯一预订解析方法，对 *rs_address* 数据类型（基于基本 *int* 数据类型）的位图与预订的 **where** 子句中的位屏蔽进行比较。主 Replication Server 通过位图比较的结果来确定某个复制节点是否应接收各行中的数据。
- 在 **where** 子句中，只有 *rs_address* 列或参数支持位图比较运算符 **&**，如下所示：

```
where rs_address_column1  & bitmask
[and rs_address_column2  & bitmask]
 [and other_search_conditions]
```

- 如果唯一更改的列是 *rs_address* 列，Replication Server 将不复制行，除非更改的位表示应该在复制数据库中插入或删除该行。

  由于这种过滤机制，复制数据库中的 *rs_address* 列可能与主数据库中相应的列不完全相同。这样可以优化使用 *rs_address* 列来指定目标复制数据库的应用程序的性能。

rs_address 数据类型的工作方式

- *rs_address* 列字段中的每一位都可以表示一类数据，例如库存或帐单。在预订位屏蔽中，对于要复制到预订节点的每类数据，将与之对应的位设置为“on”(1)。例如，仓库节点上对库存数据感兴趣的用户会将预订位图中的库存位设置为“on”。如果这些仓库用户对帐单数据不感兴趣，会将此位设置位“off”(0)。如

果预订位屏蔽和 *rs_address* 列中都将某一位设置为“on”，则包含此位的行将被复制。

`rs_address` 的基本 **int** 数据类型仅限于 32 位

- 由于基本 *int* 数据类型仅限于 32 位，您可能需要用多个 *rs_address* 列来构造主表。使用 **and** 关键字可以创建能够对多个 *rs_address* 列执行位图比较的单个预订。但是，如果要在两个或多个 *rs_address* 列中设置一个或多个位，则必须创建单独预订才能预订行。

使用 32 位十六进制数字表示 **rs_address**

- 也可以使用 **and** 关键字和命令语法中说明的比较运算符（除 & 以外），为非 *rs_address* 列指定搜索条件。如果使用 **and** 指定搜索条件，预订数据可能不会被复制或者可能迁移出预订，即使 *rs_address* 位图比较会另外复制一行。
- 可以将 *rs_address* 列与 **where** 子句中的 32 位整数值或 32 位十六进制数字进行比较。如果您使用 16 进制数，请根据需要为每个数补 0，使它成为 8 位的 16 进制值。

警告！ 将 *rs_address* 列与预订的 **where** 子句中的十六进制数字相比较时，应当十分小心。Adaptive Server 和 Replication Server 将十六进制值看作二进制字符串。二进制字符串将通过复制字节转换为整数。在不同的平台上，这可能产生代表不同整数值的位模式。

例如，0x0000100 在认为字节 0 的有效性最高的平台上，代表 65,536；而在认为字节 0 的有效性最低的平台上，则代表 256。由于这些字节排序的差别，包括十六进制数字的位图预订在多平台复制系统中可能不起作用。

- 有关 *rs_address* 和 *int* 数据类型的详细信息，请参见“数据类型”。另请参见《Replication Server 管理指南第一卷》。
- 有关数据类型之间转换的详细信息，请参见《Adaptive Server Enterprise 参考手册》和《Open Client 和 Open Server 通用库参考手册》。

监控预订

- Replication Server 实现预订时，使用预订创建者的登录名登录到主数据服务器，并从主表中选择行。使用 **check subscription** 可以监控实现的进展情况。
- **create subscription** 在数据实现完成之前返回提示。当 **check subscription** 在复制 Replication Server 上报告“VALID”时，表示实现完成。

权限

要执行 **create subscription**，必须具有以下登录名和权限：

- 在复制 Replication Server、主 Replication Server 和主 Adaptive Server 数据库上具有相同的登录名和口令。
- 在输入此命令的复制 Replication Server 上具有“创建对象”或“sa”权限。

- 在主 Replication Server 上具有“create object”、“primary subscribe”或“sa”权限。
- 在主 Adaptive Server 数据库中的主表上具有 **select** 权限。
- 在主 Adaptive Server 数据库中的 **rs_marker** 存储过程上，具有 **execute** 权限。
- 复制数据库的维护用户在复制表上必须具有 **select**、**insert**、**update** 和 **delete** 权限，而且对复制中使用的函数必须具有 **execute** 权限。

另请参见

- alter applied function replication definition （第 107 页）
- alter database replication definition （第 135 页）
- alter request function replication definition （第 159 页）
- check subscription （第 177 页）
- create alternate connection （第 202 页）
- create article （第 211 页）
- create database replication definition （第 225 页）
- create applied function replication definition （第 206 页）
- create function string （第 236 页）
- create publication （第 254 页）
- create replication definition （第 258 页）
- create request function replication definition （第 269 页）
- define subscription （第 290 页）
- drop subscription （第 312 页）
- set （第 328 页）
- sysadmin apply_truncate_table （第 339 页）
- 精确数值（整数）数据类型 （第 18 页）

create user

将新的用户登录名添加到 Replication Server。

语法

```
create user user
set password {new_password | null}
[set password_parameter to 'parameter_value']
```

参数

- **user** – 登录名。
- **new_password** – 新口令。

- **old_password** – 如果使用 *verify password* 参数，则为当前用户口令。
- **password parameter** – 请参见表 34. 口令参数。
- **parameter_value** – 口令参数的说明和值。

表 34. 口令参数

password_parameter	说明和*值*
password_expiration	口令到期之前经过的天数。 • 0 – 口令永不过期（缺省值）。 • 范围 – 0 到 32,767。 您可以将 **password_expiration** 与 **create user** 一起使用。 如果口令已过期，Replication Server 将锁定用户帐户并通知用户口令已过期。如果用户不重置其口令，则断开连接后，用户将无法登录，直到管理员重置口令。新口令必须满足所有口令要求。 对于 **rs_init** 创建的具有 **connect source** 权限的任何用户或 ID 用户，口令不会过期。这些口令会覆盖您在 Replication Server 中为所有用户设置的任何 **password_expiration** 设置。数据库、其它 Replication Server 和 Replication Agent 使用具有 **connect source** 权限的用户 ID。 管理员应考虑对为复制代理或 RSI 创建的任何用户进行适当设置，以使口令不会过期。

示例

- **示例 1** – 创建新用户登录名“louise”，口令为“EnnuI”：

```
create user louise
set password EnnuI
```

- **示例 2** – 创建名为 jsmith，初始口令为 1Buiopr89 的用户，口令有效期为 90 天：

```
create user jsmith
set password to 1Buiopr89
set password_expiration to '90'
```

用法

- **create user** 为用户创建新的登录名。
- 用户可以使用 **alter user** 命令更改自己的口令。
- 用户的登录名和口令均区分大小写。
- **password_expiration** 是管理员可用于 **alter user** 和 **create user** 命令的唯一参数。
- 使用 **create user** 命令为单个用户指定的口令设置覆盖使用 **configure replication server** 命令设置的任何值。
  请参见 **configure replication server** 中的表 24. 口令参数表。

- 口令有效期：
  - 当管理员或用户更改用户的口令时，Replication Server 会记录口令的设置日期。当用户登录时，Replication Server 会根据口令有效期设置检查登录日期。如果已针对该用户或在系统级别设置了口令有效期，并且 Replication Server 确定该口令已到期，Replication Server 会通知用户更改口令，并会锁定该用户帐户。只有当用户输入满足所有口令要求的新口令时，Replication Server 才会解锁该帐户。如果用户在更改口令之前断开连接，管理员必须重置口令。
  - Sybase 建议对具有“connect source”权限的用户（例如 Replication Agent 用户）将 **password_expiration** 设置为 0，以避免其口令到期并干扰复制数据。

权限

create user 需要“sa”权限。

另请参见

- alter user （第 170 页）
- drop user （第 315 页）
- grant （第 316 页）
- revoke （第 327 页）

define subscription

将预订添加到 Replication Server 系统表，但不实现或激活预订。预订的对象可以是数据库复制定义、表复制定义、函数复制定义或发布。此命令是批量预订实现进程或刷新发布预订进程的开始。

语法

```
define subscription sub_name
for {table_rep_def | function_rep_def |
     publication pub_name | database replication definition
db_repdef
     with primary at data_server.database} |
with replicate at data_server.database
     [where {column_name | @param_name}
     {< | > | >= | <= | = | &} value
     [and {column_name | @param_name}
     {< | > | >= | <= | = | &} value]...]
[subscribe to truncate table]
[for new articles]
[use dump marker]
```

参数

- **sub_name** – 预订的名称，此名称必须符合标识符的命名规则。预订的名称对于复制定义（如果适用）、复制数据服务器和数据库必须是唯一的。
- **for table_rep_def** – 指定所预订的表复制定义。
- **for function_rep_def** – 指定所预订的函数复制定义的名称。
- **for publication pub_name** – 指定预订的发布。
- **for database replication definition db_repdef** – 指定所预订的数据库复制定义。
- **with primary at data_server.database** – 指定主数据的位置。如果主数据库是热备份应用程序的一部分，则 *data_server.database* 是逻辑数据服务器和数据库的名称。只有对发布的预订才使用此子句。
- **with replicate at data_server.database** – 指定复制数据的位置。如果复制数据库是热备份应用程序的一部分，则 *data_server.database* 是逻辑数据服务器和数据库的名称。
- **where** – 设置要通过预订复制的列或参数值的标准。如果省略 **where** 子句，将复制所有行或参数。

  在对表复制定义或函数复制定义的预订中可以使用 **where** 子句。但是，在对发布的预订中不能使用 **where** 子句。

  where 子句由一个或多个简单比较组成，使用以下关系运算符之一对复制定义中的可搜索列或可搜索参数和某个实际值进行比较：**<**、**>**、**<=**、**>=**、**=** 或 **&**。（只有 *rs_address* 列或参数支持 **&** 运算符。）可以使用关键字 **and** 来连接比较。

  表达式中所使用的列或参数名称必须包括在表复制定义的 **searchable columns** 列表中或函数复制定义的 **searchable parameters** 列表中。

  在预订表达式中不能对 Java 列进行求值。因此，不能在 **where** 子句中包括类型为 *rawobject* 或 *rawobject in row* 的 Java 列。
- **column_name** – 主表中的列名，用于对表复制定义的预订。
- **@param_name** – 复制存储过程中的参数名，用于对函数复制定义的预订。
- **value** – 指定列或参数的值。
- **subscribe to truncate table** – 对于表复制定义或发布的预订，允许将 **truncate table** 命令复制到预订复制数据库。

  此选项的设置必须与将数据复制到相同复制表中的任何现有预订的选项设置相同。否则，新预订将被拒绝。
- **for new articles** – 刷新现有的预订。指示 Replication Server 检查对发布的预订，然后创建对未预订项目的预订。
- **use dump marker** – 指示 Replication Server 将事务应用到复制数据库。**use dump marker** 会自动激活并验证数据库预订。如果没有此选项，用户必须手动激活并验证数据库预订。

  注意： 应每次使用一个 **dump marker**，因为您无法使用 **dump marker** 定义多个数据库预订。还需要将 **dump database** 命令放在每个预订命令之间。如果在 MSA 复

制中使用跨平台 dump 和 load (XPDL) 功能，则不要对实现使用 **use dump marker** 子句。

示例

- **示例 1** – 创建名为 *titles_sub* 的预订。它指定将 *titles* 表中带有类型为“business”的列的行复制到名为 SYDNEY_DS 的数据服务器的 *pubs2* 数据库中的 *titles* 表中：

```
define subscription titles_sub
  for titles_rep
    with replicate at SYDNEY_DS.pubs2
    where type = 'business'
```

- **示例 2** – 创建名为 *titles_sub* 的预订，此预订包括 *titles* 表中价格大于或等于 $10.00 的行：

```
define subscription titles_sub
  for titles_rep
    with replicate at SYDNEY_DS.pubs2
    where price >= $10.00
```

- **示例 3** – 为函数复制定义 *myproc_rep* 创建名为 *myproc_sub* 的预订：

```
define subscription myproc_sub
  for myproc_rep
    with replicate at SYDNEY_DS.pubs2
```

- **示例 4** – 为发布 *pubs2_pub* 创建名为 *pubs2_sub* 的预订：

```
define subscription pubs2_sub
  for publication pubs2_pub
    with primary at TOKYO_DS.pubs2
    with replicate at SYDNEY_DS.pubs2
```

- **示例 5** – 为数据库复制定义 *pubs2_rep* 创建预订 *pubs2_sub*：

```
define subscription pubs2_sub
  for database replication definition pubs2_rep
    with primary at NEWYORK_DS.pubs2
    with replicate at TOKYO_DS.pubs2
    subscribe to truncate table
    use dump marker
```

有关为整个复制系统创建预订的示例，请参见《Replication Server 设计指南》。

用法

- 使用 **define subscription** 可以通过批量实现手动创建预订。通过批量实现，预订创建和实现可以在不同的步骤中执行，这样您可以从介质中装载初始数据，而不是通过 WAN 从主数据库中发送这些数据。
- 如果向当前具有预订的发布中添加了一些新项目，要创建对这些项目的新预订，必须刷新此发布预订。
- 使用 **activate subscription** 激活预订，并使用 **validate subscription** 验证预订。

- 虽然可以为同一主表创建多个复制定义，但对于同一复制表，只能预订一个复制定义。但是，可以多次预订同一复制定义。

预订发布

- 在开始将数据复制到复制数据库时，可以创建对有效发布的预订。支持所有形式的预订实现。
- 使用 **define subscription** 在发布预订中创建新的项目预订。然后，使用 **activate subscription** 手动为新项目预订装载预订数据，并使用 **validate subscription** 验证发布预订。
- 创建发布预订时，Replication Server 为此发布包含的每个项目分别创建一个基本预订。每个项目预订使用父发布预订的名称。
- 激活和验证发布预订的同时，还激活并验证其全部项目预订。
- 对发布的预订不能包括 **where** 子句。但是，您可以通过在发布所包含的每个项目中加入一个或多个 **where** 子句，自定义到复制节点的复制。

对于数据库复制定义的预订

- 创建数据库预订时，不能使用 **where** 子句限制数据预订。所有数据都将被预订。
- 对于数据库预订，只能使用不实现或批量实现方法。通过 **define subscription** 可使用转储和装载或其它批量实现方法。通过 **create subscription** 可使用不实现方法。
- 不能从同一源预订多个数据库复制定义。

复制 Truncate Table

- 创建对表的第一个预订时，可以选择包括或不包括 **subscribe to truncate table** 选项。随后每个将信息复制到同一表的预订必须以第一个预订为范例。否则，尝试创建此预订时，将被拒绝。
- 通过执行 **sysadmin apply_truncate_status**，可以查看或更改特定复制表当前的“subscribe to truncate table”状态。

使用 **rs_address** 数据类型

有关使用 *rs_address* 数据类型的列或参数的说明信息，请参见 **create subscription**。

执行 **define subscription** 的要求

除了下面列出的权限外，确保在执行此命令之前满足这些要求。

- 对于表复制定义的预订：
  - 存在要复制的主表的复制定义，并且已经使用 **sp_setreptable** 将此表标记为要复制。
  - 复制定义中引用的表必须同时存在于主数据库和复制数据库中。每个表均包含复制定义中定义的列和数据类型。

    此表对于创建预订的用户和维护预订的用户也是可见的。实现此目的最简单的方法就是让数据库所有者创建此表。

  对于函数复制定义的预订：

- 存在要复制的存储过程的复制定义，并且已经使用 **sp_setrepproc** 将此存储过程标记为要复制。
- 函数复制定义中引用的存储过程必须同时存在于主数据库和复制数据库中。每个表均包含函数复制定义中定义的参数和数据类型。

对于发布的预订：

- 存在一个发布，其中包含要复制的主表或存储过程的项目。这些项目指定的复制定义满足上述要求。
- 此发布有效。

使用 define subscription 创建预订

- 可以使用 **define subscription** 来预订表复制定义、函数复制定义或发布。
  - 对于表复制定义的预订，在管理要存储复制数据的数据库的 Replication Server 上输入 **define subscription**。
  - 对于函数复制定义的预订，在管理目标存储过程将要在其中通过应用函数传递执行的数据库的 Replication Server 上输入 **define subscription**。
  - 对于发布的预订，在管理要存储复制数据或执行目标存储过程的数据库的 Replication Server 上输入 **define subscription**。
- 在数据库中，表预订维护表（表中的选定行）的复制副本。对主版本进行的更改也应用于此副本。
- 函数预订复制与函数复制定义关联的用户定义的函数调用。复制函数通常包括参数并修改数据，但无需包括复制数据。
- 发布预订包含此发布包含的项目的基本预订，它根据此项目中的复制定义来复制表或用户定义的函数调用。
- 有关预订及其在复制中所起作用的详细信息，请参见《Replication Server 管理指南第一卷》。

create subscriptions 的替代命令

- 使用 **create subscription** 可以在一步中创建、实现、激活并验证对表复制定义、函数复制定义或发布的预订。

权限

要执行 **define subscription**，必须具有以下登录名和权限：

- 在复制 Replication Server、主 Replication Server 和主数据库上具有相同的登录名和口令。
- 在输入此命令的复制 Replication Server 上具有“创建对象”或“sa”权限。
- 在主 Replication Server 上具有“create object”、“primary subscribe”或“sa”权限。

另请参见

- alter applied function replication definition （第 107 页）
- alter request function replication definition （第 159 页）

- activate subscription （第 44 页）
- check subscription （第 177 页）
- create article （第 211 页）
- create function replication definition （第 232 页）
- create publication （第 254 页）
- create applied function replication definition （第 206 页）
- create request function replication definition （第 269 页）
- create subscription （第 280 页）
- drop subscription （第 312 页）
- sysadmin apply_truncate_table （第 339 页）
- validate subscription （第 390 页）

disconnect

终止到服务器的连接。

语法

```
{disconnect | disc} [all]
```

示例

- **示例 1** – 创建从 ost_replinuxvm_02 到 ost_replinuxvm_03 的连接，然后 ost_replinuxvm_02 断开与 ost_replinuxvm_03 的连接：

```
isql -Usa -P -S ost_replinuxvm_02
1> connect to ost_replinuxvm_03
2> go
```

```
Gateway connection to 'ost_replinuxvm_03' is created.
```

```
1> disc
2> go
```

```
Gateway connection to 'ost_replinuxvm_03' is dropped.
```

用法

- **disconnect** 每次将一个连接退出连接堆栈。要退出所有连接，请使用 **disconnect all**。
- 在 Replication Server 15.1 和更低版本中，**disconnect** 命令具有不同的工作方式。在这些版本中，**disconnect** 命令将终止网关模式，并将工作服务器身份恢复为发出第一个 **connect** 命令的 Replication Server。如果连接堆栈中包含 Replication Server 15.2 以及 15.1 或更早版本并发出了 **disconnect** 命令，**show connection** 和 **show server** 命令可能不会显示预期的输出内容。

权限

任何用户都可以执行此命令。

另请参见

- connect （第 200 页）
- show connection （第 332 页）
- show server （第 333 页）

drop article

删除项目，也可以选择同时删除其复制定义。

语法

```
drop article article_name
for pub_name
with primary at data_server.database
[drop_repdef]
```

参数

- **article_name** – 要删除的项目的名称。
- **for pub_name** – 指定项目的发布的名称。
- **with primary at data_server.database** – 指定主数据的位置。如果主数据库是热备份应用程序的一部分，则 *data_server.database* 是逻辑数据服务器和数据库的名称。
- **drop_repdef** – 可选的关键字，用于删除此项目的复制定义（如果此复制定义未在别处使用）。

示例

- **示例 1** – 删除发布 *pubs2_pub* 中名为 *titles_art* 的项目，此发布位于 TOKYO_DS.*pubs2* 数据库中：

  ```
  drop article titles_art
   for pubs2_pub
   with primary at TOKYO_DS.pubs2
  ```

- **示例 2** – 删除发布 *pubs2_pub* 中名为 *titles_art* 的项目，此发布位于 TOKYO_DS.*pubs2* 数据库中。此命令同时删除此项目的复制定义（如果未在别处使用）：

  ```
  drop article titles_art
   for pubs2_pub
   with primary at TOKYO_DS.pubs2
   drop_repdef
  ```

用法

- 使用 **drop article** 可删除发布中的项目。在管理存储主数据的数据库的 Replication Server 上执行 **drop article**。
- 如果某个项目没有预订，则可以删除此项目。如果需要，可首先删除预订。
- 也可以删除此项目的复制定义（如果此复制定义不是任何其它项目的一部分并且没有预订）。
- 只有在复制节点上执行 **create/define subscription** 时，被删除的项目才会从此复制节点中删除。

从具有预订的发布中删除项目

- 如果从某个现有发布中删除项目，则此发布将失效。必须首先使用 **drop subscription for article** 删除项目现有的所有预订，才可以删除此项目。若要创建新的发布预订，必须执行以下操作：
  - 在完成对发布的更改后，验证此发布，然后

  有关刷新发布预订的两种方法的详细信息，请参见 **create subscription** 和 **define subscription**。

权限

drop article 需要“create object”权限。

另请参见

- check subscription （第 177 页）
- create article （第 211 页）
- create publication （第 254 页）
- create subscription （第 280 页）
- define subscription （第 290 页）
- drop function replication definition （第 301 页）
- drop publication （第 307 页）
- drop replication definition （第 308 页）
- drop subscription （第 312 页）

drop connection

从复制系统中删除数据库。

语法

```
drop connection to data_server.database
```

参数

- **data_server** – 要从复制系统删除数据库的数据服务器的名称。
- **database** – 要删除其连接的数据库的名称。

示例

- **示例 1** – 删除到 SYDNEY_DS 数据服务器中 *pubs2* 数据库的连接：

```
drop connection to SYDNEY_DS.pubs2
```

用法

- 使用 **drop connection** 删除默认连接和替代连接的 Replication Server 系统表中的数据库连接信息。此命令并不从删除系统中任何数据库的复制数据。
- 删除连接之前：
  - 删除所有向此数据库复制数据的预订。
  - 如果是到主数据库的连接，则删除此数据库中表的所有复制定义。
- 重新创建到数据库的同名连接之前，可能需要使用 **sysadmin dropdb**。
- Replication Server 通过复制系统将有关被删除的数据库连接的信息分发到合格的节点。通常由于复制系统会有一定时间的滞后，因此更改不会立即显示在所有这些节点上。

权限

drop connection 需要“sa”权限。

另请参见

- admin show_connections （第 66 页）
- alter connection （第 109 页）
- create alternate connection （第 202 页）
- create connection （第 214 页）
- resume connection （第 321 页）
- suspend connection （第 334 页）
- sysadmin dropdb （第 346 页）

drop database replication definition

删除现有的数据库复制定义。

语法

```
drop database replication definition db_repdef
   with primary at server_name.db
```

参数

- **db_repdef** – 数据库复制定义的名称。
- **server_name.db** – 主服务器/数据库组合的名称。例如：*TOKYO.dbase*。

示例

- **示例 1** – 删除数据库复制定义 *dbrep1*:

```
drop database replication definition dbrep1
  with primary at PDS.my_db
```

用法

只有在指定的数据库复制定义没有数据库预订时，**drop database replication definition** 才能成功执行。

另请参见

- alter database replication definition （第 135 页）
- create database replication definition （第 225 页）

drop error class

删除错误类以及与其关联的所有操作。

语法

```
drop [replication server] error class error_class
```

参数

- **replication server** – 指示错误类是 Replication Server 错误类，而不是数据服务器错误类。
- **error_class** – 要删除的错误类的名称。

示例

- **示例 1** – 从 Replication Server 中删除 *pubs2_db_err_class* 错误类。同时删除为 *pubs2_db_err_class* 错误类指派的所有错误操作：

```
drop error class pubs2_db_err_class
```

- **示例 2** – 从 Replication Server 中删除 Replication Server 错误类 *sydney_rs_err_class*。同时删除为 *sydney_rs_err_class* 错误类指派的所有错误操作：

```
drop replication server error class sydney_rs_err_class
```

用法

- 使用 **drop error class** 命令删除错误类。删除某个错误类时，将同时删除为其指派的所有操作。
- 应在创建错误类的 Replication Server 上执行 **drop error class**。
- 不能删除以下错误类：
  - *rs_sqlserver_error_class* 错误类
  - *rs_repserver_error_class* 错误类。
  - 数据库正在使用的错误类
- 要更改某个错误类的主节点，可使用 **move primary of error class** 命令。
- Replication Server 通过复制系统将有关被删除的类的信息分发到合格节点。通常由于复制系统会有一定时间的延迟，因此更改不会立即显示在所有这些节点上。

权限

drop error class 需要“sa”权限。

另请参见

- assign action （第 172 页）
- alter error class （第 138 页）
- create connection （第 214 页）
- create error class （第 228 页）
- drop connection （第 297 页）
- move primary （第 318 页）

drop function

删除用户定义的函数及其函数字符串。

语法

```
drop function [replication_definition.]function
```

参数

- **replication_definition** – 为其创建了此函数的复制定义的名称。
- **function** – 要删除的函数的名称。

示例

- **示例 1** – 删除用于复制定义 *publishers_rep* 的用户定义函数 *upd_publishers*。同时删除为此函数定义的所有函数字符串：

```
drop function publishers_rep.upd_publishers
```

用法

- 使用 **drop function** 可以删除函数名称和为此函数创建的所有函数字符串。
- 应在创建复制定义的 Replication Server 上执行 **drop function**。
- 不能删除系统函数。有关系统函数的详细信息，请参见“Replication Server 系统函数”。
- Replication Server 通过复制系统将有关被删除的用户定义函数的信息分发到合格的节点。通常由于复制系统会有一定时间的滞后，因此更改不会立即显示在所有这些节点上。
- 为一个复制定义删除用户定义的函数同时，也为主表中的所有复制定义删除了此函数。
- 不要对复制函数执行 **drop function**。请改用 **drop function rep def**。

权限

drop function 需要“create object”权限。

另请参见

- create function （第 231 页）
- drop function string （第 302 页）
- move primary （第 318 页）

drop function replication definition

删除函数复制定义及其用户定义的函数。

语法

```
drop function replication definition function_rep_def
```

参数

- **function_rep_def** – 要删除的函数复制定义的名称。

示例

- **示例 1** – 删除名为 *titles_frep* 的函数复制定义及其用户定义函数和函数字符串：

```
drop function replication definition titles_frep
```

用法

- 使用 **drop function replication definition** 可以删除函数复制定义。
- 删除函数复制定义之前，必须删除其所有预订。
- 应在函数复制定义的主 Replication Server 上执行 **drop function replication definition**。
- 删除此函数复制定义所定义的存储过程后，在数据库中执行 **sp_setrepproc**，将此过程的复制状态设置为 **'false'**。这可阻止 RepAgent 将日志条目传送到 Replication Server。
- Replication Server 通过复制系统将有关被删除的函数复制定义的信息分发到合格的节点。通常由于复制系统会有一定时间的滞后，因此更改不会立即显示在所有这些节点上。

权限

drop function replication definition 需要“create object”权限。

另请参见

- alter applied function replication definition （第 107 页）
- alter request function replication definition （第 159 页）
- check subscription （第 177 页）
- create applied function replication definition （第 206 页）
- create request function replication definition （第 269 页）
- create subscription （第 280 页）
- define subscription （第 290 页）
- drop subscription （第 312 页）

drop function string

删除函数字符串类的函数字符串。

语法

```
drop function string
 {replication_definition |
 [owner.] table |
 stored_procedure} .function[;function_string]
 for { [function class] function_class |
[database] data_server.database}
```

参数

- **replication_definition** – 函数操作的表或函数复制定义的名称。

- **[owner.]table** – 指定函数字符串的目标表和表所有者。
- **stored_procedure** – 指定函数字符串的目标存储过程
- **function** – 为其创建函数字符串的函数的名称。
- **function_string** – 要删除的函数字符串的名称。缺省的函数字符串名称与此函数的名称相同。
- **function_class** – 将从其中删除函数字符串的函数字符串类的名称。
- **data_server.database** – 指定要在其中删除目标作用域函数字符串的备用数据库或复制数据库。

示例

- **示例 1** – 为复制定义 *publishers_rep* 从派生类 *sqlserver_derived_class* 中删除 **rs_insert** 函数的函数字符串。现在将从父类继承 **rs_insert** 函数字符串：

```
drop function string
 publishers_rep.rs_insert
 for sqlserver_derived_class
```

- **示例 2** – 在函数字符串类 *sqlserver2_function_class* 中删除复制定义 *publishers_rep* 的用户定义函数 **upd_publishers** 的函数字符串：

```
drop function string
 publishers_rep.upd_publishers
 for sqlserver2_function_class
```

- **示例 3** – 在目标数据库 `NY_DS.rdb1` 中删除 `dbo.authors` 表的函数字符串：

```
drop function string dbo.authors.rs_insert
for database NY_DS.rdb1
```

用法

- 若要使用新的函数字符串替换现有的函数字符串，请使用 **alter function string** 或者使用带有 **overwrite** 的 **create function**。

  警告！ 如果在删除某个函数字符串之后但尚未重新创建此函数字符串之前发生事务，则 Replication Server 将检测到此函数字符串丢失，导致此事务失败。
- 如果删除某个函数，则会从所有函数字符串类中删除对应的函数字符串。
- 如果从派生函数字符串类中删除自定义的函数字符串，则会导致此类从其父类继承函数字符串。
- 从 *rs_sqlserver_function_class* 删除自定义函数字符串会导致 Replication Server 删除自定义函数字符串和缺省函数字符串。要将 *rs_sqlserver_function_class* 中某个函数的自定义函数字符串转换为缺省函数字符串，可使用 **alter function string**，并省略 **output** 子句。
- Replication Server 通过复制系统将有关被删除的函数字符串的信息分发到合格的节点。通常由于复制系统会有一定时间的滞后，因此更改不会立即显示在所有这些节点上。

- 在控制目标数据库（备用数据库或复制数据库）的 Replication Server 中为目标作用域函数字符串执行 **drop function string**。

权限

drop function string 需要“create object”权限。

另请参见

- alter function string （第 143 页）
- create function （第 231 页）
- create function string （第 236 页）
- create function string class （第 249 页）
- drop function （第 300 页）

drop function string class

删除函数字符串类。

语法

```
drop function string class function_class
```

参数

- **function_class** – 要删除的函数字符串类的名称。

示例

- **示例 1** – 删除派生的函数字符串类 *sqlserver_derived_class* 及其所有自定义的函数字符串：

```
drop function string class
 sqlserver_derived_class
```

- **示例 2** – 删除函数字符串类 *sqlserver2_function_class* 及其函数字符串：

```
drop function string class
 sqlserver2_function_class
```

用法

- 使用 **drop function string class** 删除函数字符串类。函数字符串类用于对数据库的所有函数字符串进行分组。
- 如果删除函数字符串类，则还会删除所有关联的函数字符串，并删除对此类的所有引用。

- 不能删除数据库连接上仍在使用的函数字符串类。
- 不能删除系统提供的三个类（*rs_sqlserver_function_class*、*rs_default_function_class* 或 *rs_db2_function_class*）中的任何一个。
- 不能删除任何作为派生类的父类的函数字符串类。

权限

drop function string class 需要“sa”权限。

另请参见

- create function string class （第 249 页）
- drop function （第 300 页）
- drop function string （第 302 页）

drop logical connection

删除逻辑连接。逻辑连接用于管理热备份应用程序。

语法

```
drop logical connection to data_server.database
```

参数

- **data_server** – **create logical connection** 命令中指定的逻辑数据服务器。
- **database** – **create logical connection** 命令中指定的数据库名称。

示例

- **示例 1** – 删除名为 LDS 的数据服务器和名为 *pubs2* 的数据库的逻辑连接：

  ```
  drop logical connection to LDS.pubs2
  ```

用法

- 卸除热备份应用时，使用此命令可以在删除逻辑连接。
- 必须先删除到备用数据库的连接才能删除逻辑连接。

权限

drop logical connection 需要“sa”权限。

另请参见

- create connection （第 214 页）

- create logical connection （第 252 页）
- drop connection （第 297 页）
- switch active （第 338 页）

drop partition

从 Replication Server 删除磁盘分区。

语法

```
drop partition logical_name
```

参数

- **logical_name** – 为使用 **create partition** 创建的分区指派的名称。

示例

- **示例 1** – 从 Replication Server 中删除名为 *P1* 的分区：

  ```
  drop partition P1
  ```

用法

- 使用 **drop partition** 可以删除磁盘分区。此命令先将该分区标记为“pending drop”。进行标记后，将不会再向该分区中写入任何新数据。
  成功传递完此分区上存储的所有数据后，将会删除此分区。

  注意： 如果不是所有存储在分区的数据均已准备好删除，**drop partition** 会导致混乱行为。例如，如果某个分区队列包含仅部分填充的段，则在此段填满之前将无法删除此队列。由于分区已指定为“pending drop”，此段将无法填满，而此命令也将无法删除此分区。
- 有关从出现故障的分区中进行恢复的完整讨论，请参见《Replication Server 管理指南第二卷》。

权限

drop partition 需要“sa”权限。

另请参见

- admin disk_space （第 50 页）
- alter partition （第 149 页）
- create partition （第 253 页）

drop publication

删除发布及其所有项目，并且可以选择删除这些项目的复制定义。

语法

```
drop publication pub_name
with primary at data_server.database
[drop_repdef]
```

参数

- **pub_name** – 要删除的发布的名称。
- **with primary at data_server.database** – 指定主数据的位置。如果主数据库是热备份应用程序的一部分，则 *data_server.database* 是逻辑数据服务器和数据库的名称。
- **drop_repdef** – 可选的关键字，用于删除此发布的项目的复制定义（如果这些复制定义未在别处使用）。

示例

- **示例 1** – 删除主数据库 TOKYO_DS.*pubs2* 的名为 *pubs2_pub* 的发布：

  ```
  drop publication pubs2_pub
   with primary at TOKYO_DS.pubs2
  ```

- **示例 2** – 删除主数据库 TOKYO_DS.*pubs2* 的名为 *pubs2_pub* 的发布。此命令同时删除此发布的项目的所有复制定义（对于未在别处使用的复制定义）：

  ```
  drop publication pubs2_pub
   with primary at TOKYO_DS.pubs2
   drop_repdef
  ```

用法

- 使用 **drop publication** 可以删除发布。应在管理存储主数据的数据库的 Replication Server 上执行 **drop publication**。
- 只能删除没有预订的发布。如果需要，可首先删除预订。
- 删除发布时，它所包含的项目也将被删除。如果发布所包含的项目的复制定义不是其他任何项目的一部分并且没有预订，您也可以选择删除这些复制定义。
- 在复制节点上对被删除的发布执行 **define/create subscription** 或 **check publication** 时，将从此节点中删除此发布。

权限

drop publication 需要“create object”权限。

另请参见

- check publication （第 176 页）
- create publication （第 254 页）
- drop article （第 296 页）
- drop function replication definition （第 301 页）
- drop replication definition （第 308 页）
- drop subscription （第 312 页）

drop replication definition

删除复制定义及其函数。

语法

```
drop replication definition replication_definition
```

参数

- **replication_definition** – 要删除的复制定义的名称。

示例

- **示例 1** – 删除名为 *publishers_rep* 的复制定义及其全部现有的函数字符串：

  ```
  drop replication definition publishers_rep
  ```

用法

- 使用 **drop replication definition** 可以删除复制定义。删除复制定义之前，必须先删除其所有预订。
- 应在复制定义的主 Replication Server 上执行 **drop replication definition**。
- 如果删除的复制定义是 Adaptive Server 中存储的主表的最后一个复制定义，则应当在删除此复制定义后，在数据库中执行 **sp_setreplicate**。将此表的复制状态设置位“false”，使 Adaptive Server 停止记录此表的特殊复制记录。
- 如果使用 Replication Server 的多个版本（例如，Replication Server 11.5 和 11.0.*x* 版），并且创建了同一主表的多个复制定义，则会标记所创建的第一个复制定义（具有相同的主表和复制表名称以及相同的主列和复制列名称，且不包括表所有者的名称）并将其传播到 11.0.*x* 版或更低版本的 Replication Server。

  删除传播到 11.0.*x* 版或更低版本的 Replication Server 的复制定义后，则会将与 11.0.*x* 版兼容的最早的复制定义（如果有）传播到 11.0.*x* 版或更低版本的节点。有关在混合版本环境中使用复制定义的详细信息，请参见 **create replication definition**。

- Replication Server 通过复制系统将有关被删除的复制定义的信息分发到合格的节点。由于通常复制系统会有一定时间的滞后，因此更改不会立即显示在所有这些节点上。

权限

drop replication definition 需要“create object”权限。

另请参见

- alter replication definition （第 152 页）
- check subscription （第 177 页）
- create replication definition （第 258 页）
- create subscription （第 280 页）
- define subscription （第 290 页）
- drop article （第 296 页）
- drop publication （第 307 页）
- drop subscription （第 312 页）
- rs_send_repserver_cmd （第 546 页）

drop route

关闭到另一个 Replication Server 的路由。

语法

```
drop route to dest_replication_server
 [with primary at dataserver.database]
 [with nowait]
```

参数

- **dest_replication_server** – 要删除其路由的 Replication Server 的名称。
- **with primary** – 指定您要从中删除专用路由的主数据库连接。
- **with nowait** – 指示 Replication Server 关闭路由，即使它不能与目标 Replication Server 通信。仅在万不得已的情况下使用 **with nowait**。这样会强制 Replication Server 删除具有预订或某个间接路由使用的路由。要从受到影响的 Replication Server 的 RSSD 中删除无效引用，通常需要执行其它步骤。

示例

- **示例 1** – 删除在其中输入此命令的节点到 SYDNEY_RS Replication Server 的路由：

```
drop route to SYDNEY_RS
```

- **示例 2** – 要删除 NY_DS.pdb1 主连接的 RS_NY 主 Replication Server 和 RS_LON 复制 Replication Server 之间的专用路由，请在 RS_NY 上输入：

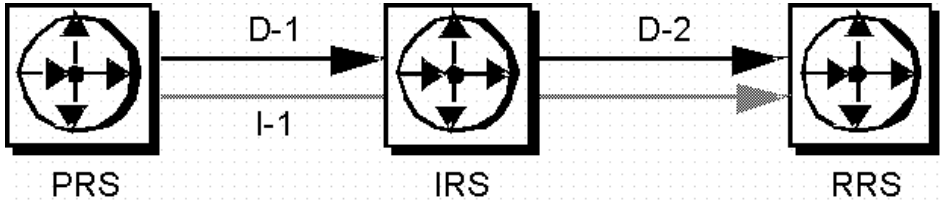
```
drop route to RS_LON
 with primary at NY_DS.pdb1
 go
```

用法

- **drop route** 关闭在其中输入此命令的 Replication Server 到指定 Replication Server 的路由。
- 必须先删除共享路由，然后才能删除专用路由。
  在删除专用路由后，从指定的主连接到目标 Replication Server 的事务将通过共享路由。
  请参见《Replication Server 管理指南第二卷》的“性能调优”的“Multi-Path Replication”中的“专用路由”。
- 删除路由之前，必须执行以下操作：
  - 在目标 Replication Server 上删除对源 Replication Server 管理的数据库中的主数据的所有预订。
  - 删除使用此路由的所有间接路由。

  例如，在下图中，路由 I-1 是从主 Replication Server (PRS) 经由中间 Replication Server (IRS)，到复制 Replication Server (RRS) 的间接路由。它使用直接路由 D-1 和 D-2。

图 4：直接路由和间接路由示例



删除直接路由 D-2 之前，必须在复制 Replication Server 上删除对主 Replication Server 或中间 Replication Server 上的复制定义的所有预订，然后才能删除间接路由 I-1。

警告！ 仅在万不得已的情况下使用 **with nowait** 子句。只有当您不打算使用目标 Replication Server 时，或者在目标 Replication Server 不可用时必须删除源自源 Replication Server 的路由时，或者您试图为直接路由添加或更改登录名和口令时，才可以使用 **with nowait** 子句。应尽量避免使用 **with nowait** 子句，这样才能正确地更新目标 Replication Server。

该子句会强制 Replication Server 删除路由，即使该路由在路由的出站队列中包含事务。因此，Replication Server 可能会放弃主连接中的一些事务。该子句指示 Replication Server 删除专用路由，即使该路由不能与目标 Replication Server 通信。

在使用 **with nowait** 子句后，请使用 **sysadmin purge_route_at_replicate** 命令从复制 Replication Server 上的系统表中删除对主 Replication Server 的所有引用，例如预订和路由信息。

- 使用 **with nowait** 删除路由后，可以在（以前的）目标节点上使用 **sysadmin purge_route_at_replicate** 删除目标系统表中的预订和路由信息。
- 如果要从中删除路由的 Replication Server 是另一个 Replication Server 的中间节点，则不能删除此路由。有关详细信息，请参见《Replication Server 管理指南第一卷》。
- 对于包含 ERSSD 的 Replication Server，如果要删除的路由是源自于此源的最后一个路由，则会：
  - 关闭 ERSSD Replication Agent
  - 删除路由结束时，会从 ERSSD 关闭日志传送

权限

drop route 需要“sa”权限。

另请参见

- alter route （第 161 页）
- create connection （第 214 页）
- create route （第 273 页）
- sysadmin purge_route_at_replicate （第 369 页）

drop schedule

删除执行命令的日程表。

语法

drop schedule *sched_name*

参数

- **sched_name** – 要删除的日程表的名称。

示例

- **示例 1** – 若要删除 **schedule1**，请输入：

```
drop schedule schedule1
```

用法

从 Replication Server 中删除日程表。

权限

drop schedule 需要“sa”权限。

另请参见

- admin schedule （第 59 页）
- alter schedule （第 168 页）
- create schedule （第 277 页）

drop subscription

删除对数据库复制定义、表复制定义、函数复制定义、项目或发布的预订。

语法

```
drop subscription sub_name
for {table_rep_def | function_rep_def |
{article article_name in pub_name |
    publication pub_name | database replication definition db_repdef
        with primary at data_server.database}
 with replicate at data_server.database
[without purge [with suspension
     [at active replicate only]] |
   [incrementally] with purge]
```

参数

- **sub_name** – 要删除的预订的名称。如果要删除对发布中某个项目的预订，请指定此发布预订的名称。
- **for table_rep_def** – 指定所预订的表复制定义的名称。
- **for function_rep_def** – 指定所预订的函数复制定义的名称。
- **for article article_name in pub_name** – 指定预订的项目的名称及包含此项目的发布的名称。
- **for publication pub_name** – 指定所预订的发布的名称。
- **for database replication definition db_repdef** – 指定所预订的数据库复制定义的名称。
- **with primary at data_server.database** – 指定主数据的位置。如果主数据库是热备份应用程序的一部分，则 *data_server.database* 是逻辑数据服务器和数据库的名称。此子句仅用于对发布的预订或对项目的预订。
- **with replicate at data_server.database** – 指定复制数据的位置。如果复制数据库是热备份应用程序的一部分，则 *data_server.database* 是逻辑数据服务器和数据库的名称。
- **without purge** – 指示 Replication Server 将预订复制的行保留在复制副本中。

通常，删除对函数复制定义的预订时，不清除复制数据。对于表复制定义或发布的预订，必须选择 **without purge** 或 **with purge**。对于数据库复制定义的预订，必须选择 **without purge**。

- **with suspension** – 用于 **without purge** 子句，在删除预订后挂起 DSI，以便您可以手动删除预订行。如果数据库是热备份应用程序的一部分，**with suspension** 将挂起活动数据库和备用数据库的 DSI 线程。您可以删除这两个数据库中的预订行。
- **with suspension at active replicate only** – 用于 **without purge** 子句，在删除预订后挂起 DSI，以便您可以手动删除预订行。在热备份应用程序中，备用 DSI 不挂起。这样可以使 Replication Server 将删除事务从活动数据库复制到备用数据库。
- **incrementally** – 用于 **with purge** 子句，指定每次对 1000 行执行删除操作。
- **with purge** – 用于表复制定义、项目或发布，指示 Replication Server 删除复制表中通过预订复制的行。

  通常，删除对函数复制定义的预订时，不清除复制数据。对于表复制定义或发布的预订，必须选择 **without purge** 或 **with purge**。

示例

- **示例 1** – 删除 *authors_rep* 表复制定义的 *authors_sub* 预订。复制数据位于 SYDNEY_DS 数据服务器的 *pubs2* 数据库中。如果通过此预订复制的行不属于其它预订，则会将其从复制表中清除：

```
drop subscription authors_sub
  for authors_rep
    with replicate at SYDNEY_DS.pubs2
    with purge
```

- **示例 2** – 删除 *titles_rep* 表复制定义的 *titles_sub* 预订。复制数据位于 SYDNEY_DS 数据服务器的 *pubs2* 数据库中。通过此预订复制的行保留在复制表中：

```
drop subscription titles_sub
  for titles_rep
    with replicate at SYDNEY_DS.pubs2
    without purge
```

- **示例 3** – 删除 *myproc_rep* 函数复制定义的 *myproc_sub* 预订。复制数据位于 SYDNEY_DS 数据服务器的 *pubs2* 数据库中。不会清除任何预订数据：

```
drop subscription myproc_sub
  for myproc_rep
    with replicate at SYDNEY_DS.pubs2
```

- **示例 4** – 删除发布 *pubs2_pub* 的预订 *pubs2_sub* 中对项目 *titles_art* 的预订。主数据在 TOKYO_DS 数据服务器的 *pubs2* 数据库中，复制数据在 SYDNEY_DS 数据服务器的 *pubs2* 数据库中。通过此预订复制的行仍保留在受影响的复制表中。删除此项目预订后，您可以删除此项目：

```
drop subscription pubs2_sub
  for article titles_art in pubs2_pub
    with primary at TOKYO_DS.pubs2
```

```
    with replicate at SYDNEY_DS.pubs2
    without purge
```

- **示例 5** – 删除 *pubs2_pub* 发布的名为 *pubs2_sub* 的预订，其主数据在 TOKYO_DS 数据服务器的 *pubs2* 数据库中，复制数据在 SYDNEY_DS 数据服务器的 *pubs2* 数据库中。如果通过此预订复制的行不属于其它预订，则会将其从受影响的复制表中清除：

```
drop subscription pubs2_sub
  for publication pubs2_pub
    with primary at TOKYO_DS.pubs2
    with replicate at SYDNEY_DS.pubs2
    with purge
```

- **示例 6** – 删除名为 *pubs2_sub* 的数据库预订。**without purge** 选项可确保 Replication Server 不会删除通过预订添加到复制数据库中的行：

```
drop subscription pubs2_sub
  for database replication definition pubs2_rep
    with primary at NEWYORK_DS.pubs2
    with replicate at TOKYO_DS.pubs2
    without purge
```

用法

- 删除预订时，Replication Server 停止复制此预订指定的数据。
- 应在创建预订的 Replication Server 上执行 **drop subscription**。
- 删除对表复制定义、函数复制定义、项目或发布的所有预订后，才能删除这些对象。

without purge 子句

- 使用 **without purge** 可以删除对表复制定义、数据库复制定义或发布的预订。复制行将保留在复制表中。
- 删除对表复制定义或发布的预订时，必须指定 **without purge** 或 **with purge**。
- 删除对函数复制定义的预订时，始终会使用“without purge”来执行删除，您无需指定 **without purge**。
- 使用“without purge”删除发布预订时，其全部项目预订也将一起被删除。

with purge 子句

- 使用 **with purge** 子句可以删除复制表中由此预订复制的行。所有不属于复制节点上其它预订的预订行都将被清除。
- 使用 **with purge** 时，Replication Server 将从复制数据库中选择可以删除的行集。然后检查所选的行是否属于其它预订，以确定是否将其删除。复制数据库的维护用户必须具有对此表的 **select** 权限。
- 使用 **with purge** 进行的删除在单个事务中发生，由复制数据库中的 **rs_select_with_lock** 函数字符串执行。

- 使用 **with purge** 和 **incrementally** 进行的删除每次对 1000 行执行操作。此操作是由复制数据库中的 **rs_select** 函数字符串执行的。
- 使用“with purge”删除发布预订时，将按照项目添加到发布的相反顺序，每次删除一个项目预订。

权限

drop subscription 要求对复制节点具备“create object”权限，对主 Replication Server 具有“primary subscribe”权限。

drop subscription ... with purge 还要求维护用户具有复制表的 **select** 权限。

另请参见

- check subscription （第 177 页）
- create subscription （第 280 页）
- define subscription （第 290 页）
- drop article （第 296 页）
- drop function replication definition （第 301 页）
- drop publication （第 307 页）
- drop replication definition （第 308 页）
- resume connection （第 321 页）
- rs_select （第 424 页）
- rs_select_with_lock （第 426 页）

drop user

删除 Replication Server 用户登录名。

语法

```
drop user user
```

参数

- **user** – 要删除的用户登录名。

示例

- **示例 1** – 从 Replication Server 中删除登录名“louise”：

```
drop user louise
```

用法

- 使用 **drop user** 可以删除 Replication Server 登录名。
- 应在创建登录名的 Replication Server 上执行此命令。

权限

drop user 需要“sa”权限。

另请参见

- alter user （第 170 页）
- create user （第 288 页）

grant

将权限指派给用户。

语法

```
grant {sa | create object | primary subscribe |
 connect source}
 to user
```

参数

- **sa** – 具有“sa”权限的用户可以执行任何 RCL 命令。
- **create object** – 允许接收者创建、更改和删除 Replication Server 对象，例如复制定义、预订和函数字符串。
- **primary subscribe** – 允许接收者创建对复制表的预订，此复制表的主数据由当前 Replication Server 管理。
- **connect source** – 此权限授予 RepAgent 和其它 Replication Server 以登录到此 Replication Server。
- **user** – 要获得此权限的用户的登录名。

示例

- **示例 1** – 允许用户“thom”执行任何 Replication Server 命令：

  ```
  grant sa to thom
  ```

- **示例 2** – 允许用户“louise”创建预订：

  ```
  grant primary subscribe to louise
  ```

用法

- 不能撤消“sa”用户的“sa”权限。
- RSI 或 RepAgent 需要具有“connect source”权限。有关详细信息，请参见适用于您的平台的 Replication Server 安装和配置指南。
- 对于本手册中介绍的每个 RCL 命令，都给出了执行此命令所需的最低权限。有关所有命令的最低权限的列表，请参见《Replication Server 管理指南第一卷》。

权限

grant 需要“sa”权限。

另请参见

- revoke （第 327 页）

ignore loss

允许 Replication Server 在检测到丢失后接受消息。

语法

```
ignore loss
 from data_server.database
[to {data_server.database | replication_server}]
```

参数

- **from data_server.database** – 指定将忽略其消息丢失的主数据服务器和数据库。
- **to data_server.database** – 为丢失的消息指定目标数据服务器和数据库。
- **to replication_server** – 为丢失的消息指定目标 Replication Server。

用法

- Replication Server 在恢复模式下重建队列或重放事务日志时检测丢失。
- Replication Server 对与它所管理的复制数据库的连接检测消息丢失。
- 对于热备份数据库，除了 Replication Server 在活动数据库和备用数据库之间检测到丢失的之外，应使用 *data_server.database* 的逻辑连接名。要忽略这些丢失，使用实际的 *data_server.database* 名称。
- 如果存在直接路由，则目标 Replication Server 从源 Replication Server 检测消息丢失。查看两个 Replication Server 的日志文件以确定是否检测到丢失。
- 当 Replication Server 检测到丢失时，在执行 **ignore loss** 之前它不接受此连接的任何消息。
- 执行 **ignore loss** 之后，可能需要稍作更新才能使消息重新开始传播。

- 执行 **ignore loss** 之后，需要采取一些步骤来更新复制数据。

有关详细的恢复说明，请参见《Replication Server 管理指南第二卷》。

权限

ignore loss 需要“sa”权限。

另请参见

- allow connections （第 106 页）
- configure route （第 200 页）
- rebuild queues （第 320 页）
- set log recovery （第 330 页）

move primary

更改错误类或函数字符串类的主 Replication Server。

语法

```
move primary
 of {[replication server] error class | function string class}
class_name
 to replication_server
```

参数

- **replication server** – 指定此参数可修改 Replication Server 错误类。省略此参数则修改数据服务器错误类。
- **error class** – 指定要更改某错误类的主 Replication Server。
- **function string class** – 指定要更改函数字符串类的主 Replication Server。
- **class_name** – 要更改其主 Replication Server 的错误类或函数字符串类的名称。
- **replication_server** – 为错误类或函数字符串类指定新的主 Replication Server。由于 **move primary** 必须在新的主 Replication Server 上执行，因此它是执行此命令的 Replication Server 的名称。

示例

- **示例 1** – 将 *pubs2_db_err_class* 错误类的主 Replication Server 更改为 SYDNEY_RS Replication Server。此命令是在 SYDNEY_RS 上输入的：

```
move primary
   of error class pubs2_db_err_class
   to SYDNEY_RS
```

- **示例 2** – 将 Replication Server 错误类 *my_rs_error_class* 的主 Replication Server 更改为 SYDNEY_RS Replication Server。此命令是在 SYDNEY_RS 上输入的：

```
move primary
   of replication server error class my_rs_error_class
   to SYDNEY_RS
```

- **示例 3** – 将 *sqlserver2_function_class* 函数字符串类的主 Replication Server 更改为 SYDNEY_RS Replication Server。此命令是在 SYDNEY_RS 上输入的：

```
move primary
   of function string class sqlserver2_function_class
   to SYDNEY_RS
```

用法

- 如果您已经更改了路由配置，请使用 **move primary** 确保错误响应和函数字符串可通过新的路由分发到需要它们的 Replication Server。
- **move primary** 必须在新的主 Replication Server 上执行。
- 只有在从 A 到 B 和从 B 到 A 都存在路由的情况下，才能使用 **move primary** 将主 Replication Server 从 A 更改为 B。
- 系统提供的 *rs_sqlserver_function_class* 没有主点，除非您为其指派一个。*rs_default_function_class* 和 *rs_db2_function_class* 都是系统提供的，它们不能修改，也没有主节点。
- 派生函数字符串类的主节点是其父类的节点，除非父类是 *rs_default_function_class* 或 *rs_db2_function_class*。在此情况下，派生类的主节点就是它创建时所在的节点。
- 如果使用 *rs_sqlserver_function_class*，在修改缺省函数字符串之前，必须指定主节点。要指定函数字符串类的主节点，请在主节点上执行 **create function string class rs_sqlserver_function_class**。然后使用 **move primary** 命令更改此类的主节点。
- 缺省错误类 *rs_sqlserver_error_class* 和 *rs_repserver_error_class* 没有主节点，除非您为其指派一个。使用 **assign action** 更改缺省错误操作之前，必须指定主节点。若要指定主节点，请在主节点上执行 **create error class rs_sqlserver_error_class** 或 **create replication server error class rs_repserver_error_class**。然后才能使用 **move primary** 更改此主节点。

权限

move primary 需要“sa”权限。

另请参见

- alter error class （第 138 页）
- alter route （第 161 页）
- assign action （第 172 页）
- create error class （第 228 页）
- create function string class （第 249 页）

rebuild queues

重建 Replication Server 稳定队列。

语法

```
rebuild queues
```

用法

- 重建稳定队列，以从出现故障的分区或丢失的分区中恢复。

  警告！ 只能按照《Replication Server 管理指南第二卷》中的说明使用此命令。**rebuild queues** 将删除复制系统中的消息，可能会导致纠正其它问题更加困难。
- 必要时，在重建队列之前，删除已损坏的分区并替换它们。在执行 **rebuild queues** 之前，被删除的分区实际上可能并未从系统中删除。
- **rebuild queues** 将断开执行此命令的 Replication Server 与所有其它 Replication Server 的连接。队列重建完成之前，连接尝试将被拒绝。
- **rebuild queues** 将清除 Replication Server 的所有稳定队列，并“放弃”所有正在使用的已损坏分区。
- 如果在独立模式下启动 Replication Server（使用 **-M** 命令行标志），然后执行 **rebuild queues**，Replication Server 将进入恢复模式。
- 在恢复重建稳定队列的消息时，Replication Server 将确定从队列中清除的数据已恢复还是已丢失。查看包含重建队列的 Replication Server 的日志文件中的错误消息，以及与此 Replication Server 之间存在直接路由的那些 Replication Server 的日志文件中的错误消息。丢失检测可能不会立即完成，因为从每个主数据库或上游节点流入新的数据都需要一定的时间。
- 如果检测到丢失，可能需要重新创建预订或从脱机转储中恢复数据。
- 如果使用 **rebuild queues** 时某个预订正在实现，请将其删除，然后再重新创建。即使实现进程看起来已成功完成，某些数据仍可能已经丢失。
- 重建队列之后，Replication Server 将从与当前 Replication Server 之间存在路由的 Replication Server 请求积压消息来尝试恢复丢失的消息。
- 无法为某些特定的数据库连接或路由重建队列。

有关恢复过程的帮助，请参见《Replication Server 管理指南第二卷》。

权限

rebuild queues 需要“sa”权限。

另请参见

- add partition （第 47 页）
- alter partition （第 149 页）

- configure connection （第 180 页）
- create partition （第 253 页）
- drop partition （第 306 页）
- ignore loss （第 317 页）
- resume log transfer （第 324 页）
- set log recovery （第 330 页）

resume connection

恢复挂起的连接。

语法

```
resume connection to data_server.database
    [skip [n] transaction | execute transaction | skip to resync
marker]
```

参数

- **data_server** – 数据服务器名称，此服务器包含要恢复其连接的数据库。
- **数据库** – 要恢复其连接的数据库的名称。
- **skip [n] transaction** – 指示 Replication Server 在恢复连接前跳过连接队列中指定数目的事务。跳过的事务会写入数据库例外日志，并会写入 Replication Server 日志或 **sysadmin dump_file** 命令指定的备用日志文件。**resume connection** 可以跳过的最大事务数为 DSI 出站队列中的事务的数量。

  如果未指定 *n*，则 Replication Server 会恢复执行连接队列中的第二个事务。
- **execute transaction** – 如果系统事务是 DSI 队列中的第一个事务，在 DSI 启动后，将覆盖针对此系统事务应用的 Replication Server 限制。
- **skip to resync marker** – 指示 Replication Server 跳过指定复制数据库的 DSI 外发队列中的事务，直到 Replication Server 收到并确认 Replication Agent 发送的 dump database 标记。Replication Server 路过出站队列中记录的处理，因为复制数据库中的数据将替换为转储内容。

示例

- **示例 1** – 恢复到 SYDNEY_DS 数据服务器中 *pubs2* 数据库的连接：

  ```
  resume connection to SYDNEY_DS.pubs2
  ```
- **示例 2** – 在跳过两个事务后，恢复到 SYDNEY_DS 数据服务器中 *pubs2* 数据库的连接。事务会记录在数据库例外日志和 Replication Server 日志中：

  ```
  resume connection to SYDNEY_DS.pubs2 skip 2 transaction
  ```

- **示例 3** – 在跳过两个事务后，恢复到 SYDNEY_DS 数据服务器中 *pubs2* 数据库的连接。事务会记录在数据库例外日志和 SYDNEY_RS.log 文件中。最后的 **sysadmin dump_file** 命令将关闭 SYDNEY_RS.log 文件：

```
sysadmin dump_file SYDNEY_RS.log
resume connection to SYDNEY_DS.pubs2 skip 2 transaction
sysadmin dump_file
```

- **示例 4** – 指示 Replication Server 删除复制数据库外发队列中的数据，并等待来自主数据库 Replication Agent 的 resync 标记：

```
resume connection to SYDNEY_DS.pubs2 skip to
resync marker
```

用法

- 重新开始连接可以使挂起数据库的复制活动重新开始。
- 挂起连接后才能使用 **alter connection** 来更改连接，或对挂起的数据库进行维护。在预订实现或取消实现期间也将挂起连接。
- Replication Server 可能由于发生错误而挂起数据库连接。
- **resume connection** 也可用于恢复因错误而挂起的连接。
- 如果确定某系统事务已经执行，可使用 **skip transaction** 子句。
- 只有在执行系统事务失败，而且您已经纠正了妨碍其执行的问题之后，才能使用 **execute transaction** 子句。系统事务不包含 **begin tran/commit tran** 对。如果 Replication Server 重新启动时用某系统事务作为第一个事务，您会看到以下消息：

```
E. 1998/02/16 14:43:49. ERROR #5152 DSI (206 hookip01.rdb1) -
dsisched.c (2196)
 There is a system transaction whose state is not known. DSI will
be shut down.
```

  确定数据库是否已执行此事务，并根据具体情况使用 **skip transaction** 或使用 **execute transaction**。

- 在设置 **skip to resync** 后，Replication Server 不会在 Replication Server 日志或数据库例外日志中记录跳过的事务。在您设置 **skip [n] transaction** 后，Replication Server 将记录跳过的事务。

  如果在执行带有 **skip to resync marker** 的 **resume connection** 后，Replication Agent 不发出正确的标记，或您对错误的连接发出了该标记，或者出于其它原因 DSI 连接不处理 resync database 标记，您可以恢复正常复制处理而不等待 resync database 标记，方法是执行 **suspend connection**，然后执行不带 **skip to resync** 选项的 **resume connection**。

  注意： 如果在错误的连接上执行带有 **skip to resync marker** 选项的 **resume connection**，则复制数据库上的数据将变得不同步。

权限

resume connection 需要“sa”权限。

另请参见

- activate subscription （第 44 页）
- alter connection （第 109 页）
- assign action （第 172 页）
- create connection （第 214 页）
- drop connection （第 297 页）
- drop subscription （第 312 页）
- suspend connection （第 334 页）

resume distributor

为与数据库的连接恢复挂起的分配器线程。

语法

```
resume distributor data_server.database [skip transaction]
```

参数

- **data_server** – 数据服务器名称。如果数据库是热备份应用程序的一部分，则 *data_server* 是逻辑数据服务器名称。
- **database** – 数据库名称。如果数据库是热备份应用程序的一部分，则 *database* 是逻辑数据库名称。
- **skip transaction** – 指示 Replication Server 恢复执行连接队列中的第二个事务。第一个事务将被写入数据库例外日志。

示例

- **示例 1** – 恢复逻辑数据服务器 LDS 和 *pubs2* 数据库的分配器线程：

  ```
  resume distributor LDS.pubs2
  ```

用法

- 使用 **resume distributor** 可以重新开始用 **suspend distributor** 挂起的分发器线程或由 Replication Server 挂起的分发器线程。
- 当分配器由于以下原因关闭时，请使用 **skip transaction** 恢复连接：
  - 入站队列中的消息长度超过 16,000 个字节，并且节点版本尚未升级到 Replication Server 12.5 和更高版本，或者
  - 下游 Replication Server 不能接受新功能命令，例如，*bigint*。

权限

resume distributor 需要“sa”权限。

另请参见

- suspend distributor （第 335 页）

resume log transfer

使 RepAgent 能够连接到 Replication Server。

语法

```
resume log transfer
 from {data_server.database | all}
```

参数

- **data_server** – 数据服务器的名称，其中包含的数据库的 RepAgent 将连接到 Replication Server。
- **database** – 其 RepAgent 将连接到 Replication Server 的数据库。
- **all** – 允许 Replication Server 所管理的所有数据库的 RepAgent 进行连接。

示例

- **示例 1** – Replication Server 将接受来自任何 RepAgent 的连接：

  ```
  resume log transfer from all
  ```

- **示例 2** – Replication Server 将接受来自 SYDNEY_DS 数据服务器中 *pubs2* 数据库的 RepAgent 的连接：

  ```
  resume log transfer from SYDNEY_DS.pubs2
  ```

用法

- 停顿 Replication Server 或复制系统时，使用 **suspend log transfer** 使 Replication Server 拒绝 RepAgent 连接。
- **resume log transfer** 允许 RepAgent 线程连接到已执行 **suspend log transfer** 的 Replication Server。
- 通常，执行 **suspend log transfer** 之后，RepAgent 会重试到 Replication Server 的连接，直至 **resume log transfer** 允许它重新连接为止。但是，如果 RepAgent 由于任何原因而关闭，则 **resume log transfer** 不会重新启动它。
- 在从 ERSSD 恢复日志传送后，恢复守护程序将在 ERSSD RepAgent 唤醒时自动重新启动它。

权限

resume log transfer 需要“sa”权限。

另请参见

- admin quiesce_check （第 57 页）
- admin quiesce_force_rsi （第 58 页）
- resume connection （第 321 页）

resume queue

重新启动在传送超过 16K 个字节的消息后停止的稳定队列。仅适用于 Replication Server 为 12.5 版或更高版本而节点版本尚未进行类似升级的情况。

语法

```
resume queue, q_number, q_type [, skip transaction with large
message]
```

参数

- **q_number** – 稳定队列的队列号。
- **q_type** – 稳定队列的队列类型。值为“0”表示出站队列，值为“1”表示入站队列。
- **skip transaction with large message** – 指定 SQM 在重新启动后应跳过遇到的第一条大消息。

示例

- **示例 1** – 指定 2 号出站队列跳过 RepAgent 传递给它的第一条大消息：

  ```
  resume queue, 2, 0, skip transaction with large message
  ```

用法

- 只有在 Replication Server 为 12.5 版或更高版本并且节点版本未升级时，才能使用此命令。
- 如果节点为 12.5 或更高版本，则 **resume queue** 不会跳过任何消息。

权限

resume queue 需要“sa”权限。

另请参见

- alter queue （第 150 页）

resume route

恢复挂起的路由。

语法

```
resume route to dest_replication_server
 [with primary at dataserver.database |
 skip transaction with large message]
```

参数

- **dest_replication_server** – 目标 Replication Server 的名称；即要恢复的挂起的路由。
- **with primary** – 指定您要为其恢复专用路由的主数据库连接。
- **skip transaction with large message** – 跳过遇到的第一个消息超过 16,000 个字节的事务。

示例

- **示例 1** – 恢复到 SYDNEY_RS Replication Server 的路由：

  ```
  resume route to SYDNEY_RS
  ```

- **示例 2** – 要恢复 `NY_DS.pdb1` 主连接的 RS_NY 主 Replication Server 和 RS_LON 复制 Replication Server 之间的专用路由，请在 RS_NY 上输入：

  ```
  resume route to RS_LON
   with primary at NY_DS.pdb1
   go
  ```

用法

- 重新开始路由使 Replication Server 可以开始再次将已入队消息发送到远程 Replication Server。
- **resume route** 也可用于恢复因错误而挂起的路由。
- **skip transaction with large message** 仅适用于复制节点上节点版本为 12.1 或更低版本的直接路由。

权限

resume route 需要“sa”权限。

另请参见

- alter route （第 161 页）
- create route （第 273 页）

- drop route （第 309 页）
- suspend route （第 337 页）

revoke

撤消用户的权限。

语法

```
revoke {sa | connect source | create object |
 primary subscribe}
 from user
```

参数

- **sa** – 拒绝执行要求“sa”权限才能执行的命令的权限。
- **connect source** – 拒绝为 RepAgent 或其它 Replication Server 所使用的 RCL 命令授予执行权限。
- **create object** – 拒绝创建、更改和删除 Replication Server 对象（如复制定义、预订和函数字符串）所需的权限。
- **primary subscribe** – 如果主数据由当前 Replication Server 管理，拒绝创建对复制表的预订所需的权限。
- **user** – 要撤消其权限的用户的登录名。

示例

- **示例 1** – 禁止用户“thom”执行用于创建或修改 Replication Server 对象的命令：

  ```
  revoke create object from thom
  ```

- **示例 2** – 禁止用户“louise”创建对此 Replication Server 管理的主数据的预订，除非她在主 Replication Server 上具有“create object”或“sa”权限：

  ```
  revoke primary subscribe from louise
  ```

用法

- **revoke** 需要“sa”权限。
- 不能撤消“sa”用户登录名的“sa”权限。

权限

revoke 需要“administrator”权限。

另请参见

- create replication definition （第 258 页）

- check subscription （第 177 页）
- create user （第 288 页）
- grant （第 316 页）

set

控制复制连接的复制定义属性。

语法

```
set {autocorrection | dynamic_sql} {on | off}
 for replication_definition
with replicate at data_server.database
```

参数

- **autocorrection** – 防止由于复制表中丢失行或出现重复行而导致的故障。缺省值为 off。
- **dynamic_sql** – 控制对于动态 SQL 应用程序，是否考虑表。缺省值为 on。
- **on** – 为指定的复制定义启用自动更正或动态 SQL。
- **off** – 为指定的复制定义禁用自动更正或动态 SQL。
- **replication_definition** – 要更改其自动更正或动态 SQL 状态的复制定义的名称。
- **data_server** – 数据服务器名称，此数据服务器包含要更改其自动更正或动态 SQL 状态的复制数据库。如果复制数据库是热备份应用程序的一部分，则 *data_server* 是逻辑数据服务器名称。
- **database** – 要更改其自动更正或动态 SQL 状态的复制数据库的名称。如果复制数据库是热备份应用程序的一部分，则 *database* 是逻辑数据库名称。

示例

- **示例 1** – 启用对 SYDNEY_DS 数据服务器上 *pubs2* 数据库中的 *publishers_rep* 复制定义的自动更正：

  ```
  set autocorrection on
   for publishers_rep
   with replicate at SYDNEY_DS.pubs2
  ```

- **示例 2** – 为 SYDNEY_DS 数据服务器上的 *pubs2* 数据库中的 *publishers_rep* 复制定义禁用动态 SQL：

  ```
  set dynamic_sql off
  for publishers_rep
  with replicate at SYDNEY_DS.pubs2
  ```

用法

- 使用 **set dynamic_sql off** 可对指定的复制定义和复制连接禁用动态 SQL 命令。
- 使用 **set autocorrection** 可防止非原子实现期间可能发生的重复键错误。
- 只有在复制定义的预订采用非原子实现（指定了带 **without holdlock** 的 **create subscription**）的情况下，才能启用对此复制定义的自动更正。完成实现并且预订的状态为“VALID”之后，禁用自动更正可以改善性能。
- 缺省情况下，对复制定义的自动更正是关闭的。

自动更正的工作方式

- **set autocorrection** 确定 Replication Server 如何处理对复制表的插入和更新操作。启用自动更正时，Replication Server 将每个更新或插入操作转换为一个删除操作，并随后执行一个插入操作。

  例如，如果插入主表的某一行已经存在于复制表中，并且自动更正已关闭，则此操作将产生一个错误。启用自动更正时，Replication Server 将插入操作转换为一个删除操作，并随后执行一个插入操作，这样此插入操作不会因某行已经存在而失败。

  如果要复制的行中的主键已经更改，则 Replication Server 在插入此行之前，先在删除复制表中删除两行。它删除的是其主键与前映像匹配的行和其主键与后映像匹配的行。
- 启用自动更正后，在主数据库上进行插入或更新操作可能会触发复制数据库上的删除和插入触发器。只有满足以下条件才能触发删除触发器：在主数据库上插入或更新的行已经存在于复制数据库上。
- Replication Server 在 *rs_repobjs* 系统表中创建启用了自动更正的复制定义条目。

自动更正与复制存储过程

- 如果由于使用修改主数据的复制存储过程而需要在复制数据上更新行，Replication Server 不会对这种行更新执行自动更正。有关复制存储过程的详细信息，请参见《Replication Server 管理指南第一卷》。

注意： 如果使用复制存储过程来修改主数据，确保在复制 Replication Server 上写入存储过程，以更正在非原子实现期间可能发生的失败的更新和插入操作。复制 Replication Server 上的存储过程应当模拟自动更正，将更新和插入操作处理为“删除 - 插入”组合操作。也可以在检测到失败的更新和插入操作后，由存储过程来更正这些操作。

自动更正与 replicate minimal columns

- 如果复制定义使用了 **replicate minimal columns**，则不能执行 **set autocorrection on**。如果在指定最少列数（例如，使用 **alter replication definition**）之前执行 **set autocorrection on**，则自动更正将不会执行。Replication Server 将记录任何更新操作的信息性消息。

自动更正和 `text`、`unitext` 或 `image` 数据类型

- 如果复制定义在 **replicate_if_changed** 列列表中有 *text*、*unitext* 或 *image* 列，则尝试对此复制定义启用自动更正会产生错误。自动更正要求所有 *text*、*unitext* 和 *image* 列出现在复制定义的 **always_replicate** 列表中。

自动更正和批量拷入

在正常复制过程中，如果将 autocorrection 设置为 on，则会禁用批量操作。不过，在预订实现过程中，即使启用了自动更正也会应用批量操作，除非是从故障中恢复的非基本预订。

权限

set 需要“create object”权限。

另请参见

- alter replication definition （第 152 页）
- create replication definition （第 258 页）
- create subscription （第 280 页）

set log recovery

指定要从脱机转储恢复其日志的数据库。

语法

```
set log recovery
 for data_server.database
```

参数

- **data_server** – 要恢复的数据库所在的数据服务器。
- **database** – 要恢复的数据库。

用法

- 以独立模式重新启动 Replication Server 之后执行 **set log recovery**。
- 使用 **set log recovery** 进入恢复模式之后再执行 **allow connections**。对于在 **set log recovery** 中指定的数据库，Replication Server 只接受来自以恢复模式启动的 RepAgent 的连接。这可确保在接受新日志记录之前重放旧日志记录。

有关详细的恢复过程，请参见《Replication Server 管理指南第二卷》。

权限

set log recovery 需要“sa”权限。

另请参见

- allow connections （第 106 页）
- ignore loss （第 317 页）
- rebuild queues （第 320 页）

set proxy

切换到另一个用户。

语法

```
set proxy [to] [user_name [verify password passwd]]
```

参数

- **user_name** – 有效的 Replication Server 登录名。
- **verify password** – 检验 Replication Server 用户的口令。
- **passwd** – 有效 Replication Server 用户的口令。

用法

- **set proxy** *user_name* 切换到新用户，具备新用户的所有权限，同时失去原用户的所有权限。
- 不管新用户是否具有“sa”权限，只要输入不带用户名的 **set proxy**，就可以切换回原来的用户。
- 如果输入了 *user_name* 的正确口令，**set proxy** *user_name* **verify password** *passwd* 允许不具备 **sa** 权限的用户切换到另一个用户。

权限

set proxy *user_name* 需要“sa”权限。任何用户都可以执行 **set proxy** 和 **set proxy** *user_name* **verify password** *passwd*。

另请参见

- alter connection （第 109 页）
- alter route （第 161 页）
- configure replication server （第 181 页）
- create connection （第 214 页）
- create route （第 273 页）

show connection

列出连接堆栈的内容。

语法

```
show connection
```

示例

- **示例 1** – 在 ID Server ost_replinuxvm_02 创建到 ost_replinuxvm_03 的网关后显示连接堆栈:

```
isql -Usa -P -S ost_replinuxvm_02
1> connect to ost_replinuxvm_03
2> go
```

```
Gateway connection to 'ost_replinuxvm_03' is created.
```

```
1> show connection
2> go
```

```
ost_replinuxvm_03
ost_replinuxvm_02(IDServer)
```

用法

- 在网关中创建的级联连接将保存在连接堆栈中，并将发出第一个 **connect** 命令的 Replication Server 放在堆栈底部。
- 在 Replication Server 15.1 或更低版本中，**disconnect** 命令具有不同的工作方式。在这些版本中，**disconnect** 命令将终止网关模式，并将工作服务器身份恢复为发出第一个 **connect** 命令的 Replication Server。如果连接堆栈中包含 Replication Server 15.2 以及 15.1 或更低版本并发出了 **disconnect** 命令，**show connection** 和 **show server** 命令可能不会显示预期的输出内容。

权限

任何用户都可以执行此命令。

另请参见

- connect （第 200 页）
- disconnect （第 295 页）
- show server （第 333 页）

show server

显示给定连接堆栈的当前工作服务器。

语法

```
show server
```

示例

- **示例 1** – 在创建从 ost_replinuxvm_02 到 ost_replinuxvm_03 的连接后，显示当前工作服务器：

```
isql -Usa -P -S ost_replinuxvm_02
1> connect to ost_replinuxvm_03
2> go
```

```
Gateway connection to 'ost_replinuxvm_03' is created.
```

```
1> show server
2> go
```

```
ost_replinuxvm_03
```

用法

有关用法信息，请参见 **show connection**。

权限

任何用户都可以执行此命令。

另请参见

- connect （第 200 页）
- disconnect （第 295 页）
- show connection （第 332 页）

shutdown

关闭 Replication Server。

语法

```
shutdown
```

示例

- **示例 1** – 指示 Replication Server 关闭：

```
shutdown
```

用法

使用 **shutdown** 命令可关闭 Replication Server。此命令指示 Replication Server 拒绝其它连接、终止各进程，然后退出。

权限

shutdown 需要“sa”权限。

suspend connection

挂起与某一数据库的连接。

语法

```
suspend connection
 to data_server.database
[with nowait]
```

参数

- **data_server** – 其连接要被挂起的数据库所在的数据服务器的名称。
- **database** – 其连接要被挂起的数据库的名称。
- **with nowait** – 立即挂起连接。

示例

- **示例 1** – 挂起到 SYDNEY_DS 数据服务器中 *pubs2* 数据库的连接：

```
suspend connection to SYDNEY_DS.pubs2
```

用法

- 挂起连接将暂时中断数据库的复制活动。
- 必须将连接挂起，才能用 **alter connection** 更改连接，或者对连接执行维护。还可以使用 **suspend connection** 控制何时更新复制数据库。
- 某连接被挂起后，Replication Server 会将相应数据库的事务保存在稳定队列中。
- 如果执行不带 **with nowait** 子句的 **suspend connection**，Replication Server 将尝试完成所有正在进行的事务。但是，与数据服务器的连接可能在事务完成之前就被挂起。

- 要重新激活连接，请使用 **resume connection**。

权限

suspend connection 需要“sa”权限。

另请参见

- alter connection （第 109 页）
- create connection （第 214 页）
- drop connection （第 297 页）
- resume connection （第 321 页）

suspend distributor

挂起与主数据库连接的分配器线程。

语法

```
suspend distributor data_server.database
```

参数

- **data_server** – 数据服务器名称。如果数据库是热备份应用程序的一部分，则 *data_server* 是逻辑数据服务器名称。
- **database** – 数据库名称。如果数据库是热备份应用程序的一部分，则 *database* 是逻辑数据库名称。

示例

- **示例 1** – 挂起 LDS 数据服务器中 *pubs2* 数据库的分配器线程：

  ```
  suspend distributor LDS.pubs2
  ```

用法

- 使用 **suspend distributor** 挂起主数据库的逻辑连接或物理连接的分配器线程。
- 要恢复分配器线程，请使用 **resume distributor**。
- 分配器线程读取入站主数据库事务，然后将它们转发给预订者。在只包含备用数据库而没有预订者的“仅热备份”环境中，关闭分配器可以改善性能。

权限

suspend distributor 需要“sa”权限。

另请参见

- resume distributor （第 323 页）

suspend log transfer

将 RepAgent 与 Replication Server 断开连接，并禁止 RepAgent 进行连接。

语法

```
suspend log transfer
 from {data_server.database | all}
```

参数

- **data_server** – 数据服务器，其中包含要挂起 RepAgent 的数据库。
- **database** – 要挂起 RepAgent 的数据库，或禁止进行连接的数据库。
- **all** – 指示 Replication Server 挂起所有 RepAgent，并禁止以后连接所有 RepAgent。

示例

- **示例 1** – 将 *pubs2* 数据库的 RepAgent 断开连接，并禁止其重新进行连接：

  ```
  suspend log transfer from TOKYO_DS.pubs2
  ```

- **示例 2** – 将连接的所有 RepAgent 断开连接，并禁止任何 RepAgent 重新连接到 Replication Server：

  ```
  suspend log transfer from all
  ```

用法

- 使用 **suspend log transfer** 与 RepAgent 断开连接。这是停顿复制系统的第一步。**suspend log transfer** 不会关闭 RepAgent。
- 要测试挂起 RepAgent 后系统是否停顿，请使用 **admin quiesce_check**。
- 要允许 RepAgent 连接到 Replication Server，请执行 **resume log transfer**。

权限

suspend log transfer 需要“sa”权限。

另请参见

- admin quiesce_check （第 57 页）
- admin quiesce_force_rsi （第 58 页）
- resume log transfer （第 324 页）

suspend route

挂起到另一个 Replication Server 的路由。

语法

```
suspend route to dest_replication_server
 [with primary at dataserver.database]
```

参数

- **dest_replication_server** – 目标 Replication Server 的名称，将挂起到此目标 Replication Server 的路由。
- **with primary** – 指定您要从中挂起专用路由的主数据库连接。

示例

- **示例 1** – 挂起到 SYDNEY_RS Replication Server 的路由：

  ```
  suspend route to SYDNEY_RS
  ```

- **示例 2** – 要挂起 NY_DS.pdb1 主连接的 RS_NY 主 Replication Server 和 RS_LON 复制 Replication Server 之间的专用路由，请在 RS_NY 上输入：

  ```
  suspend route to RS_LON
   with primary at NY_DS.pdb1
   go
  ```

用法

- 使用 **suspend route** 挂起到另一个 Replication Server 的路由。利用此命令，可控制消息何时从一个 Replication Server 发送到另一个 Replication Server，从而管理网络的使用情况。
- 路由被挂起时，Replication Server 会将目标 Replication Server 的消息保存在稳定队列中。
- 您只能挂起直接路由。
- 要重新激活挂起的路由，请使用 **resume route**。

权限

suspend route 需要“sa”权限。

另请参见

- alter route （第 161 页）
- resume connection （第 321 页）
- resume route （第 326 页）

- suspend connection （第 334 页）

switch active

更改热备份应用程序中的活动数据库。

语法

```
switch active
 for logical_ds.logical_db
to data_server.database
[with suspension]
```

参数

- **logical_ds** – 逻辑连接的逻辑数据服务器名称。
- **logical_db** – 逻辑连接的逻辑数据库名称。
- **data_server** – 逻辑连接的新活动数据库的数据服务器名称。
- **database** – 逻辑连接的新活动数据库的数据库名称。
- **with suspension** – 完成切换后，挂起到新活动数据库的 DSI 连接。

示例

- **示例 1** – 此命令启动 switch active 进程：

```
switch active for LDS.pubs2 to OSAKA.pubs2
```

```
Switch of the active for this logical database is in progress.
```

用法

- **switch active** 是在热备份应用程序中切换到备用数据库的过程的一部分。有关完整过程，请参见《Replication Server 管理指南第二卷》。
- **switch active** 将立即返回结果，但在 **admin logical_status** 的“State of Operation in Progress”显示为“None”之前，此切换实际上没有完成。
- 使用 **admin logical_status** 可以监控 switch active 进程的状态。
- 如果使用 **with suspension** 选项，必须在切换完成后手动恢复与新活动数据库的 DSI 连接。
- 输入 **switch active** 之后，可以尝试使用 **abort switch** 来取消此命令。

权限

switch active 需要“sa”权限。

另请参见

- abort switch （第 43 页）

- admin logical_status （第 54 页）
- create logical connection （第 252 页）
- wait for switch （第 393 页）

sysadmin apply_truncate_table

为特定表的所有现有预订打开或关闭“subscribe to truncate table”选项，以启用或禁用 **truncate table** 的复制。

语法

```
sysadmin apply_truncate_table, data_server,
 database, {table_owner | '' | ""}, table_name
{'on'| 'off'}
```

参数

- **data_server** – 复制数据服务器的名称。
- **database** – 此数据服务器管理的复制数据库的名称。
- **table_owner** – 标识复制表的所有者。如果未指定所有者，Replication Server 将所有者设置为“dbo”。
- **table_name** – 标识要为其打开或关闭现有预订的“subscribe to truncate table”选项的复制表。
- **on** – 打开现有预订的“subscribe to truncate table”选项。
- **off** – 关闭现有预订的“subscribe to truncate table”选项。

示例

- **示例 1** – 为对 *pubs2* 数据库中 *emily* 所拥有的 *publishers* 表的所有预订打开“subscribe to truncate table”：

```
sysadmin apply_truncate_table, SYDNEY_DS,
 pubs2, emily, publishers, 'on'
```

用法

- **sysadmin apply_truncate_table** 用于 Adaptive Server 11.5 或更高版本的数据库。
- 如果未在复制定义中指定复制表所有者，则应为表所有者名称输入 "（两个单引号字符）或 ""（两个双引号字符）。
- 对某一特定数据库的特定表的预订必须全部支持或全部不支持 **truncate table** 的复制。例如，如果 **sysadmin apply_truncate_table** 关闭，您就不能新建包含“subscribe to truncate table”选项的预订，除非您打开此表所有预订的 **sysadmin apply_truncate_table**。

有关设置新预订的“subscribe to truncate table”选项的详细信息，请参见 **create subscription** 或 **define subscription**。

- Replication Server 作为维护用户在复制数据库上执行 **truncate table**。“replication_role”是为维护用户授予的权限之一。如果撤消维护用户的“replication_role”权限，则无法复制 **truncate table**，除非
  - 维护用户已被授予“sa_role”，
  - 维护用户拥有此表，或
  - 维护用户的别名为“Database Owner”。
- 热备份数据库无须预订 **truncate table**；**truncate table** 命令的执行将被自动复制到备用数据库。使用 **alter logical connection** 命令可为备用数据库打开 **truncate table** 的复制。

权限

sysadmin apply_truncate_table 需要“sa”权限。

另请参见

- create subscription （第 280 页）
- define subscription （第 290 页）

sysadmin cdb

在向 Sybase IQ 的实时装载 (RTL) 复制和到 Adaptive Server 的高容量自适应复制 (HVAR) 中管理净更改数据库。

语法

要持有、检查和释放净更改数据库，请使用：

```
sysadmin cdb, q_number, q_type,{hold | hold_next | unhold}
```

注意： 如果数据服务器接口执行程序 (DSI/E) 线程当前正在处理事务，您必须先执行带有 **hold** 或 **hold next** 的 **sysadmin cdb**，然后才能使用 **sysadmin cdb** 显示净更改数据库信息。

要显示有关净更改数据库的所有信息，或仅显示有关特定跟踪表的信息，请使用：

```
sysadmin cdb,
       [q_number[,q_type][list[,[“table_owner.]table_name”] |
       [[dump_i | dump_d | dump_u | dump_nc],table_name] |dump_nc]]
```

参数

- **持有** – 指示 DSI/E 挂起当前的净更改数据库实例，以便可以对其进行检查。

- **hold_next** – 指示 DSI/E 提交已准备好提交的第一个事务，释放数据库实例，然后保留下一个事务。
- **unhold** – 指示 DSI/E 释放 DSI 当前保留的所有净更改数据库实例并恢复正常的 DSI/E 活动。
- **q_number** – 标识复制数据库的出站 DSI 稳定队列。检查 **admin who, sqm** 命令的输出以标识该队列号。
- **q_type** – 标识稳定队列类型，其中 0 表示出站队列，1 表示入站队列。缺省值为 0。如果不指定 *q_type*，则使用缺省值。
- **table_name** – 指定复制表的名称。
- **list** – 显示有关净更改数据库的信息。如果不指定表名，**list** 将显示通过 *q_number* 指定的出站 DSI 稳定队列的所有实例。指定表可以只显示该表的内容
- **dump_i** – 返回包含内存 *Insert_Table* 表中所有列和行的结果。
- **dump_u** – 返回包含内存 *Update_Table* 表中所有列和行的结果。
- **dump_d** – 返回包含内存 *Delete_Table* 表中所有列和行的结果。
- **dump_nc** – 返回包含要应用于复制表的不可编译命令的结果。对于插入，返回所有列。对于删除，只返回主键。对于更新，只返回主键和更新列。

示例

- **示例 1** – 在数据库完全填充后，指示 DSI/E 挂起净更改数据库以进行检查。如果 DSI/E 当前未处理事务，则持有命令将在下次创建并填充净更改数据库时生效。Replication Server 会在下次创建并填充净更改数据库之后且可以将净更改数据库的内容应用到复制数据库之前挂起 DSI/E。例如，若要挂起当前的净更改数据库，请输入：

  ```
  sysadmin cdb,101,hold
  ```

- **示例 2** – 列出活动的 DSI 执行程序线程及相应的状态，包括有关 Replication Server 当前正在处理的任何净更改数据库的信息：

  ```
  sysadmin cdb
  ```

  输出显示了两个数据服务器及各自数据库的 RTL 状态、活动 DSI 执行程序 (DSI/E) 线程的队列号和队列类型：

  ```
  DSName   DBName    Queue  QType  Compile  Hold  CdbName  Commands
  _in_Group
  -------  --------  -----  -----  -------  ----  -------  --------
  ---------
  IQSRVR2  asiqdemo  105    0      On       No             0
  IQSRVR   iqdemo    104    0      On       No             0
  ```

  状态列为：

  - Compile – 如果 RTL 处于活动状态，则状态为“On”
  - Hold – 如果您为相同的 *q_number* 和 *q_type* 执行了带有 **hold** 的 **sysadmin cdb** 以持有特定 DSI/E，则状态为“Yes”

  - CdbName – Replication Server 当前正在处理的净更改数据库的内部名称，即在该 DSI/E 线程上处于“hold”状态。在本例中，Replication Server 当前未处理任何净更改数据库。
  - Commands_in_Group – Replication Server 作为一组进行编译的命令数。在本例中，未处理任何命令。
- **示例 3** – 在列出有关特定 DSI/E 线程的信息之前，您无需通过将 DSI/E 设置为 **hold** 或 **hold_next** 状态而将其挂起。由于 DSI/E 未处于 **hold** 或 **hold_next** 状态，连续执行该命令时，任何值都可能变化，但 Queue 和 QType 列下面的值除外：

```
sysadmin cdb,107,1
```

输出：

```
Queue  QType  CdbName              TargetDB     Compilable_Tables
-----  -----  -----------------    -----------  -----------------
107    1      asiqdemo_ws_46_3     asiqdemo_ws  1

Non_Compilable_Tables  Commands_in_Group  Compiled_Rows  Non_Comp
ilable
                                                        Commands
---------------------  -----------------  -------------  -------
0                      3                  2              0
```

- **示例 4** – 显示有关 DSI/E 当前正在运行的净更改数据库的信息：

注意： 在列出有关 DSI/E 当前正在运行净更改数据库的信息之前，您必需挂起具有“**hold**”状态的数据库。

```
sysadmin cdb,107,1,hold
go
sysadmin cdb,107,1,list
go
```

输出为：

```
CdbName            Replicate_Table          Status       Cmd_Convert
-----------------   ----------------------   ----------
asiqdemo_ws_46_3   dbo.test_alltypes_ws_1   compilable   i2di

AutoCorrection   Nb_Columns   PK_Cols   CdbTable
--------------   ----------   -------   --------------------
No               25           22        test_allpes_ws_1_46_1

Insert_Table                Inserts   Update_Table
Updates
-------------------------   -------   ------------
rs_itest_allpes_ws_1_46_1   1         rs_utest_allpes_ws_1_46_1  0

Delete_Table                Deletes   Non_Compilable_Cmds
-------------------------   -------   -------------------
rs_dtest_allpes_ws_1_46_1   1         0

Update_Worktable             Delete_Worktable
--------------------------   ----------------
#rs_dtest_allpes_ws_1_46_1
```

```
Reduced_Inserts   Reduced_Updates   Reduced_Deletes
---------------   ---------------   ---------------
0                 0                 0
(1 rows affected)
```

报告中有如下列：

- CdbName – Replication Server 当前正在处理的净更改数据库的内部名称，即在该 DSI/E 线程上处于“hold”状态。
- Replicate_Table – 复制表名称
- Status – “可编译”或“不可编译”表
- Cmd_Convert – 应用的命令转换，例如 **none**、**ud2i**、**i2di** 或 **i2none**
- AutoCorrection – 是否应用自动更正
- Nb_Columns – 净更改数据库表中的列数
- PK_Cols – 净更改数据库表中的主键列数
- CdbTable – 净更改数据库表的唯一名称
- Insert_Table – 净更改数据库中插入操作的内存表名称
- Inserts – 插入数量
- Update_Table – 净更改数据库中更新操作的内存表名称
- Updates – 更新数量
- Delete_Table – 净更改数据库中删除操作的内存表名称
- Deletes – 删除数量
- Non_Compilable_Cmds – 不可编译命令的数量。
- Update_Worktable – 应用更新时在复制数据服务器上创建的工作表的名称。此工作表将被填充并与复制表连接
- Delete_Worktable – 应用删除时在复制数据服务器上创建的工作表的名称。此工作表将被填充并与复制表连接
- Reduced_Inserts – 因编译而减少的插入数量
- Reduced_Updates – 因编译而减少的更新数量
- Reduced_Deletes – 因编译而减少的删除数量

- **示例 5** – 通过在查询中包括 **dump_i**、**dump_u**、**dump_d** 或 **dump_nc** 选项以返回表中的信息，您可以列出有关净更改数据库中特定表的详细信息。这些选项是在净更改表上执行的 SQL **select** 语句。

  例如，要显示内存表中 *dbo.test_alltypes_msa_1* 和 *Insert_Table* 的内容，请输入：

  ```
  sysadmin cdb,106,0,dump_i,dbo.test_alltypes_msa_1
  ```

  如果复制成功，将会显示下面的输出：

  ```
  c1          c2          c3
  ---------   ---------   ---------
  4           v           ddd
  3           upd         qqq
  ```

```
(2 rows affected)
```

- **示例 6** – 要显示所有不可编译的命令，请输入：

```
sysadmin cdb,105,1,dump_nc
```

输出为：

```
Cmd      Text
------   -----------------------------------------------------
1        insert into dbo.test_alltypes_msa_1 (c1, c2, c3, c4, c5,
c6, c7,
        c8, c9, c10, c11, c12, c13, c14, c15, c16, c17, c18, c19,
c20, c21,
        c22, c23, c24, c25) values (2, 1, 1, 'abcdefg', 'fghijkl',
         ‘20091125 08:50:42:100’, ‘20091125 08:51:00:000’,
1001.0000, 10,
         10, 0.555544443333222211110000111122223333, 0x01234567,
         ‘20091125’, 1, 254.0000, 4967295, 65500, 92233720,
922337203,
         ‘08:50:42:113’, ‘mnopqrst’,
‘abcdefghijklmnopqrtsuvwxyz
       jjjjjzzzzzzzzjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjz
0
         ddddddddddddddddddddddddddddddddddddddddddddddddddddddddd
ddddddddd
         ddddddddddddddddddddd e
         ddddddddddddddddddddddddddddddddddddddddddddddddddddddddd
ddddddddd
         ddddddddddddddddddddd f’, ‘uvwxy’,
         convert(varchar,
0x6100640064006400640064006400640064006400640064
         0064006400640064006400640064006400640064006400640064006400640064
00640064
         0064006400640064006400640064006400640064006400640064006400640064
006400640
         0640064006400640064006400640064006400640064006400640064006400640
064006400
         6400640064006400640064006400640064006400640064006400640064006400
640064006
         4006400640064006400640064006400640020006500200064006400640064006
40064006
         4006400640064006400640064006400640064006400640064006400640064006
40064006
         4006400640064006400640064006400640064006400640064006400640064006
40064006
         4006400640064006400640064006400640064006400640064006400640064006
40064006
         4006400640064006400640064006400640020006600), 0x48656c6c6f20)
2        update article set price = price*1.015 where id = 108
3        delete from dbo.test_alltypes_msa_1 where c1=1 and c2=1
and c3=1
         and c4=‘ABCDEFG’ and c5=‘FGHIJKL’ and c6=‘20091125
8:50:42:100’ and
        c7=‘20091125 08:51:00:000’ and c8=1001.0000 and c9=10 and
c10=10
```

```
         and c11=0.555544443333222211110000111122223333 and
c12=0x01234567
         and c13= '20091125' and c14=1 and c15=254.0000 and
c16=4967295 and
         c17=65500 and c18=92233720 and c19=922337203 and
         c20='08:50:42:113' and c21= 'MNOPQRST' and
c23='UVWXY'
```

- **示例 7** – 要显示有关净更改数据库中特定表的详细信息，请输入：

```
sysadmin cdb,107,1,hold
go
sysadmin cdb,107,1,list,test_alltypes_ws_1
go
```

输出中显示的信息包括以下内容：

1. 操作状态和内存表的名称：

```
CdbName
           Replicate_Table          Status       Cmd_Convert
----------------   ----------------------   ----------   -----
------
asiqdemo_ws_46_3   dbo.test_alltypes_ws_1   compilable   i2di

AutoCorrection   Nb_Columns   PK_Cols   CdbTable
--------------   ----------   -------   --------------------
No               25           22        test_allpes_ws_1_46_1

Insert_Table                Inserts  Update_Table
  Updates
-------------------------   -------  -------------------------
  -------
rs_itest_allpes_ws_1_46_1   1        rs_utest_allpes_ws_1_46_1
  0

Delete_Table                Deletes   Non_Compilable_Cmds
-------------------------   -------   -------------------
rs_dtest_allpes_ws_1_46_1   1         0

Update_Worktable             Delete_Worktable
--------------------------   ----------------
#rs_dtest_allpes_ws_1_46_1

Reduced_Inserts   Reduced_Updates   Reduced_Deletes
---------------   ---------------   ---------------
0                 0                 0
(1 row affected)
```

2. 有关表中所有列的信息：

```
Colname Coltype Maxlength Cdbtype Cdbvtype Primary_key Changed
 HasNull
------- ------- --------- ------- -------- ----------- -------
 -------
c1      int     4         8       8        1           1       0
...
c8      money   10        1       0        1           1       0
```

```
...c25     image   5         5       19       0          1       1
(25 rows affected)
```

用法

通过在查询中包括这些 SQL 命令之一，您可以列出有关净更改数据库中特定内存表的详细信息。内存表用于内部处理，其内容不会保存在磁盘上。

您必须先执行 **sysadmin net_change_db hold** 或 **sysadmin net_change_db hold next**，然后才能使用 **sysadmin net_change_db list** 来显示净更改数据库信息。

权限

sysadmin net_change_db 需要“sa”权限。

另请参见

- admin who （第 88 页）
- admin config （第 47 页）

sysadmin dropdb

从 ID Server 中删除数据库。

语法

```
sysadmin dropdb, data_server, database
```

参数

- **data_server** – 数据服务器名称。
- **数据库** – 要删除的数据库的名称。

示例

- **示例 1** – 从 ID Server 中删除 SYDNEY_DS 数据服务器中的 *pubs2* 数据库：

  ```
  sysadmin dropdb, SYDNEY_DS, pubs2
  ```

用法

- 使用 **sysadmin dropdb** 从 ID Server 中删除数据库。此命令必须在 ID Server 上执行。
- 只有在 ID Server 系统表包含有关此系统中不存在的数据库的信息时，才使用 **sysadmin dropdb**。这种情况只在系统出现故障后发生。

  例如，用 **drop connection** 删除某一数据库后，可能会因为网络故障，ID Server 没有接到通知，因而未从它的表中删除此数据库。如果此后尝试将同一数据服务器

和数据库添加到此系统，请求将会失败，这是因为此数据库及其数据服务器已经在 ID Server 系统表中注册。
- 如果重新安装某 Replication Server，请使用 **sysadmin dropdb** 删除此 Replication Server 所管理的每个数据库（包括其 RSSD）的 ID Server 信息。否则，重新安装 Replication Server 时会出错。
- 如果使用此命令时输入的参数无效，您不会得到通知。

警告！ 切勿对有活动连接的任何数据库使用 **sysadmin dropdb**。

权限

sysadmin dropdb 需要“sa”权限。

另请参见

- sysadmin dropldb （第 347 页）

sysadmin dropldb

从 ID Server 中删除逻辑数据库。

语法

```
sysadmin dropldb, data_server, database
```

参数

- **data_server** – 逻辑数据服务器名称。
- **数据库** – 要删除的逻辑数据库的名称。

示例

- **示例 1** – 从 ID Server 删除 LDS 逻辑数据服务器中的 *pubs2* 逻辑数据库：

  ```
  sysadmin dropldb, LDS, pubs2
  ```

用法

- 使用 **sysadmin dropldb** 从 ID Server 删除逻辑数据库。此命令必须在 ID Server 上执行。
- 只有在 ID Server 系统表中包含有关此系统中不存在的逻辑数据库的信息时，才使用 **sysadmin dropldb**。这种情况只在系统出现故障后发生。

  例如，用 **drop logical connection** 删除某逻辑数据库后，可能因为网络故障，ID Server 没有接到通知，因而未从它的表中删除此逻辑数据库。如果此后尝试将同

样的逻辑数据服务器和逻辑数据库添加到此系统中，请求将会失败，这是因为此逻辑数据库及其逻辑数据服务器已经在 ID Server 系统表中注册。
- 如果您重新安装 Replication Server，首先要用 **sysadmin dropldb** 删除此 Replication Server 所管理的每个逻辑数据库的 ID Server 信息。否则，重新安装 Replication Server 时会出错。
- 如果使用此命令时输入的参数无效，您不会得到通知。

警告！ 切勿对有活动连接的任何逻辑数据库使用 **sysadmin dropldb**。

权限

sysadmin dropldb 需要“sa”权限。

另请参见

- sysadmin dropdb （第 346 页）

sysadmin drop_queue

删除稳定队列。此命令用于删除失败的实现队列。

语法

```
sysadmin drop_queue, q_number, q_type
```

参数

- **q_number** – 作为此队列的源或目标的 Replication Server 或数据库的节点 ID。
- **q_type** – 队列类型。

用法

- 使用 **sysadmin drop_queue** 可以停止并删除在发生不可恢复的预订错误以后仍然存在，必须手动清除的那些实现队列。

  警告！ sysadmin drop_queue 仅用于删除失败的实现队列。
- 使用 **admin who** 可以查找队列的 *q_number* 和 *q_type*。这些值出现在命令的 SQM 线程输出中。

权限

sysadmin drop_queue 需要“sa”权限。

另请参见

- rebuild queues （第 320 页）
- sysadmin purge_route_at_replicate （第 369 页）

sysadmin droprs

从 ID Server 中删除 Replication Server。

语法

```
sysadmin droprs, replication_server
```

参数

- **replication_server** – 要删除的 Replication Server 的名称。

示例

- **示例 1** – 从 ID Server 中删除 SYDNEY_RS Replication Server：

```
sysadmin droprs, SYDNEY_RS
```

用法

- 使用 **sysadmin droprs** 从 ID Server 删除 Replication Server。可以在 ID Server 上执行此命令。
- 只有在 ID Server 包含的 Replication Server 相关信息在复制系统中不存在时，才能使用 **sysadmin droprs**。这种情况通常是由系统故障造成的。例如，如果 Replication Server 安装失败，则 ID Server 系统表可能会包含此 Replication Server 的一些条目，这将妨碍此后安装 Replication Server。
- 如果输入的参数无效，您不会得到通知。

警告！ 删除活动 Replication Server 时，应谨慎使用 **sysadmin droprs**。有关用于删除活动 Replication Server 的正确过程，请参见《Replication Server 管理指南第一卷》。

权限

sysadmin droprs 需要“sa”权限。

sysadmin dump_file

指定转储 Replication Server 稳定队列时使用的备用日志文件名。

语法

```
sysadmin dump_file [, file_name]
```

参数

- **file_name** – 要将稳定队列转储写入其中的新日志文件的名称。

示例

- **示例 1** – 指定 *pubs2.log* 作为记录稳定队列输出的文件：

```
sysadmin dump_file, 'pubs2.log'
```

用法

- 使用 **sysadmin dump_queue** 将日志转储到文件之前，使用 **sysadmin dump_file** 指定日志文件名称。
- 要将当前的转储文件重置为缺省转储文件，请执行 **sysadmin dump_file**（不指定文件名）。
- 如果指定了文件名，将关闭当前转储文件，并打开新文件。新文件使用指定的文件名。
- 缺省的转储文件为 Replication Server 日志。使用 **admin log_name** 可显示此文件的路径。
- 如果输入的日志文件名包含字母和数字以外的字符，要用引号将此文件名引起来。

权限

sysadmin dump_file 需要“sa”权限。

另请参见

- admin log_name （第 54 页）
- sysadmin dump_queue （第 350 页）
- sysadmin sqt_dump_queue （第 382 页）

sysadmin dump_queue

转储 Replication Server 稳定队列的内容。

语法

```
sysadmin dump_queue {, q_number | server[,database]}, qtype
{
        , seg, blk, cnt
        [, num_cmds]
        [, {L0 | L1 | L2 | L3}]
        [, {RSSD | client | “log” | file_name}]
        |
        “next” [, num_cmds]
}
```

参数

- **q_number | server[, database]** – 标识要转储的稳定队列。使用 *q_number* 或 *server*[, *database*] 可指定队列号。使用 **admin who**、**admin who, sqm** 和 **admin who, sqt** 可以标识队列号。
- **q_type** – 稳定队列的队列类型。值为“0”表示出站队列；值为“1”表示入站队列。使用 **admin who**、**admin who, sqm** 和 **admin who, sqt** 可以标识队列类型。
- **seg** – 标识起始段。
- **blk** – 标识要开始转储的段中的 16K 块。块的编号是从 1 到 64。

  sysadmin dump_queue 可识别 *seg* 和 *blk* 的 4 种特殊设置：

  - 将 *seg* 设置为 -1，表示从队列中第一个活动段开始。
  - 将 *seg* 设置为 -2，表示从队列中第一个段开始，包括由于设置保存间隔而保留的所有非活动段。
  - 将 *seg* 设置为 -1 并将 *blk* 设置为 -1，表示从队列中第一个取消删除的块开始。
  - 将 *seg* 设置为 -1 并将 *blk* 设置为 -2，表示从队列中第一个未读取的块开始。
- **cnt** – 指定要转储的块数。这个数字可以跨多个段。如果将 *cnt* 设置为 -1，则转储的最后一个块便是当前段的末尾。如果将它设置为 -2，则转储的最后一个块便是此队列的末尾。
- **num_cmds** – 指定要转储的命令数。此数字会覆盖 *cnt*。如果 *num_cmds* 设置为 –1，则当前段的结尾是所转储的最后一个命令。如果 *num_cmds* 设置为 -2，则队列的结尾是所转储的最后一个命令。
- **L0** – 转储稳定队列的所有内容。如果未指定 **L0**、**L1**、**L2** 或 **L3**，则这是缺省行为。
- **L1** – 仅转储稳定队列中的事务的 **begin** 和 **end** 命令。
- **L2** – 将稳定队列事务的 **begin** 和 **end** 命令与事务中所有其它命令的前 100 个字符一起转储。
- **L3** – 转储稳定队列的所有内容。除 **SQL** 语句外，所有其它命令均以注释形式显示。仅当使用 *file_name* 选项或 **sysadmin dump_file** 命令指定备用日志文件时，才可以使用 **L3**。不能将 **L3** 用于 **RSSD** 或 **client** 选项。
- **RSSD** – 强制输出到 RSSD 中的系统表。
- **客户端** – 强制输出到发出此命令的客户端。
- **"log"** – 强制输出到 Replication Server 日志文件。
- **file_name** – 强制输出到 *file_name* 日志文件中。也可以使用 **sysadmin dump_file** 命令设置备用日志文件。此文件的位置记录在 Replication Server 日志中。
- **"next"[, num_cmds]** – 从上次为特定队列和会话运行 **sysadmin dump_queue** 时的停止位置开始，并且转储的命令或块的数量与上次运行所处理的命令或块的数量相同。使用 *num_cmds* 可以覆盖以前的 *cnt* 或 *num_cmds* 的值。

  如果在以前未调用 **sysadmin dump_queue** 的情况下使用 **"next"[, num_cmds]**，转储将从队列的开头（缺省值为 seg -1、blk -1 和 cnt -2）开始并将 *num_cmds* 作为命令数处理。

示例

- **示例 1** – 处理队列 103:1 时，将段 0 的块 15 – 64 和段 1 的块 1 – 15 转储到 Replication Server 日志中：

```
sysadmin dump_queue, 103, 1, 0, 15, 65
```

- **示例 2** – 将队列 103:1 的所有内容转储到 RSSD 中：

```
sysadmin dump_queue, 103, 1, -1, 1, -2, RSSD
```

- **示例 3** – 将队列 103:1 的内容转储到 `SYDNEY_RS.log` 日志文件中。最后一个 **sysadmin dump_file** 命令会关闭 `SYDNEY_RS.log`，并且所有后续转储均将定向到 Replication Server 日志：

```
sysadmin dump_file, SYDNEY_RS.log
sysadmin dump_queue, 103, 1, -1, 1, -2
sysadmin dump_file
```

- **示例 4** – 将 SYDNEY_DS.*pubs2* 的入站队列的内容转储到 Replication Server 日志中：

```
sysadmin dump_queue, SYDNEY_DS, pubs2, 1, -1, 1,
     -2, 10, "log"
```

- **示例 5** – 将队列 103:1 的 10 个命令转储到 Replication Server 日志中：

```
sysadmin dump_queue, 103, 1, -1, 1, -2, 10, "log"
```

- **示例 6** – 仅将队列 103:1 的 **begin** 和 **end** 命令转储到 Replication Server 日志中：

```
sysadmin dump_queue, 103, 1, -1, 1, -2, L1
```

- **示例 7** – 将队列 103:1 的内容转储到 Replication Server 日志中：

```
sysadmin dump_queue, 103, 1, -1, 1, -2, "next"
```

- **示例 8** – 以大块的形式将队列 103:1 的内容转储到 Replication Server 日志中。**"next"** 从上次运行 **sysadmin dump_queue** 时的停止位置开始转储队列。在此示例中，第一次调用 **sysadmin dump_queue** 时将转储前 10 个命令，第二次调用转储接下来 10 个命令，最后一次调用转储再接下来的 20 个命令：

```
sysadmin dump_queue, 103, 1, -1, 1, -2, 10
sysadmin dump_queue, 103, 1, "next"
sysadmin dump_queue, 103, 1, "next", 20
```

用法

- 使用 **sysadmin dump_queue** 转储 Replication Server 稳定队列的内容。
- **sysadmin dump_queue** 将稳定队列转储到以下位置之一：
  - Replication Server 日志
  - 备用日志文件
  - RSSD
  - 发出此命令的客户端

  要将队列转储到 RSSD 或客户端，**sysadmin dump_queue** 的最后一个参数必须是 **RSSD** 或 **client**。

如果未指定 **RSSD** 或 **client** 选项，或者指定了“**log**”选项，则会输出到 Replication Server 日志中。

如果通过 **sysadmin dump_file** 命令或者 *file_name* 选项指定用于转储队列的备用日志文件，则输出到此备用转储文件中。

- 通过设置 **queue_dump_buffer_size** 配置参数来指定 **sysadmin dump_queue** 命令的最大长度。

转储到 RSSD

如果使用 **RSSD** 选项，则会将转储写入到 RSSD 中的两个系统表：*rs_queuemsg* 和 *rs_queuemsgtxt*。

如果队列转储到 RSSD，则首先清除这两个系统表中的那些与要转储的块有相同的 *q_number*、*q_type*、*seg* 和 *blk* 的段。

有关 *rs_queuemsg* 系统表内容的信息，请参见“Replication Server 系统表”。

rs_queuemsgtxt 系统表包含从稳定队列转储的命令的文本。如果一个命令的文本超过 255 个字符，将存储在多行中，这些行用 *q_seq* 列进行编号。

转储到客户端

如果使用 **client** 选项，此转储将被写入发出此命令的客户端，如 **isql** 或 Replication Server Manager。

权限

sysadmin dump_queue 需要“sa”权限。

另请参见

- admin who （第 88 页）
- rs_queuemsg （第 607 页）
- rs_queuemsgtxt （第 608 页）
- sysadmin dump_file （第 349 页）

sysadmin dump_thread_stacks

转储 Replication Server 堆栈。

语法

```
sysadmin dump_thread_stacks [, module_name]
```

参数

- **module_name** – Replication Server 线程的类型。有效的模块名称与 **admin who** 命令显示的 *name* 列下面的名称相同。

示例

- **示例 1** – RSI 转储 RSI 队列堆栈：

```
sysadmin dump_thread_stacks, RSI
```

```
T. 2006/10/23 15:37:39. (259): RS Thread Type  = 'RSI'
T. 2006/10/23 15:37:39. (259): RS Thread State =
   'Awaiting Wakeup'
T. 2006/10/23 15:37:39. (259): RS Thread Info  =
   'ost_columbia_02'
T. 2006/10/23 15:37:39. (259): Open Server Process ID:
   50, SRV_PROC address 0xed79c8
T. 2006/10/23 15:37:39. (259): Start of stack trace for
   spid 50.
T. 2006/10/23 15:37:39. (259): Native thread #70,
   FramePointer: 0xfe34f050
T. 2006/10/23 15:37:39. (259): 0x00362fc8
   sqm_read_message     (0x3345ed0, 0xfe34fdf4, 0xea60,
   0x0, 0xfe34fdf0, 0x47105f0) +0x48
T. 2006/10/23 15:37:39. (259): 0x00300908
   _rsi_sender_wrapper  (0x30c390, 0x30c230, 0x476f1f0,
   0x47105f0, 0x1f2, 0x47105f0) +0x2f28
T. 2006/10/23 15:37:39. (259): 0x002fe960
   _rsi_sender_wrapper  (0x1d794f0, 0xffffd8f1,
   0x268d14, 0xffffd800, 0x800, 0x0) +0xf80
T. 2006/10/23 15:37:39. (259): 0x0054dabc
   srv__start_function  (0xed79c8, 0x0, 0x800,
   0x862a04, 0x0, 0x0) +0x1c0
T. 2006/10/23 15:37:39. (259): 0xff265d48 _resume_ret
   (0x0, 0x0, 0x0, 0x0, 0x0, 0x0) +0x2d0
T. 2006/10/23 15:37:39. (259): End of stack trace for
   spid 50.
T. 2006/10/23 15:37:39. (259):
```

用法

- 在 Replication Server 速度异常缓慢时，使用 **sysadmin dump_thread_stacks** 检查 Replication Server 内部进程。
- **sysadmin dump_thread_stack** 不适用于以下平台：
  - Sun Solaris
  - HPUX
  - Linux
  - IBM

请参见《Open Server Server-Library/C 参考手册》中的 **srv_dbg_stack()**。

权限

sysadmin dump_thread_stacks 需要“sa”权限。

sysadmin dump_tran

将特定稳定队列事务的语句转储到日志文件中。

语法

```
sysadmin dump_tran {{, q_number, | server [,database]},
        q_type, lqid
        [, num_cmds]
        [, {L0 | L1 | L2 | L3}]
        [, {RSSD | client | “log” | file_name}] |
         “next” [, num_cmds]}
```

参数

- **q_number | server[, database]** – 标识稳定队列。使用 *q_number* 或 *server*[, *database*] 可指定队列号。使用 **admin who**、**admin who, sqm** 和 **admin who, sqt** 可以标识队列号。
- **q_type** – 稳定队列的队列类型。值为“0”表示出站队列；值为“1”表示入站队列。使用 **admin who**、**admin who, sqm** 和 **admin who, sqt** 可以标识队列类型。
- **lqid** – 稳定队列事务的任何命令的本地队列 ID。*lqid* 标识要转储的事务。格式：*seg,blk,row*。
- **num_cmds** – 指定要转储的命令数。
- **L0** – 转储指定事务的内容。如果未指定 **L0**、**L1**、**L2** 或 **L3**，则这是缺省行为。
- **L1** – 仅转储指定事务的 **begin** 和 **end** 命令。
- **L2** – 将指定事务的 **begin** 和 **end** 命令与事务中其它命令的前 100 个字符一起转储。
- **L3** – 转储指定事务的所有命令。除了 **SQL** 语句以外，所有其它命令均以注释形式显示。仅当使用 *file_name* 选项或 **sysadmin dump_file** 命令指定备用日志文件时，才可以使用 **L3**。不能将 **L3** 用于 **RSSD** 或 **client** 选项。
- **RSSD** – 强制输出到 RSSD 中的系统表。
- **客户端** – 强制输出到发出此命令的客户端。
- **"log"** – 强制输出到 Replication Server 日志文件。
- **file_name** – 强制输出到 *file_name* 日志文件中。您可以使用 **sysadmin dump_file** 命令设置备用日志文件。
- **"next"[, num_cmds]** – 此选项会继续 **sysadmin dump_tran** 的上次运行。 **“next"[, num_cmds]** 从上次为特定事务运行 **sysadmin dump_tran** 时的停止位置开始，并且转储的命令的数量与上次运行所处理的命令的数量相同。使用 *num_cmds* 可以覆盖以前的 *cnt* 或 *num_cmds* 的值。

如果以前未调用过 **sysadmin dump_tran**，则不能使用 **"next"[, num_cmds]**。

示例

- **示例 1** – 将队列 103:1 中 LQID 为 0:15:2 的事务转储到 Replication Server 日志中：

```
sysadmin dump_tran, 103, 1, 0, 15, 2
```

- **示例 2** – 将 *SYDNEY_DS.pubs2* 的入站队列中 LQID 为 0:15:2 的事务的 10 个命令转储到 Replication Server 日志中：

```
sysadmin dump_tran, SYDNEY_DS, pubs2, 1, 0, 15, 2,
     10, “log”
```

- **示例 3** – 仅将队列 103:1 中 LQID 为 0:15:2 的事务的 **begin** 和 **end** 命令转储到 Replication Server 日志中：

```
sysadmin dump_tran, 103,1, 0, 15, 2, L1
```

- **示例 4** – 将队列 103:1 中 LQID 为 0:15:2 的事务的所有命令转储到 Replication Server 日志中。所有命令均被截断为 100 个字符：

```
sysadmin dump_tran, 103,1, 0, 15, 2, L2
```

- **示例 5** – 将队列 103:1 中 LQID 为 0:15:2 的事务转储到 `SYDNEY_RS.log` 文件中：

```
sysadmin dump_tran, 103,1, 0, 15, 2, L3, SYDNEY_RS.log
```

- **示例 6** – 将队列 103:1 中 LQID 为 0:15:2 的事务转储到 RSSD 中：

```
sysadmin dump_tran, 103, 1, 0, 15, 2, RSSD
```

- **示例 7** – 将队列 103:1 中 LQID 为 0:15:2 的事务转储到客户端中：

```
sysadmin dump_tran, 103, 1, 0, 15, 2, client
```

- **示例 8** – 以大块的形式将队列 103:1 中 LQID 为 0:15:2 的事务转储到 Replication Server 日志中。**"next"** 从上次运行 **sysadmin dump_tran** 时的停止位置开始转储事务。在此示例中，第一次调用 **sysadmin dump_tran** 将转储事务的前 10 个命令，第二次调用将转储接下来的 10 个命令，最后一次调用将转储再接下来的 20 个命令：

```
sysadmin dump_tran, 103,1, 0, 15, 2, 10
sysadmin dump_tran,  “next”
sysadmin dump_tran,  “next”, 20
```

用法

- 可使用 **sysadmin dump_tran** 转储由 LQID 标识的稳定队列事务的内容。
- **sysadmin dump_tran** 将输出到下列位置之一：
  - Replication Server 日志
  - 备用日志文件
  - RSSD
  - 发出此命令的客户端

要将稳定队列事务转储到 RSSD 或客户端中，**sysadmin dump_tran** 的最后一个参数必须是 **RSSD** 或 **client**。

如果未指定 **RSSD** 或 **client** 选项，或者指定了 **log** 选项，则会输出到 Replication Server 日志中。

如果通过 **sysadmin dump_file** 命令或者 *file_name* 选项指定用于转储稳定队列事务的备用日志文件，则输出到此备用转储文件中。

- 通过设置 **queue_dump_buffer_size** 配置参数来指定 **sysadmin dump_tran** 命令的最大长度。

转储到 RSSD

如果使用 **RSSD** 选项，则会将转储写入到 RSSD 中的两个系统表：*rs_queuemsg* 和 *rs_queuemsgtxt*。

如果事务转储到 RSSD，则首先清除这两个系统表中与要转储的事务具有相同的 *q_number*、*q_type*、*seg* 和 *blk* 的那些段。

有关 *rs_queuemsg* 系统表内容的信息，请参见“Replication Server 系统表”。

rs_queuemsgtxt 系统表包含从稳定队列转储的命令的文本。如果一个命令的文本超过 255 个字符，将存储在多行中，这些行用 *q_seq* 列进行编号。

转储到客户端

如果使用 **client** 选项，此转储将被写入发出此命令的客户端，如 **isql** 或 Replication Server Manager。

权限

sysadmin dump_tran 需要“sa”权限。

另请参见

- admin who （第 88 页）
- rs_queuemsg （第 607 页）
- rs_queuemsgtxt （第 608 页）
- sysadmin dump_file （第 349 页）

sysadmin erssd

用于检查 ERSSD 文件位置和备份配置，或者执行非预定的 ERSSD 备份。

此命令返回 ERSSD 的状态，其中包括：

- ERSSD 名称
- 数据库文件位置
- 事务日志文件位置

- 事务镜像位置
- 备份开始时间、开始日期和间隔
- 备份目录位置

语法

```
sysadmin erssd [, backup | dbfile_dir, 'path' |
translog_dir, 'path' |
logmirror_dir, 'path' | defrag]
```

参数

- **backup** – 执行 ERSSD 的单个非预定备份。
- **dbfile_dir, 'path'** – 指定 ERSSD 数据库文件的新目录。
- **translog_dir, 'path'** – 指定事务日志文件的新目录。
- **logmirror_dir, 'path'** – 指定事务日志镜像文件的新目录。
- **defrag** – 重建 ERSSD 数据库，不包含空段。
- **path** – 新目录的路径名。

注意： 应谨慎使用这些目录路径更改选项。带有这些选项执行 **sysadmin erssd** 时，将自动重新启动 ERSSD，并且可能会导致系统中断。

示例

- **示例 1** – 以下示例显示了 **sysadmin erssd** 输出：

```
sysadmin erssd
----------------------

ERSSD Name  ERSSD Database File   ERSSD Transaction Log
----------  --------------------  ------------------
erssd.db    /dbfile/erssd.db      /log/erssd.log


ERSSD Transaction Log Mirror ERSSD Backup Start Time
---------------------------- ------------------------
/backup/erssd.mlg            2am

ERSSD Backup Start Date   ERSSD Backup Interval
-----------------------   ----------------------
March 20, 2003            12 hours

ERSSD Backup Location
---------------------
/backup
```

用法

- 使用此命令时不带任何选项可显示数据库文件路径、事务日志路径、事务日志镜像路径以及预定事务的开始时间、开始日期和位置。
- 使用此命令时带有 **backup** 选项，可执行一个非预定备份。
- 使用此命令时带有选项 **dbfile_dir**，可关闭 ERSSD，将数据库移动到新目录，更新 Replication Server 配置文件并重新启动 ERSSD，从而在新位置使用数据库。
- 使用此命令时带有选项 **translog_dir**，可关闭 ERSSD，将事务日志文件移到新目录，更新 ERSSD 以使用新目录中的事务日志镜像，更新 Replication Server 配置文件并重新启动 ERSSD。
- 使用此命令时带有选项 **logmirror_dir**，可关闭 ERSSD，将事务日志镜像文件移到新目录，更新 ERSSD 以使用新目录中的事务日志镜像，更新 Replication Server 配置文件并重新启动 ERSSD。
- 使用此命令时带有选项 **defrag**，可关闭 ERSSD，重建数据库文件并重新启动 ERSSD。
- 使用此命令时带有选项 **defrag**、**dbfile_dir**、**translog_dir** 和 **logmirror_dir** 会占用较多的资源。在此操作期间，ERSSD 将无法使用，所有尝试访问 ERSSD 的线程均会失败。重新启动 ERSSD 之前，这些线程将一直处于阻塞状态。
- 要使用 **defrag**，节点版本必须为 15.0 或更高版本。此选项自动将经过碎片整理的文件升级到 SQL Anywhere 11.0，执行此命令后，无法将该文件降级。
- 在需要将文件移动到更大、更快速的磁盘时，可使用此命令。
- 用单引号（不是双引号）将 *path* 引起来。

权限

您必须具有“sa”特权才能执行此命令。

sysadmin fast_route_upgrade

将路由版本更新为主 Replication Server 或复制 Replication Server 版本较低者的节点版本。

升级路由将重新实现系统表中的数据，使新升级的 Replication Server 可以使用与新功能相关的信息。

注意： 只有在主 Replication Server 尚未使用要求实现的新功能的情况下，才能使用 **sysadmin fast-route-upgrade**。

语法

```
sysadmin fast_route_upgrade, dest_replication_server
```

参数

- **dest_replication_server** – 路由的目标 Replication Server。

示例

- **示例 1** – 在这些示例中，TOKYO_RS 的节点版本为 1200。SYDNEY_RS 刚从 11.5 升级到 12.0；它的节点版本是 1200。此命令在此路由的源 Replication Server (SYDNEY_RS) 上发出，路由目标为 Tokyo Replication Server (TOKYO_RS)，此命令将路由版本设置为 12.0。SYDNEY_RS 上尚未使用新功能：

```
sysadmin fast_route_upgrade, TOKYO_RS
```

- **示例 2** – 此命令在此路由的源 Replication Server (TOKYO_RS) 上发出，路由目标为 Sydney Replication Server (SYDNEY_RS)，由于 TOKYO_RS 上已经使用了新功能，因而会拒绝此命令，必须使用 Sybase Central 的 Replication Manager 插件升级此路由：

```
 sysadmin fast_route_upgrade, SYDNEY_RS
```

用法

- 只要路由两端的 Replication Server 已经升级，而且节点版本设置为 11.5 或更高版本，您就必须 升级连接这两个服务器的每一个路由，这样新功能才能通过路由得以应用。应在源 Replication Server 上发出这个命令来更新路由版本。
- 如果源 Replication Server 上尚未使用新功能，请使用 **sysadmin fast_route_upgrade** 升级路由。
- 如果在源 Replication Server 上已经使用了新功能，此命令将被拒绝。您必须使用 Replication Manager (RM) 升级路由。

权限

sysadmin fast_route_upgrade 需要“sa”权限。

另请参见

- admin show_route_versions （第 70 页）
- admin show_site_version （第 71 页）
- sysadmin site_version （第 372 页）

sysadmin hibernate_off

关闭 Replication Server 的休眠模式并使其返回活动状态。

语法

```
sysadmin hibernate_off [, string_ID]
```

参数

- **string_ID** – 有效标识符。如果使用 **sysadmin hibernate_on** 指定了 *string_ID*，则必须指定用于 **sysadmin hibernate_on** 的相同 string_ID。

  如果忘记了 *string_ID*，可以在 *rs_recovery* 系统表的 *text* 列中找到它。

  在路由升级或路由升级恢复成功完成后，如果需要为复制 Replication Server 关闭休眠模式，请使用此 Replication Server 的名称作为 *string_ID*。

示例

- **示例 1** – 此命令将关闭 Replication Server (TOKYO_RS) 的休眠模式：

```
sysadmin hibernate_off, TOKYO_RS
```

用法

- 休眠模式是 Replication Server 的一种状态，在此状态下：
  - 所有数据定义语言 (DDL) 命令都将被拒绝。
  - 大多数服务线程，例如，数据服务器接口 (DSI)、分配器和 Replication Server 接口 (RSI) 发送器线程，都会挂起。
  - 所有路由和连接全部挂起，并且
  - RSI 用户注销，并且无法重新登录到 Replication Server。
- 在休眠模式下可以执行系统信息 (**admin**) 和系统管理 (**sysadmin**) 类型命令。
- 在要为其关闭休眠模式的 Replication Server 上执行此命令。
- 在升级路由失败时，目标 Replication Server 可能会处于休眠模式。请不要使用 **sysadmin hibernate_off** 重新激活 Replication Server。应使用 Replication Manager 恢复路由升级。有关详细信息，请参见 Replication Manager 联机帮助。
- 有时，在成功升级路由之后，目标 Replication Server 会进入休眠模式。使用 **sysadmin hibernate_off** 可重新激活目标 Replication Server。

权限

sysadmin hibernate_off 需要“sa”权限。

另请参见

- sysadmin hibernate_on （第 361 页）

sysadmin hibernate_on

打开 Replication Server 的休眠模式（即挂起 Replication Server）。

警告！ 当 Replication Server 具有路由时，执行 **sysadmin hibernate_on** 命令可能会导致丢失检测。

语法

```
sysadmin hibernate_on [, string_ID]
```

参数

- **string_ID** – 有效标识符。必须使用与执行 **sysadmin hibernate_off** 时相同的 *string_ID*。使用 *string_ID* 可以确保在使用 Replication Server 时，其它人不会意外地关闭其休眠模式。

  如果忘记了 *string_ID*，可以在 *rs_recovery* 系统表的 *text* 列中找到它。

示例

- **示例 1** – 此命令将打开 Replication Server (TOKYO_RS) 的休眠模式：

  ```
  sysadmin hibernate_on, TOKYO_RS
  ```

用法

- 休眠模式是 Replication Server 的一种状态，在此状态下：
  - 所有数据定义语言 (DDL) 命令都将被拒绝。
  - 大多数服务线程，例如，数据服务器接口 (DSI)、分配器和 Replication Server 接口 (RSI) 发送器线程，都会挂起。
  - 所有路由和连接全部挂起，并且
  - RSI 用户注销，并且无法重新登录到 Replication Server。
- 在休眠模式下可以执行系统信息 (**admin**) 和系统管理 (**sysadmin**) 类型命令。
- 在要为其打开休眠模式的 Replication Server 上执行此命令。
- 为 Replication Server 打开休眠模式可以帮助您调试问题。

权限

sysadmin hibernate_on 需要“sa”权限。

另请参见

- sysadmin hibernate_off （第 360 页）

sysadmin issue_ticket

将 **rs_ticket** 标记插入到入站或出站队列中。

语法

```
sysadmin issue_ticket {,q_number} |{,ds_name, db_name},[,q_type], h1
[, h2 [, h3 [, h4]]] [,v]
```

参数

- **ds_name** – 数据库所在的数据服务器的名称。
- **db_name** – 数据库的名称。
- **q_number** – 标识稳定队列。
- **q_type** – 标识稳定队列类型。出站队列的值为 0，入站队列的值为 1。q_type 是可选的，如果未指定，则缺省队列类型为入站队列。
- **h1,h2,h3** – 每个参数各包含 1 – 10 个字符；由于这些参数用作标识符，因此必须遵循数据库标识符命名约定，采用任何合适的方式命名。标头参数不能以数字开头，并且不能是保留字。如果选择数字作为标头参数，则必须用引号括起。例如 '1'。
- **h4** – 包含 1 – 50 个字符。与 h1、h2 和 h3 一样，您能够以任何合适的方式使用此参数作为标识符。
- **v** – 标识 **rs_ticket** 的版本号。其值应为 1 或 2。如果未指定，则缺省值为 2。

示例

- **示例 1** – 此命令将一个事务插入到 Replication Server 入站队列中。

```
sysadmin issue_ticket, 103, 'start'
go
```

  其中：

  103 是到 Replication Server 的逻辑连接的 *q_number*。

- **示例 2**

  此示例将一个事务插入到 Replication Server 出站队列中。

```
sysadmin issue_ticket,103,0,'t6'
go
```

  其中：

  103 是 q_number，*0* 是出站队列类型，*t6* 是逻辑 Replication Server 的 h1 标头。

用法

使用 **sysadmin issue_ticket** 命令时：

- 您必须至少有一个来自 Replication Server 中复制数据库的预订。如果没有预订，分配器 (DIST) 模块不会将 **rs_ticket** 标记发送到相应的数据服务器接口 (DSI)。
- 主数据库 (PDB) 和 EXEC 模块的时间戳是插入的 **rs_ticket** 标记中的一个任意值。
- 您只能使用 *q_number*、*q_type* 或 *ds_name*、*db_name* 和 *q_type* 来指定稳定队列。在热备份环境中，入站队列与逻辑连接有关，Replication Server 没有用于备用数据库的入站队列。在对热备份使用 **sysadmin issue_ticket** 命令时：
  - 如果用户通过现有逻辑连接或活动数据库的物理连接指定稳定队列，则会将特定的 **rs_ticket** 标记写到 Replication Server 入站队列中。可以在主节点的复制数据库和备用数据库中找到 **rs_ticket** 记录。

注意: 在双 Replication Server(RS) DR 设置中，如果主服务器关闭，则会关闭主 Adaptive Server Enterprise (ASE) 和 RS。在这种情况下，请确保清除出站队列，之后，客户端才能故障切换到 DR ASE。为此，您需要向出站队列中插入一个票据。只要在 rs_ticket_history 表中找到该票据，客户端就可以故障转移。

- 如果用户通过备用数据库的现有物理连接指定稳定队列，则会显示一条错误消息，指示不存在该入站队列。

sysadmin lmconfig

在 Replication Server 中配置和显示与许可证管理相关的信息。

语法

```
sysadmin lmconfig,
[ , edition [, edition_type ]
  , license_type [, license_type_name ]
  , smtp_host [, smtp_host_name ]
  , smtp_port [, smtp_port_number ]
  , email_sender [, sender_email_address ]
  , email_recipients [, email_recipients ]
  , email_severity [, email_severity ]]
```

参数

- **sysadmin lmconfig** – 不使用任何参数时，可显示基本的许可证状态信息。
- **edition, `edition_type`** – 是指定许可证版本的静态配置参数。

  `edition_type` 的值包括:

  - `null` – 缺省值。指定 null 值时，不配置任何产品版本，Replication Server 将使用任意版本的许可证启动。
  - `EE` – 指示 Enterprise Edition。
  - `RL` – 指示实时装载 (RTL) 版本。
- **license type, `license_type_name`** – 是用来为 Replication Server 的安装指定许可证类型的静态配置参数，该参数仅在指定非 null 版本时才有效。

  `license_type_name` 最常见的有效值为:

  - `SR` – 服务器许可证
  - `SV` – 备用服务器许可证
  - `AR` – 应用程序服务器许可证
  - `BR` – 特定于应用程序的备用服务器许可证
  - `IC` – Internet 访问 CPU 许可证
  - `AC` – 特定于应用程序的 CPU 许可证
  - `BC` – 特定于应用程序的备用 CPU 许可证

- `CP` – CPU 许可证
- `SF` – 备用 CPU 许可证
- `null` – 缺省值

注意： 除了此列表外，**sysadmin lmconfig** 还接受表示专门和遗留许可证类型的两字母缩写形式。如果不接受该许可证类型，请将该类型设置为 null 并使用网络许可证服务器选项文件来控制 Replication Server 所使用的许可证。

- **smtp host, smtp host name** – 指定用于发送许可证事件通知的电子邮件的 SMTP 主机。
- **smtp port, smtp port number** – 指定用于发送许可证事件通知的电子邮件的 SMTP 端口。
- **email sender, sender email address** – 指定在许可证事件电子邮件通知中用作发件人地址的电子邮件地址。
- **email recipients, email recipients** – 是以逗号分隔的接收许可证事件电子邮件通知的收件人列表。
- **email severity, email severity** – 是导致发送电子邮件通知的错误的最低严重性。缺省为错误，其它可能性包括警告和信息性消息。

示例

- **示例 1** – 显示系统的基本许可证配置信息：

```
sysadmin lmconfig
go
```

返回的结果为：

```
Parameter Name     Config Value
-------------------------  ------------
edition                          null
license_type                     null
smtp_host                        smtp
email_recipients                cshi
email_severity                  ERROR
smtp_port                          25
email_sender                       cshi
License Name     Version     Quantity Status         Expiry Date
-------------- ---------    -------- ------ --------------
REP_SERVER       2005.0425      1        graced     Sep 11 2005
2:40AM
REP_HVAR_ASE     2005.0425      1        graced     Sep 11 2005
2:40AM
Property Name     Property Value
-------------         --------------
PE                       null
LT                       null
```

用法

- 如果不指定版本或使用“null”，Replication Server 将查找并使用它在启动时找到的任意许可证版本。
- **sysadmin lmconfig** 设置的配置选项存储在 `sylapi` 属性文件中。

权限

sysadmin lmconfig 需要“sa”权限。

sysadmin log_first_tran

将 DSI 队列中的第一个事务写入例外日志。

语法

```
sysadmin log_first_tran, [n], data_server, database
```

参数

- **n** – 指定要写入数据库例外日志以及写入 Replication Server 日志或 **sysadmin dump_file** 命令指定的备用日志文件的事务数。
- **data_server** – 包含此数据库的数据服务器的名称。
- **database** – 数据库名称，此数据库的 DSI 队列中的第一个事务将被写入。

示例

- **示例 1** – 将此 DSI 队列中的第一个事务写入例外日志：

  ```
  sysadmin log_first_tran, SYDNEY_DS, pubs2
  ```

- **示例 2** – 将 DSI 队列中的前五个事务写入数据库例外日志中，并写入 Replication Server 日志或由 **sysadmin dump_file** 命令指定的位置：

  ```
  sysadmin log_first_tran, 5, SYDNEY_DS, pubs2
  ```

- **示例 3** – 将 DSI 队列中的前两个事务写入数据库例外日志以及 `SYDNEY_RS.log` 文件中。最后的 **sysadmin dump_file** 命令将关闭 `SYDNEY_RS.log` 文件：

  ```
  sysadmin dump_file SYDNEY_RS.log
  sysadmin log_first_tran, 2, SYDNEY_DS, pubs2
  sysadmin dump_file
  ```

用法

- 使用 **sysadmin log_first_tran** 将 DSI 队列中的前 *n* 个事务写入例外日志，并写入 Replication Server 日志或 **sysadmin dump_file** 命令指定的备用日志文件。
- 此命令不会将前 *n* 个事务从队列中删除。

- 例外日志由 3 个表组成，分别是：*rs_exceptshdr*、*rs_exceptscmd* 和 *rs_systext*。有关这些表的详细说明，请参见“Replication Server 系统表”。

权限

sysadmin log_first_tran 需要“sa”权限。

另请参见

- admin who （第 88 页）

sysadmin purge_all_open

从 Replication Server 的入站队列中清除所有打开的事务。

语法

```
sysadmin purge_all_open, q_number, q_type
```

参数

- **q_number, q_type** – 标识要进行清除的稳定队列。使用 **admin who**、**admin who, sqm** 和 **admin who, sqt** 可以查找这些值。

示例

- **示例 1** – 从队列 103:1 中清除所有打开的事务：

```
sysadmin purge_all_open, 103, 1
```

用法

- 使用 **sysadmin purge_all_open** 从 Replication Server 的入站队列中清除所有打开的事务。只能从入站队列中清除打开的事务。

  注意： 如果 RepAgent 已经转发事务开始记录，还可能已转发事务内部的某些命令，但尚未转发事务提交或中止记录时，这个事务就处于打开状态。

- 如果在数据服务器日志完全转发到 Replication Server 之前必须截断此日志，从而在 Replication Server 入站队列中留下一些打开的事务，**sysadmin purge_all_open** 就非常有用。必须使用 **sysadmin purge_all_open** 来明确删除这些打开的事务。

  警告！ 只有 在入站队列中存在打开的事务，并且您确定 RepAgent 不会从此日志转发提交或中止记录，您才能使用 **sysadmin purge_all_open**。

- Replication Server 需要足够的存储空间来清除稳定队列。如果存储空间不足，就会显示以下错误消息：

```
This RS is out of Disk Space. Use another session to
add disk space for this command to proceed.
```

如果发生这种情况，则启动另一个 **isql** 会话并为 Replication Server 添加稳定存储空间。在具有足够的存储空间之前，**sysadmin purge_all_open** 无法继续执行。
- 要查看被删除事务的内容，请在使用此命令之前执行 **sysadmin sqt_dump_queue**。
- 如果此队列没有打开的事务，执行此命令后，队列保持不变。如果在清除事务后重新启动 Replication Server，这些事务可能会因恢复操作而再次出现。

权限

sysadmin purge_all_open 需要“sa”权限。

另请参见

- admin who （第 88 页）
- alter partition （第 149 页）
- create partition （第 253 页）
- sysadmin purge_first_open （第 368 页）
- sysadmin sqt_dump_queue （第 382 页）

sysadmin purge_first_open

从 Replication Server 的入站队列清除第一个打开的事务。

语法

```
sysadmin purge_first_open, q_number, q_type
```

参数

- **q_number, q_type** – 标识要清除的稳定队列。使用 **admin who**、**admin who, sqm** 和 **admin who, sqt** 可以查找这些值。

示例

- **示例 1** – 从队列 103:1 中清除第一个打开的事务：

```
sysadmin purge_first_open, 103, 1
```

用法

- **sysadmin purge_first_open** 从 Replication Server 的入站队列中删除第一个打开的事务。RepAgent 线程从数据库日志传送事务，每次传送一条记录。如果 RepAgent 已经转发事务开始记录，还可能已转发事务内部的某些命令，但尚未转发事务提交或中止记录时，这个事务就处于打开状态。
- **sysadmin purge_first_open** 只能用于入站对列。

- Replication Server 需要足够的空间才能从稳定队列清除第一个打开的事务。如果磁盘空间不足，就会显示以下错误消息：

  ```
  This RS is out of Disk Space. Use another session to
  add disk space for this command to proceed.
  ```

  如果发生此情况，则启动另一个 **isql** 会话并为 Replication Server 添加稳定存储空间（磁盘空间）。在具有足够的存储空间之前，**sysadmin purge_first_open** 无法继续执行。
- 要查看被删除事务的内容，请在使用此命令之前执行 **sysadmin sqt_dump_queue**。
- 要显示有关入站队列中第一个事务的信息，请使用 **admin who, sqt**。如果第一个事务的状态为“打开”(ST:O)，则可以将其从队列中删除。
- 当 Adaptive Server 日志中存在未提交的事务时，**sysadmin purge_first_open** 命令是非常有用的。打开的事务将被 RepAgent 传递到 Replication Server。因为存在打开的事务，Replication Server 不能截断此入站队列。如果事务长时间保持打开状态，入站队列将被占满，Replication Server 可能会用尽队列空间。
- 如果此队列的第一个事务未打开，使用此命令后，队列将保持不变。如果在删除事务后重新启动 Replication Server，事务可能会因恢复操作而再次出现。

警告！ 只有在已经确定入站队列是在未提交的事务处卡住的（使用 **admin who, sqt** 和 **admin who, sqm**），您才能使用 **sysadmin purge_first_open**。

权限

sysadmin purge_first_open 需要“sa”权限。

另请参见

- admin who （第 88 页）
- alter partition （第 149 页）
- create partition （第 253 页）
- sysadmin dump_queue （第 350 页）
- sysadmin purge_all_open （第 367 页）

sysadmin purge_route_at_replicate

从复制 Replication Server 中删除对主 Replication Server 的所有引用。

语法

```
sysadmin purge_route_at_replicate, replication_server
```

参数

- **replication_server** – 要从复制的 RSSD 中清除的主 Replication Server 的名称。

示例

- **示例 1** – 从复制的 RSSD 中清除主 Replication Server，即 TOKYO_RS：

```
sysadmin purge_route_at_replicate, TOKYO_RS
```

用法

- 从指定的主 Replication Server 删除路由之后，使用 **sysadmin purge_route_at_replicate** 删除所有源自此 Replication Server 的预订和路由信息。在主 Replication Server 上执行 **drop route with nowait** 之后，此命令很有用。
- 如果当前 Replication Server 与指定的主 Replication Server 之间存在路由，在执行此命令之前必须删除此路由。
- 如果在主 Replication Server 上执行 **drop route with nowait** 时实现预订，则实现队列可能保留在复制 Replication Server 中。使用 **sysadmin drop_queue** 可以删除此队列。

警告! 只有在主 Replication Server 上执行了 **drop route with nowait** 命令或主 Replication Server 已丢失且无法恢复的情况下，才可以使用 **sysadmin purge_route_at_replicate**。

权限

sysadmin purge_route_at_replicate 需要“sa”权限。

另请参见

- drop route （第 309 页）
- rs_helproute （第 541 页）

sysadmin restore_dsi_saved_segments

恢复积压的事务。

语法

```
sysadmin restore_dsi_saved_segments, data_server, database
```

参数

- **data_server** – 数据服务器名称。
- **database** – 数据库名称。

示例

- **示例 1** – 恢复 TOKYO_DS 数据服务器中 *pubs2* 数据库的积压事务：

```
sysadmin restore_dsi_saved_segments, TOKYO_DS, pubs2
```

用法

- 必须明确挂起 DSI，才能使用此命令恢复保存的段。
- 由于为此连接指定了保存间隔（使用 **alter connection**）而保存的任何积压事务，都将被选择恢复到此数据库。Replication Server 使用 **rs_get_lastcommit** 确定要过滤哪些事务。

权限

sysadmin restore_dsi_saved_segments 需要“sa”权限。

另请参见

- configure connection （第 180 页）

sysadmin set_dsi_generation

更改 Replication Server 中的数据库生成号，以防止在复制数据库恢复后应用 DSI 稳定队列中的事务。

语法

```
sysadmin set_dsi_generation, gen_number, primary_data_server,
       primary_database, replicate_data_server, replicate_database
```

参数

- **gen_number** – 数据库的新生成号。这是一个 0 至 65,535 之间的整数。
- **primary_data_server** – 主节点上数据服务器的名称。
- **primary_database** – 主数据库的名称。
- **replicate_data_server** – 复制数据服务器的名称。
- **replicate_database** – 复制数据库的名称。

示例

- **示例 1** – 将新的 DSI 生成号设置为 105。以前的生成号为 104 或更低：

```
sysadmin set_dsi_generation 105 NY_DS, ny_db, SF_DS,
         sf_db
```

用法

在数据库转储恢复期间使用 **sysadmin set_dsi_generation**。如果在恢复期间以外的时间更改生成号，可能会导致复制数据库上出现数据错误。

有关恢复过程的完整说明，请参见《Replication Server 管理指南第二卷》。

权限

sysadmin set_dsi_generation 需要“sa”权限。

另请参见

- admin get_generation （第 52 页）
- configure connection （第 180 页）
- dbcc dbrepair （第 445 页）
- dbcc settrunc （第 447 页）
- rebuild queues （第 320 页）

sysadmin site_version

设置 Replication Server 的节点版本号。通过此命令，可以使用相应版本中的软件功能，防止降级到更低的版本。如果 Replication Server 使用 ERSSD，此命令还会关闭 ERSSD、升级其数据库文件并重新启动 ERSSD。

注意： 如果 Replication Server 使用 ERSSD，此命令可能会由于正在重新启动 ERSSD 而导致关闭某些线程。在重新启动关闭的所有线程后，才会继续进行复制。

语法

```
sysadmin site_version [, version]
```

参数

- **version** – Replication Server 的节点版本号。

版本号	节点版本
11.5 之前	不适用
11.5	1150
12.0	1200
12.5	1250
12.6	1260
15.0, 15.0.1	1500
15.1	1510
15.2	1520

版本号	节点版本
15.5	1550

低于 11.5 的版本不存在节点版本号。维护版本可能支持更高的节点版本号。

示例

- **示例 1** – 显示 Replication Server 的当前节点版本号：

```
sysadmin site_version
```

- **示例 2** – 更改节点版本号以便与版本 15.5 相对应：

```
sysadmin site_version, 1550
```

用法

- 要设置当前 Replication Server 的节点版本号，请执行带 *version* 参数的 **sysadmin site_version**。
  输入的节点版本号不得高于软件版本号或 Replication Server 的版本级别。
- 要显示 Replication Server 的节点版本号，请执行不带 *version* 参数的 **sysadmin site_version**。
- 在 Replication Server 的节点版本中设置的版本之前，您可以使用新软件功能。
- 对于新安装的 Replication Server 15.5 版，节点版本号为 1550。
- 有关特定 Replication Server 软件版本中引入的功能的详细信息，请参见适用于该版本的《Replication Server 新增功能指南》。

警告！ 设置节点版本号时，不能降级到更低的版本。

- 有关安装或升级 Replication Server 的详细信息，请参见适用于您的平台的 Replication Server 安装指南和配置指南。

混合版本复制系统

在混合版本复制系统中，不同的 Replication Server 具有不同的节点版本。在这种系统中，有些功能只适用于具有较高节点版本的 Replication Server。例如，主 Replication Server 及其某个复制 Replication Server 的节点版本为 1550，而它的其它复制 Replication Server 的节点版本为 1260。当表复制定义具有 *timestamp* 列时，节点版本较低的复制 Replication Server 只能将 *timestamp* 作为 *varbinary* (8) 预订，而节点版本为 1550 的复制 Replication Server 可以直接预订 *timestamp* 列。

升级路由

- 如果将路由任意一端的一个或两个 Replication Server 升级为更高版本级别，而且将节点版本设置为更高的级别，则需要升级路由。升级路由将重新实现系统表中的数据，使新升级的 Replication Server 可以使用与新功能相关的信息。
  路由升级有两种可能的情形：

- 如果有 Replication Manager，请使用 Replication Manager 升级路由。有关升级路由的说明，请参见 Replication Manager 联机帮助。
- 如果源 Replication Server 上尚未使用新功能，请使用 **sysadmin fast_route_upgrade** 升级路由。

例如，如果将一个 Replication Server 从版本 12.6 升级到版本 15.0，并相应设置了其节点版本，则需要从另一个版本为 15.0 的 Replication Server 升级路由。升级路由时，新升级的 Replication Server 将接收来自 Replication Server 15.0 版的信息，如表的其它复制定义。

有关升级路由的详细信息，请参见《Replication Server 配置指南》。

有关版本信息的系统表

版本信息存储在 *rs_version* 系统表中。*rs_routes* 系统表中也包含版本信息。路由版本信息存储在 *rs_routeversions* 系统表中。

权限

sysadmin site_version 需要“sa”权限。

另请参见

- admin version （第 86 页）
- sysadmin fast_route_upgrade （第 359 页）
- sysadmin system_version （第 385 页）

sysadmin skip_bad_repserver_cmd

指示 Replication Server 在 Replication Agent 下次启动时跳过失败的复制定义请求。

对复制定义更改请求的更改进程使用 **sysadmin skip_bad_repserver_cmd**。在使用 **sysadmin skip_bad_repserver_cmd** 之前，请参见《Replication Server 管理指南第一卷》中的“管理复制的表”。

警告! 使用 **sysadmin skip_bad_repserver_cmd** 需小心。如果您在主 Replication Server 中执行该命令，然后重新启动 Replication Agent 而不执行更正的复制定义命令，主数据可能会使用错误的复制定义版本进行复制。

语法

```
sysadmin skip_bad_repserver_cmd, pds_name, pdb_name
```

参数

- **pds_name** – 主数据服务器名。
- **pdb_name** – 主数据库名称。

示例

- **示例 1** – 在此示例中，**sysadmin skip_bad_repserver_cmd** 指示 Replication Server 和 Replication Agent 在 SYDNEY_DS 数据服务器的 pubs2 数据库中跳过最后一个失败的复制定义命令：

```
sysadmin skip_bad_repserver_cmd, SYDNEY_DS, pubs2
```

用法

- *pds_name* 和 *pdb_name* 标识受失败的复制定义请求影响的特定 Replication Agent。
- 使用 **sysadmin skip_bad_repserver_cmd** 指示 Replication Server 跳过由 Replication Agent 发送的失败复制定义请求。当复制定义命令在主 Replication Server 上失败时，Replication Agent 将关闭。如果您重新启动 Replication Agent，则失败的命令将再次执行，除非 Replication Server 跳过该命令。执行 **sysadmin skip_bad_repserver_cmd** 之后，在启动 Replication Agent 之前在 Replication Server 上执行更正的复制定义请求。

权限

sysadmin skip_bad_repserver_cmd 需要“sa”权限。

另请参见

- admin verify_repserver_cmd （第 84 页）
- rs_send_repserver_cmd （第 546 页）
- alter replication definition （第 152 页）
- alter applied function replication definition （第 107 页）
- alter request function replication definition （第 159 页）
- create replication definition （第 258 页）
- create applied function replication definition （第 206 页）
- create request function replication definition （第 269 页）
- drop replication definition （第 308 页）

sysadmin sqm_purge_queue

清除稳定队列中的所有消息。

警告！ 清除稳定队列中的消息可能造成数据丢失，因此只有在 Sybase 技术支持人员的指导下才能进行。Replication Server 无法将已清除的消息发送到目标数据库或目标 Replication Server，这样就会导致复制系统中出现不一致问题。如果某一队列包含预订标记消息或路由消息，使用此命令就可能会导致严重后果。

语法

```
sysadmin sqm_purge_queue, q_number, q_type
```

参数

- **q_number, q_type** – 标识要清除的稳定队列。请使用 **admin who**、**admin who, sqm** 或 **admin who, sqt** 查找这些值。

示例

- **示例 1** – 清除编号为 103 的入站队列中的所有消息:

```
sysadmin sqm_purge_queue, 103, 1
```

用法

- **sysadmin sqm_purge_queue** 从稳定队列中删除要发送到另一个 Replication Server 的消息。在队列被消息占满的情况下使用此命令。
- 只有在 Replication Server 以独立模式启动后才能执行 **sysadmin sqm_purge_queue**。

权限

需要“sa”权限。

另请参见

- admin who （第 88 页）
- repserver （第 553 页）

sysadmin sqm_unzap_command

取消删除稳定队列中的消息。

语法

```
sysadmin sqm_unzap_command, q_number, q_type,
 seg, blk, row
```

参数

- **q_number, q_type** – 标识包含待恢复消息的稳定队列。使用 **admin who**、**admin who, sqm** 和 **admin who, sqt** 可以查找这些值。
- **seg** – 标识稳定队列中包含要取消删除的消息的段。
- **blk** – 标识段中的 16K 块。块的编号是从 1 到 64。

- **row** – 块中要取消删除的命令所在的行号。

用法

- 要使用 **sysadmin sqm_unzap_command**，Replication Server 必须处于独立模式。
- **sysadmin sqm_unzap_command** 删除稳定队列中消息的删除标记。使用此命令可恢复已用 **sysadmin sqm_zap_command** 标记为删除的消息。
- 使用 **sysadmin dump_queue** 可定位要恢复的消息。

权限

sysadmin sqm_unzap_command 需要“sa”权限。

另请参见

- admin who （第 88 页）
- sysadmin drop_queue （第 348 页）
- sysadmin sqm_zap_command （第 379 页）
- sysadmin dump_queue （第 350 页）

sysadmin sqm_unzap_tran

将特定事务恢复到稳定队列中，并返回指出已恢复命令的数量的消息。

语法

```
sysadmin sqm_unzap_tran {, q_number, | server [,database]},
        q_type, lqid
        [, {L0 | L1 | L2 | L3}]
        [, {RSSD | client | “log” | file_name}]
```

参数

- **q_number | server[, database]** – 标识稳定队列。使用 *q_number* 或 *server*[, *database*] 可指定队列号。使用 **admin who**、**admin who, sqm** 和 **admin who, sqt** 可以标识队列号。
- **q_type** – 稳定队列的队列类型。值为“0”表示出站队列；值为“1”表示入站队列。使用 **admin who**、**admin who, sqm** 和 **admin who, sqt** 可以标识队列类型。
- **lqid** – 稳定队列事务的任何命令的本地队列 ID。*lqid* 标识要恢复到稳定队列中的事务。格式：*seg,blk,row*。
- **L0** – 转储已恢复事务的内容。如果未指定 **L0**、**L1**、**L2** 或 **L3**，则这是缺省行为。
- **L1** – 仅转储已恢复事务的 **begin** 和 **end** 命令。
- **L2** – 将已恢复事务的 **begin** 和 **end** 命令与事务中其它命令的前 100 个字符一起转储。

- **L3** – 转储已恢复事务的所有命令。除了 SQL 语句以外，所有其它命令均以注释形式显示。仅当使用 *file_name* 选项或 **sysadmin dump_file** 命令指定备用日志文件时，才可以使用 **L3**。不能将 **L3** 用于 **RSSD** 或 **client** 选项。
- **RSSD** – 强制输出到 RSSD 中的系统表。
- **客户端** – 强制输出到发出此命令的客户端。
- **"log"** – 强制输出到 Replication Server 日志文件。
- **file_name** – 强制输出到 *file_name* 日志文件。也可以使用 **sysadmin dump_file** 命令设置备用日志文件。

示例

- **示例 1** – 恢复队列 103:1 中 LQID 为 0:15:2 的事务并将其转储到 Replication Server 日志中：

```
sysadmin sqm_unzap_tran, 103, 1, 0, 15, 2
```

- **示例 2** – 恢复 SYDNEY_DS.*pubs2* 的入站队列中 LQID 为 0:15:2 的事务并将其转储到 Replication Server 日志中：

```
sysadmin sqm_unzap_tran, SYDNEY_DS, pubs2, 1, 0, 15,
     2, “log”
```

- **示例 3** – 恢复队列 103:1 中 LQID 为 0:15:2 的事务并将该事务的 **begin** 和 **end** 命令转储到 Replication Server 日志中：

```
sysadmin sqm_unzap_tran, 103,1, 0, 15, 2, L1
```

- **示例 4** – 恢复队列 103:1 中 LQID 为 0:15:2 的事务并将其转储到 Replication Server 日志中。所有命令均被截断为 100 个字符：

```
sysadmin sqm_unzap_tran, 103,1, 0, 15, 2, L2
```

- **示例 5** – 恢复队列 103:1 中 LQID 为 0:15:2 的事务并将其转储到 `SYDNEY_RS.log` 文件中：

```
sysadmin sqm_unzap_tran, 103,1, 0, 15, 2, L3,
     SYDNEY_RS.log
```

- **示例 6** – 恢复队列 103:1 中 LQID 为 0:15:2 的事务并将其转储到 RSSD 中：

```
sysadmin sqm_unzap_tran, 103, 1, 0, 15, 2, RSSD
```

- **示例 7** – 恢复队列 103:1 中 LQID 为 0:15:2 的事务并将其转储到客户端：

```
sysadmin sqm_unzap_tran, 103, 1, 0, 15, 2, client
```

用法

- 要使用 **sysadmin sqm_unzap_tran**，Replication Server 必须处于独立模式。
- **sysadmin sqm_unzap_tran** 删除稳定队列中事务的删除标记。使用此命令可恢复已用 **sysadmin sqm_zap_tran** 标记为删除的事务。
- 使用 **sysadmin dump_queue** 可定位要恢复的事务。
- **sysadmin sqm_unzap_tran** 将已恢复的事务内容转储到以下位置之一：

- Replication Server 日志
- 备用日志文件
- RSSD
- 发出此命令的客户端

要将队列转储到 RSSD 或客户端，**sysadmin dump_queue** 的最后一个参数必须是 **RSSD** 或 **client**。

如果未指定 **RSSD** 或 **client** 选项，或者指定了“**log**”选项，则会输出到 Replication Server 日志中。

如果通过 **sysadmin dump_file** 命令或者 *file_name* 选项指定用于转储队列的备用日志文件，则输出到此备用转储文件中。

权限

sysadmin sqm_unzap_tran 需要“sa”权限。

另请参见

- admin who （第 88 页）
- sysadmin sqm_unzap_command （第 376 页）
- sysadmin sqm_zap_command （第 379 页）
- sysadmin sqm_zap_tran （第 380 页）

sysadmin sqm_zap_command

删除稳定队列中的单个消息。

语法

```
sysadmin sqm_zap_command, q_number, q_type,
 seg, blk, row
```

参数

- **q_number, q_type** – 标识包含要删除的消息的稳定队列。使用 **admin who**、**admin who, sqm** 和 **admin who, sqt** 可以查找这些值。
- **seg** – 标识稳定队列中的段。
- **blk** – 标识段中的 16K 块。块的编号是从 1 到 64。
- **row** – 块中要删除的命令所在的行号。

示例

- **示例 1** –

```
sysadmin sqm_zap_command
```

```
sysadmin sqm_zap_command, 103, 1, 15, 65, 2
```

用法

- 要使用 **sysadmin sqm_zap_command**，Replication Server 必须处于独立模式。
- 使用 **sysadmin dump_queue** 可定位要删除的消息。
- **sysadmin sqm_zap_command** 将稳定队列中的消息标记为删除。当 Replication Server 处理此队列时，将忽略已标记的消息。
- 可使用 **sysadmin sqm_unzap_command** 恢复消息。此命令将删除消息的删除标记。
- 如果删除消息后以正常模式重新启动 Replication Server，队列中包含此消息的部分可能已被处理。如果是这样，则无法使用 **sysadmin sqm_unzap_command** 恢复此消息。

权限

sysadmin sqm_zap_command 需要“sa”权限。

另请参见

- admin who （第 88 页）
- sysadmin dump_queue （第 350 页）
- sysadmin sqm_unzap_command （第 376 页）

sysadmin sqm_zap_tran

从稳定队列中删除特定事务，并返回指出已删除命令的数量的消息。

语法

```
sysadmin sqm_zap_tran {, q_number, | server [,database]},
        q_type, lqid
        [, {L0 | L1 | L2 | L3}]
        [, {RSSD | client | “log” | file_name}]
```

参数

- **q_number | server[, database]** – 标识稳定队列。使用 *q_number* 或 *server*[, *database*] 可指定队列号。使用 **admin who**、**admin who, sqm** 和 **admin who, sqt** 可以标识队列号。
- **q_type** – 稳定队列的队列类型。值为“0”表示出站队列；值为“1”表示入站队列。使用 **admin who**、**admin who, sqm** 和 **admin who, sqt** 可以标识队列类型。
- **lqid** – 稳定队列事务的任何命令的本地队列 ID。*lqid* 标识要从稳定队列中删除的事务。格式：*seg,blk,row*。
- **L0** – 转储已删除事务的内容。如果未指定 **L0**、**L1**、**L2** 或 **L3**，则这是缺省行为。

- **L1** – 仅转储已删除事务的 **begin** 和 **end** 命令。
- **L2** – 将已删除事务的 **begin** 和 **end** 命令与事务中其它命令的前 100 个字符一起转储。
- **L3** – 转储已删除事务的所有命令。除了 **SQL** 语句以外，所有其它命令均以注释形式显示。仅当使用 *file_name* 选项或 **sysadmin dump_file** 命令指定备用日志文件时，才可以使用 **L3**。不能将 **L3** 用于 **RSSD** 或 **client** 选项。
- **RSSD** – 强制输出到 RSSD 中的系统表。
- **客户端** – 强制输出到发出此命令的客户端。
- **"log"** – 强制输出到 Replication Server 日志文件。
- **file_name** – 强制输出到 *file_name* 日志文件中。也可以使用 **sysadmin dump_file** 命令设置备用日志文件。

示例

- **示例 1** – 删除队列 103:1 中 LQID 为 0:15:2 的事务并将其转储到 Replication Server 日志中：

```
sysadmin sqm_zap_tran, 103, 1, 0, 15, 2
```

- **示例 2** – 删除 *SYDNEY_DS.pubs2* 的入站队列中 LQID 为 0:15:2 的事务删除并将其转储到 Replication Server 日志中：

```
sysadmin sqm_zap_tran, SYDNEY_DS, pubs2, 1, 0, 15,
     2, "log"
```

- **示例 3** – 删除队列 103:1 中 LQID 为 0:15:2 的事务并将该事务的 **begin** 和 **end** 命令转储到 Replication Server 日志中：

```
sysadmin sqm_zap_tran, 103,1, 0, 15, 2, L1
```

- **示例 4** – 删除队列 103:1 中 LQID 为 0:15:2 的事务并将其转储到 Replication Server 日志中。所有命令均被截断为 100 个字符：

```
sysadmin sqm_zap_tran, 103,1, 0, 15, 2, L2
```

- **示例 5** – 删除队列 103:1 中 LQID 为 0:15:2 的事务并将其转储到 `SYDNEY_RS.log` 文件中：

```
sysadmin sqm_zap_tran, 103,1, 0, 15, 2, L3,
     SYDNEY_RS.log
```

- **示例 6** – 删除队列 103:1 中 LQID 为 0:15:2 的事务并将其转储到 RSSD 中：

```
sysadmin sqm_zap_tran, 103, 1, 0, 15, 2, RSSD
```

- **示例 7** – 删除队列 103:1 中 LQID 为 0:15:2 的事务并将其转储到客户端：

```
sysadmin sqm_zap_tran, 103, 1, 0, 15, 2, client
```

用法

- 要使用 **sysadmin sqm_zap_tran**，Replication Server 必须处于独立模式。
- 使用 **sysadmin dump_queue** 可定位要删除的事务。

- **sysadmin sqm_zap_tran** 将稳定队列中的事务标记为删除。当 Replication Server 处理此队列时，将忽略已标记的事务。
- 可使用 **sysadmin sqm_unzap_tran** 恢复事务。**sysadmin sqm_unzap_tran** 命令可删除事务的删除标记。
- 如果删除事务后以正常模式重新启动 Replication Server，则队列中包含此事务的部分可能已被处理。如果是这样，则无法使用 **sysadmin sqm_unzap_tran** 恢复此事务。
- **sysadmin sqm_zap_tran** 将标记为删除的事务转储到以下位置之一：
  - Replication Server 日志
  - 备用日志文件
  - RSSD
  - 发出此命令的客户端

  要将队列转储到 RSSD 或客户端，**sysadmin dump_queue** 的最后一个参数必须是 **RSSD** 或 **client**。

  如果未指定 **RSSD** 或 **client** 选项，或者指定了“**log**”选项，则会输出到 Replication Server 日志中。

  如果通过 **sysadmin dump_file** 命令或者 *file_name* 选项指定用于转储队列的备用日志文件，则输出到此备用转储文件中。

权限

sysadmin sqm_zap_command 需要“sa”权限。

另请参见

- admin who （第 88 页）
- sysadmin dump_queue （第 350 页）
- sysadmin sqm_unzap_command （第 376 页）
- sysadmin sqm_unzap_tran （第 377 页）
- sysadmin sqm_zap_command （第 379 页）

sysadmin sqt_dump_queue

转储入站队列或 DSI 队列的事务高速缓存。

语法

```
sysadmin sqt_dump_queue {, q_number | server[, database]},
        q_type, reader
        [, {open | closed | read}]
        [, num_cmds]
        [, {L0 | L1 | L2 | L3}]
        [, {RSSD | client | “log” | file_name}]
```

参数

- **q_number | server[, database]** – 标识入站队列或 DSI 队列。使用 *q_number* 或 *server*[, *database*] 可指定队列号。使用 **admin who**、**admin who, sqm** 和 **admin who, sqt** 可以标识队列号。
- **q_type** – 稳定队列的队列类型。值为“0”表示出站队列；值为“1”表示入站队列。使用 **admin who**、**admin who, sqm** 和 **admin who, sqt** 可以标识队列类型。
- **读取程序** – 标识要为其转储稳定队列的读取器。此参数适用于要求多个读取器的功能，如热备份应用程序。可从 **admin sqm_readers** 或 **admin who, sqt** 获取读取号。如果您不使用多个读取器，请为读取器输入“0”。
- **open** – 仅转储打开的事务。如果使用此选项，请在 *q_type* 和 **open** 标志之间插入一个逗号。
- **closed** – 转储 SQT 高速缓存中所有已提交的事务。
- **read** – 转储 SQT 高速缓存中所有已恢复的读取事务。
- **num_cmds** – 指定要转储的命令数。将 *num_cmds* 设置为 -1 可转储 SQT 高速缓存中的所有命令。
- **L0** – 转储 SQT 高速缓存的所有内容。如果未指定 **L0**、**L1**、**L2** 或 **L3**，则这是缺省行为。
- **L1** – 仅转储 SQT 高速缓存中的事务的 **begin** 和 **end** 命令。
- **L2** – 将 SQT 高速缓存事务的 **begin** 和 **end** 命令与事务中所有其它命令的缩短版本一起转储。
- **L3** – 转储高速缓存中的所有内容。除 **SQL** 语句外，所有其它命令均以注释形式显示。仅当使用 *file_name* 选项或 **sysadmin dump_file** 命令指定备用日志文件时，才可以使用 **L3**。不能将 **L3** 用于 **RSSD** 或 **client** 选项。
- **RSSD** – 强制输出到 RSSD 中的系统表。
- **客户端** – 强制输出到发出此命令的客户端。
- **"log"** – 强制输出到 Replication Server 日志文件。
- **file_name** – 强制输出到 *file_name* 指定的备用日志文件。也可以使用 **sysadmin dump_file** 命令设置备用日志文件。此文件的位置记录在 Replication Server 日志中。

示例

- **示例 1** – 从事务高速缓存中转储队列 103:1 中所有已恢复的事务：

```
sysadmin sqt_dump_queue, 103, 1, 0
```

- **示例 2** – 将事务高速缓存中 SYDNEY_DS.*pubs2* 的入站队列中的所有已恢复事务转储到 Replication Server 日志中：

```
sysadmin sqt_dump_queue, SYDNEY_DS, pubs2, 1, 0
```

- **示例 3** – 将事务高速缓存中队列 103:1 的所有已恢复的打开事务转储到 Replication Server 日志中：

```
sysadmin sqt_dump_queue, 103,1, 0, open
```

- **示例 4** – 将事务高速缓存中队列 103:1 的所有已恢复的关闭事务转储到 Replication Server 日志中：

```
sysadmin sqt_dump_queue, 103,1, 0, closed
```

- **示例 5** – 将事务高速缓存中队列 103:1 的所有已恢复的读取事务转储到 Replication Server 日志中：

```
sysadmin sqt_dump_queue, 103,1, 0, read
```

- **示例 6** – 将事务高速缓存中队列 103:1 的已恢复事务的前 10 个命令转储到 Replication Server 日志中：

```
sysadmin sqt_dump_queue, 103,1, 0, 10
```

- **示例 7** – 将事务高速缓存中队列 103:1 的所有已恢复事务的 **begin** 和 **end** 命令转储到 Replication Server 日志中：

```
sysadmin sqt_dump_queue, 103,1, 0, L1
```

- **示例 8** – 将事务高速缓存中队列 103:1 的所有已恢复事务转储到 Replication Server 日志中。所有命令均被截断为 100 个字符：

```
sysadmin sqt_dump_queue, 103,1, 0, L2
```

- **示例 9** – 将事务高速缓存中队列 103:1 的所有已恢复事务转储到 `SYDNEY_RS.log` 文件中：

```
sysadmin sqt_dump_queue, 103,1, 0, L3, SYDNEY_RS.log
```

- **示例 10** – 将事务日志中队列 103:1 的所有已恢复事务转储到 RSSD 中：

```
sysadmin sqt_dump_queue, 103,1, 0, RSSD
```

- **示例 11** – 将事务日志中队列 103:1 的所有已恢复事务转储到客户端：

```
sysadmin sqt_dump_queue, 103,1, 0, client
```

用法

- 使用 **sysadmin sqt_dump_queue** 之前，先执行 **admin who, sqt** 以确保此数据库的事务高速缓存存在。
- 此命令转储事务高速缓存中事务的所有语句。
- **sysadmin sqt_dump_queue** 将事务语句转储到以下位置之一：
  - Replication Server 日志
  - 备用日志文件
  - RSSD
  - 发出此命令的客户端

  要将事务转储到 RSSD 或客户端，**sysadmin sqt_dump_queue** 的最后一个参数必须为 **RSSD** 或 **client**。

  如果通过 **sysadmin dump_file** 命令或者 *file_name* 选项指定用于转储事务的备用日志文件，则输出到此备用转储文件中。

如果未指定 **RSSD** 或 **client** 选项，或者指定了 **log** 选项，则会输出到 Replication Server 日志中。

- **sysadmin sqt_dump_queue** 的输出指出事务高速缓存中事务的状态为已打开、已关闭或已读取。打开的事务是尚未提交的事务。已关闭的事务已提交，但尚未完全读取。已读取的事务已完全读取，当尚未删除。
- 通过设置配置参数 **sqt_max_cache_size** 可以修改高速缓存的大小。

权限

sysadmin sqt_dump_queue 需要“sa”权限。

另请参见

- admin who （第 88 页）
- sysadmin dump_file （第 349 页）

sysadmin system_version

显示或设置复制系统的系统范围版本号，以便使用相应版本级别中的软件功能。

从版本 11.5 开始，各个 Replication Server 的节点版本也启用一些新功能。系统版本号不需要与当前的软件版本相对应。

语法

```
sysadmin system_version [, version]
```

参数

- **version** – 用于复制系统的系统版本号。

示例

- **示例 1** – 在 ID Server 上执行时，显示当前的系统版本号：

  ```
  sysadmin system_version
  ```

- **示例 2** – 在 ID Server 上执行时，将系统版本号更改为对应于版本 15.1。在以下情况下可以使用这个版本号：

  - 所有 Replication Server 的版本均为 15.1
  - 不需要将任何 Replication Server 降级为更低版本
  - 不需要安装任何更低版本的 Replication Server

  ```
  sysadmin system_version, 1510
  ```

用法

- 要设置系统版本号，请在 ID Server 上执行带 *version* 参数的 **sysadmin system_version**。
  - 输入的系统版本号不得高于复制系统中任何 Replication Server 的最低软件版本号（即 Replication Server 的版本级别）。
  - 在 ID Server 以外的任何其他 Replication Server 上都无法设置系统版本号。
- 要显示当前的系统版本号，请在 ID Server 上执行不带 *version* 参数的 **sysadmin system_version**。
  如果在其它 Replication Server 上执行此命令，Replication Server 将尝试联络 ID Server 以确定当前系统版本号。在极少数情况下，Replication Server 可能无法联络 ID Server。因此，只能保证 ID Server 上的值是正确的。

系统版本与节点版本

- 从 Replication Server 版本 11.5 开始，如果将 Replication Server 的节点版本号设置为当前软件版本（例如，对应版本 15.1 设置为 1510），就能使用某些新的软件功能。有关详细信息，请参见 **sysadmin site_version**。
  同时要求最低系统版本号为 1102。
- 如果安装 Replication Server 版本 11.5 或更高版本作为新复制系统的 ID Server，应将系统版本号设置为 1102。这样您就可以在系统中安装版本 11.0.2 或更高版本的其它 Replication Server。
- 有关安装或升级 Replication Server 的详细信息，请参见适用于您的平台的 Replication Server 安装指南和配置指南。

混合版本复制系统

如果所有 Replication Server 均为版本 11.0.2 或更高版本，则系统版本号要求的最高设置为 1102。将系统版本号设置为 1102 之后，再也无须重置它了。

单个 Replication Server 的 1102 系统版本号和节点版本号允许混合版本的复制系统，在该系统中，节点版本不同的 Replication Server 可以一起运行。每个 Replication Server 都可以使用它的版本所包含的全部可用功能。

在混合版本复制系统中，有些功能只适用于具有较高节点版本的 Replication Server。例如，主 Replication Server 及其某个复制 Replication Server 的节点版本为 1510，而它的其它复制 Replication Server 的节点版本为 1260。当表复制定义具有 *timestamp* 列时，节点版本较低的复制 Replication Server 只能将 *timestamp* 作为 *varbinary* (8) 预订，而节点版本为 1510 的复制 Replication Server 可以直接预订 *timestamp* 列。有关详细信息，请参见 sysadmin site_version。

有关特定 Replication Server 软件版本中引入的功能的详细信息，请参见适用于该版本的《Replication Server 新增功能指南》。

系统版本与 ID Server

在除 ID Server 以外的 Replication Server 中，如果执行某个要求某最低系统版本的命令，Replication Server 将联系 ID Server 以确定当前系统版本号，然后才允许使用此命令。

有关版本信息的系统表

版本信息存储在 *rs_version* 系统表中。*rs_routes* 系统表中也包含版本信息。

权限

sysadmin system_version 需要“sa”权限。

另请参见

- admin version （第 86 页）
- sysadmin site_version （第 372 页）

sysadmin upgrade, "database"

升级由 Replication Server 提供服务的用户数据库。

语法

```
sysadmin upgrade, "database"{, data_server,database | all}
```

参数

- ***data_server*、*database*** – 指定要升级的数据库。为每个数据库输入单独的命令。
- **all** – 升级由 Replication Server 提供服务的所有数据库。如果数据库不符合升级条件，Replication Server 将显示错误消息

示例

- **示例 1** – 升级 pds 数据服务器中的 pdb01 数据库。在为 pdb01 提供服务的 Replication Server 上输入：

  ```
  sysadmin upgrade, database, pds, pdb01
  ```

  如果任何数据库无法升级，请查看 Replication Server 错误日志中的条目，例如：

  ```
  Database is not accessible.
  Fail to upgrade data_server.database.
  ```

用法

- 在升级的 Replication Server 中输入 **admin version, “connection”** 以找出必须升级的用户数据库。

- 在将 Replication Server 升级到 15.7.1 或更高版本后，Replication Server 挂起与 Sybase IQ 复制的复制连接；如果您使用 **admin who**，则会看到“等待升级” (Awaiting Upgr) 状态。使用 **sysadmin upgrade, "database"** 升级 Sybase IQ 数据库。

请参见《配置指南》中的“Fixing a Failed or Missed User Database Upgrade with sysadmin upgrade, "database"”（使用 sysadmin upgrade, "database" 修复失败或忽略的用户数据库升级）。

权限

sysadmin upgrade, "database" 需要“sa”权限。

另请参见

- admin version, "connection" （第 86 页）
- admin who （第 88 页）
- repserver （第 553 页）

sysadmin upgrade, route

将路由从当前 Replication Server 升级到目标 Replication Server 并恢复任何失败的升级路由。

语法

```
sysadmin upgrade, "route", dest_rs [,"recovery"]
```

参数

- ***dest_rs*** – 目标 Replication Server。
- ***recovery*** – 路由升级失败时恢复路由升级。

示例

- **示例 1** – 将路由从 NY_RS Replication Server 升级到 LON_RS Replication Server。在 NY_RS 上运行以下命令：

  ```
  sysadmin upgrade, "route", LON_RS
  ```

  您会看到：

  ```
  Route upgrade for route 'NY_RS.LON_RS' is in progress in the
  background.
  ```

- **示例 2** – 恢复失败的路由升级。在 NY_RS 上运行以下命令：

  ```
  sysadmin upgrade,"route", LON_RS, "recovery"
  ```

用法

- 使用 **sysadmin upgrade, route, dest_rs** 命令可将路由从当前 Replication Server 升级到目标 Replication Server，其中 `dest_rs` 是目标 Replication Server 名称。
- 如果路由升级失败，则使用 **recovery** 选项可将路由升级恢复为上次升级状态。
- 用于运行该命令的用户 ID 和口令还必须在目标 Replication Server 以及目标 Replication Server 的 RSSD 中存在。

请参见《配置指南》中的“升级路由”。

权限

sysadmin upgrade, route 需要在目标 Replication Server 上具有“sa”权限，在目标 Replication Server 的 RSSD 上具有“dba”权限。

validate publication

将发布的状态设置为“VALID”，以便为此发布创建新的预订。

语法

```
validate publication pub_name
with primary at data_server.database
```

参数

- **pub_name** – 要验证的发布的名称。
- **with primary at data_server.database** – 指定主数据的位置。如果主数据库是热备份应用程序的一部分，则 *data_server.database* 是逻辑数据服务器和数据库的名称。

示例

- **示例 1** – 验证发布 *pubs2_pub*:

```
validate publication pubs2_pub
    with primary at TOKYO_DS.pubs2
```

用法

- 创建了某发布的所有项目后，必须先使用 **validate publication** 验证此发布，复制节点才能对其进行预订。在验证发布的过程中，将检验发布是否至少包含一个项目，然后将此发布标记为可以预订。
- 在使用 **create publication** 创建发布的 Replication Server 上执行 **validate publication**。
- 要检查发布的状态，请使用 **check publication**。此命令显示发布包含的项目数，并指出此发布是否有效。

有关预订实现的详细信息，请参见《Replication Server 管理指南第一卷和第二卷》。

权限

validate publication 需要“create object”权限。

另请参见

- check publication （第 176 页）
- check subscription （第 177 页）
- create publication （第 254 页）
- create subscription （第 280 页）
- define subscription （第 290 页）
- drop publication （第 307 页）

validate subscription

对于复制定义或发布的预订，将预订状态设置为“VALID”。此命令是批量实现进程的一部分，或刷新发布预订进程的一部分。

语法

```
validate subscription sub_name
for {table_rep_def | function_rep_def |
      publication pub_name
      with primary at data_server.database}
with replicate at data_server.database
```

参数

- **sub_name** – 要验证的预订的名称。
- **for table_rep_def** – 指定所预订的表复制定义的名称。
- **for function_rep_def** – 指定所预订的函数复制定义的名称。
- **for publication pub_name** – 指定所预订的发布的名称。
- **with primary at data_server.database** – 指定主数据的位置。如果主数据库是热备份应用程序的一部分，则 *data_server.database* 是逻辑数据服务器和数据库的名称。只有对发布的预订才使用此子句。
- **with replicate at data_server.database** – 指定复制数据的位置。如果复制数据库是热备份应用程序的一部分，则 *data_server.database* 是逻辑数据服务器和数据库的名称。

示例

- **示例 1** – 验证对表复制定义 *titles_rep* 的预订 *titles_sub*，其中复制数据库是 SYDNEY_DS.*pubs2*:

```
validate subscription titles_sub
 for titles_rep
 with replicate at SYDNEY_DS.pubs2
```

- **示例 2** – 验证对函数复制定义 *myproc_rep* 的预订 *myproc_sub*，其中复制数据库是 SYDNEY_DS.*pubs2*:

```
validate subscription myproc_sub
 for myproc_rep
 with replicate at SYDNEY_DS.pubs2
```

- **示例 3** – 验证对发布 *pubs2_pub* 的预订 *pubs2_sub*，其中主数据库是 TOKYO_DS.*pubs2*，复制数据库是 SYDNEY_DS.*pubs2*:

```
validate subscription pubs2_sub
 for publication pubs2_pub
 with primary at TOKYO_DS.pubs2
 with replicate at SYDNEY_DS.pubs2
```

用法

- 使用 **validate subscription** 在主 Replication Server 和复制 Replication Server 上验证预订。预订的对象可以是表复制定义、函数复制定义或发布。
- 此命令完成批量实现进程。第一步是使用 **define subscription** 创建预订。第二步是使用 **activate subscription** 激活预订。
- 如果向当前具有预订的发布中添加了一些新项目，要创建对这些项目的新预订，必须刷新此发布预订。

  使用 **define subscription** 和 **activate subscription** 在发布预订中创建并激活新项目预订。然后为新项目预订手动装载预订数据，并使用 **validate subscription** 来验证发布预订。
- 在使用 **define subscription** 创建预订的 Replication Server 上执行 **validate subscription**。
- 验证发布预订时，其中所有的项目也被同时验证。
- **validate subscription** 将预订的状态从“ACTIVE”更改为“VALID”。此后在主数据服务器上进行的更新将通过主 Replication Server 分发，并应用于复制 Replication Server。
- 此命令修改多个节点上的 RSSD 表。在主 Replication Server 和复制 Replication Server 上均使用 **check subscription** 查看对每个预订的影响。

有关预订实现的详细信息，请参见《Replication Server 管理指南第一卷和第二卷》。

权限

在复制数据的节点上，**validate subscription** 需要“create object”权限；而在存储主数据的节点上，则需要“primary subscribe”或“create object”权限。

另请参见

- activate subscription （第 44 页）

- check subscription （第 177 页）
- create article （第 211 页）
- create publication （第 254 页）
- create subscription （第 280 页）
- define subscription （第 290 页）
- drop subscription （第 312 页）

wait for create standby

一个阻塞命令，使 Replication Server 中的客户端会话等待备用数据库创建进程完成。

语法

```
wait for create standby
 for logical_ds.logical_db
```

参数

- **logical_ds** – 逻辑连接的数据服务器名称。
- **logical_db** – 逻辑连接的数据库名称。

用法

- 创建备用数据库后，**wait for create standby** 显示状态信息。
- **wait for create standby** 可能非常适于在脚本中使用。

权限

wait for create standby 需要“sa”权限。

另请参见

- abort switch （第 43 页）
- switch active （第 338 页）
- wait for switch （第 393 页）

wait for delay

指定阻塞此命令的时间间隔。

语法

```
wait for delay ’ time_string’
```

参数

- **time_string** – 执行前等待的一段时间。请使用格式 hh:mm[:ss[.xxx]] [am|pm]。

示例

- **示例 1** – 此命令指示 Replication Server 将命令阻塞 1 小时 30 分钟：

```
wait for delay '01:30'
```

用法

- 使用 **wait for delay** 指示 Replication Server 等待指定的一段时间。一种典型用途是实现预订。通常，在两个预订之间发出 **wait for delay**。
- 指定的时间可以包括小时、分钟和秒，最多为 24 小时。

权限

任何用户都可以执行此命令。

另请参见

- wait for time （第 394 页）

wait for switch

一个阻塞命令，使 Replication Server 中的客户端会话等待切换至新活动数据库的操作完成。

语法

```
wait for switch
 for logical_ds.logical_db
```

参数

- **logical_ds** – 逻辑连接的数据服务器名称。
- **logical_db** – 逻辑连接的数据库名称。

用法

- 在 **switch active** 操作完成后，**wait for switch** 显示状态信息。
- **wait for switch** 可能非常适于在脚本中使用。

权限

wait for switch 需要“sa”权限。

另请参见

- abort switch （第 43 页）
- switch active （第 338 页）
- wait for create standby （第 392 页）

wait for time

指定一天中取消阻塞此命令的时间。

语法

```
wait for time ’ time_string’
```

参数

- **time_string** – 特定执行时间。请使用格式 hh:mm[:ss[.xxx]] [am|pm]。

示例

- **示例 1** – 此命令指示 Replication Server 一直等到下午 5:30：

  ```
  wait for time '05:30 pm'
  ```

用法

- 使用 **wait for time** 指示 Replication Server 一直等到指定的时间。
- 指定的时间可以包括小时、分钟和秒，最多为 24 小时。
  如果当前时间为下午 6:00，**wait for time '5:00 pm'** 指示等到明天下午 5:00。

权限

任何用户都可以执行此命令。

另请参见

- wait for delay （第 392 页）

Replication Server 系统函数

提供 Replication Server 系统函数的参考页。

有关自定义系统函数的函数字符串的信息，请参见《Replication Server 管理指南第二卷》。

介绍的系统函数可能具有*函数字符串类作用域*或*复制定义作用域*。

具有函数字符串类范围的函数仅为函数字符串类定义一次。之后，按同样的方式将其应用到该类被指派到的每个数据库。

具有复制定义范围的函数要为各个复制定义分别定义一次。之后，按相同的方式将其应用于使用该复制定义复制的每个操作（更新、插入等）。

rs_autoc_on

更新 `rs_status` 表以指示自动更正设置为 on。

当数据服务器接口 (DSI) 在主数据库日志中遇到 **autocorrection on** 记录时，Replication Server 会调用 **rs_autoc_on**。

示例

- **示例** – 为 **rs_iq_function_class** 创建 **rs_autoc_on** 函数字符串。

```
create function string rs_autoc_on
 for rs_iq_function_class
 output language
  'insert into rs_status (schema, tablename, action, starttime,
status) values
    (?rs_repl_objowner!sys?,
     ?rs_deliver_as_name!sys?,
     "A",
     current timestamp,
     "P");
  commit'
```

用法

- **rs_autoc_on** 函数具有函数字符串类作用域。
- Replication Server 在安装期间创建一个初始 **rs_autoc_on** 函数字符串。
- **rs_autoc_on** 使用系统定义的变量 *rs_deliver_as_name*，用以指示复制数据库中被自动更正影响的表。

- **rs_autoc_on** 使用系统定义的变量 *rs_repl_objowner*，用以指示复制数据库中被自动更正影响的表的所有者。如果未指定所有者，**rs_repl_objowner** 会包含一个空格。

rs_autoc_off

更新 rs_status 表以指示自动更正设置为 off。

当 Replication Server 在主数据库日志中遇到 **autocorrection off** 记录时，它会调用 **rs_autoc_off**。

示例

- 示例 – 为 **rs_iq_function_class** 创建 **rs_autoc_off** 函数字符串。

```
create function string rs_autoc_off
 for rs_iq_function_class
 output language
  'update rs_status
   set endtime = current timestamp,
   status = "X" where schema = ?rs_repl_objowner!sys?
   and tablename = ?rs_deliver_as_name!sys?
   and action = "A" and endtime is null;
   insert into rs_status (schema, tablename, action, starttime,
status) values
   (?rs_repl_objowner!sys?,
    ?rs_deliver_as_name!sys?,
    "R",
    current timestamp,
    "P");
   commit'
```

用法

- **rs_autoc_off** 函数具有函数字符串类作用域。
- Replication Server 在安装期间创建一个初始 **rs_autoc_off** 函数字符串。
- **rs_autoc_off** 使用系统定义的变量 *rs_deliver_as_name*，用以指示复制数据库中被自动更正影响的表。
- **rs_autoc_off** 使用系统定义的变量 *rs_repl_objowner*，用以指示复制数据库中被自动更正影响的表的所有者。如果未指定所有者，**rs_repl_objowner** 会包含一个空格。

rs_autoc_ignore

更新 rs_status 表以指示自动更正失败而且针对该表忽略 DML。

在自动更新期间进行主键更新时，Replication Server 会调用 **rs_autoc_ignore**。

示例

- **示例** – 为 **rs_iq_function_class** 创建 **rs_autoc_ignore** 函数字符串。

```
create function string rs_autoc_ignore
 for rs_iq_function_class
 output language
  'update rs_status
   set endtime = current timestamp,
   status = 'E' where schema = ?rs_repl_objowner!sys?
   and tablename = ?rs_deliver_as_name!sys?
   and action = 'A' and endtime is null;
   commit'
```

用法

- **rs_autoc_ignore** 函数具有函数字符串类作用域。
- Replication Server 在安装期间创建一个初始 **rs_autoc_ignore** 函数字符串。
- **rs_autoc_ignore** 使用系统定义的变量 *rs_deliver_as_name*，用以指示复制数据库中被自动更正影响的表。
- **rs_autoc_ignore** 使用系统定义的变量 *rs_repl_objowner*，用以指示复制数据库中被自动更正影响的表的所有者。如果未指定所有者，**rs_repl_objowner** 会包含一个空格。

rs_batch_end

用户可以使用 **rs_batch_end** 将批处理命令发送到非 Adaptive Server 数据库服务器。此函数字符串存储标记批处理命令结尾所需的 SQL 语句。

示例

- **示例 1** – 更改 **rs_batch_end** 函数字符串以使 **sqlserver_derived_class** 函数字符串类的 SQL 输出为 `END`。

```
alter function string publishers.rs_batch_end
for sqlserver_derived_class
output language
'END'
```

用法

- **rs_batch_end** 函数具有函数字符串类作用域。
- 此函数字符串与 **rs_batch_start** 一起使用。
- 将 **rs_batch_end** 作为批处理命令中的最后一个命令发送到复制数据服务器。只有在将 **use_batch_markers** 设置为 on 时，才会发送该函数字符串。
- 在数据服务器处理顺序中，**rs_batch_end** 在 **rs_commit** 之前。

- 如果由于所刷新的命令受 **dsi_cmd_batch_size** 等限制而需要多个批处理命令，则可以为给定事务重复执行 **rs_batch_start**、批处理命令和 **rs_batch_end**。

另请参见

- rs_batch_start （第 398 页）

rs_batch_start

用户可以使用 **rs_batch_start** 将批处理命令发送到非 Adaptive Server 数据库服务器。此函数字符串存储标记批处理命令开头所需的 SQL 语句。

示例

- **示例 1** – 更改 **rs_batch_start** 函数字符串以使 **sqlserver_derived_class** 函数字符串类的 SQL 输出为 `BEGIN`。

```
alter function string publishers.rs_batch_start
for sqlserver_derived_class
output language
'BEGIN'
```

用法

- **rs_batch_start** 函数具有函数字符串类作用域。
- 对于 Adaptive Server 或任何其它通过函数字符串 **rs_begin** 和 **rs_commit** 支持命令批处理的数据服务器，不必使用 **rs_batch_start**。
- 仅当 **use_batch_markers** 设置为 on 时，才会将 **rs_batch_start** 及其后面的批处理命令发送至复制数据服务器。**rs_batch_start** 在 **rs_begin** 之后发送。
- Replication Server 不在 **rs_batch_start** 后面使用命令分隔符。如果复制数据库服务器需要在批处理开始标记后面使用命令分隔符，则会将其作为 **rs_batch_start** 字符串的一部分。此分隔符不论是否与 **dsi_cmd_separator** 参数相同，都必须将其作为函数字符串的一部分。
- 如果由于所刷新的命令受 **dsi_cmd_batch_size** 等限制而需要多个批处理命令，则可以重复执行 **rs_batch_start**、批处理命令和 **rs_batch_end**。

另请参见

- rs_batch_end （第 397 页）

rs_begin

开始数据服务器中的事务。

示例

- **示例 1** – 为 *oth_sql_class* 函数字符串类创建 **rs_begin** 函数字符串。如果事务没有名称，则 *rs_origin_xact_name* 系统变量的值为 null 值。在系统变量之前放置“t_”可以防止数据服务器语法错误，并允许函数字符串支持命名的和未命名的事务。

```
alter function string rs_begin
 for oth_sql_class
 output language
 'begin transaction
   t_?rs_origin_xact_name!sys_raw?'
```

- **示例 2** – 为不支持 **begin transaction** 操作的数据服务器的函数字符串类创建 **rs_begin** 函数字符串。

```
create function string rs_begin
 for oth_sql_class
 output language ''
```

用法

- **rs_begin** 函数具有函数字符串类作用域。
- Replication Server 会在安装过程中为系统提供的函数字符串类创建一个初始 **rs_begin** 函数字符串。
- 如果使用用户创建的基本函数字符串类，则必须创建 **rs_begin** 函数字符串。
- 在作为函数字符串类主节点的 Replication Server 上创建或自定义 **rs_begin** 函数字符串。
- 某些数据服务器不支持显式的 **begin transaction** 操作。每当提交或回退上一个事务时，它们便隐式地开始事务。对于这些数据服务器，**rs_begin** 函数字符串可以是空字符串 ('')。
- 此函数的函数字符串通常使用 *rs_origin_xact_name* 系统变量。其值是从 RepAgent 接收的。事务的名称是在 Transact-SQL 中使用 **begin transaction** 指派的。

另请参见

- alter function string （第 143 页）
- create function string （第 236 页）
- rs_commit （第 400 页）
- rs_rollback （第 424 页）

rs_check_repl

检查表是否被标记为复制。

示例

- **示例 1** – 创建 **rs_check_repl** 函数字符串，用于执行 **rs_check_repl_stat** 存储过程。

```
create function string rs_check_repl
 for sqlserver_derived_class
 output language
 'execute rs_check_repl_stat
  @rs_repl_name = ?rs_repl_name!param?'
```

用法

- **rs_check_repl** 函数具有函数字符串类作用域。
- Replication Server 会在安装过程中为系统提供的函数字符串类创建一个初始 **rs_check_repl** 函数字符串。
- 如果使用用户创建的基本函数字符串类，则必须创建 **rs_check_repl** 函数字符串。
- 在作为函数字符串类主节点的 Replication Server 上创建或自定义 **rs_check_repl** 函数字符串。

另请参见

- create function string （第 236 页）
- create replication definition （第 258 页）

rs_commit

提交数据服务器中的事务。

示例

- **示例 1** – 此示例说明了 *rs_sqlserver_function_class* 和 *rs_default_function_class* 类的缺省 **rs_commit** 函数字符串。该函数字符串执行名为 **rs_update_lastcommit** 的存储过程，然后执行 Transact-SQL **commit transaction** 命令。

```
create function string rs_commit
 for sqlserver_derived_class
 output language
 'execute rs_update_lastcommit
   @origin = ?rs_origin!sys?,
   @origin_qid = ?rs_origin_qid!sys?,
   @secondary_qid = ?rs_secondary_qid!sys?,
```

```
   @origin_time = ?rs_origin_commit_time!sys?;
commit transaction'
```

以下是 *rs_sqlserver_function_class* 的 **rs_update_lastcommit** 过程的文本：

```
/* Create a procedure to update the
 ** rs_lastcommit table. */
create procedure rs_update_lastcommit
    @origin  int,
    @origin_qid  binary(36),
    @secondary_qid  binary(36),
    @origin_time  datetime
as
begin
    update rs_lastcommit
        set origin_qid = @origin_qid,
        secondary_qid = @secondary_qid,
        origin_time = @origin_time,
        commit_time = getdate()
    where origin = @origin
    if (@@rowcount = 0)
    begin
        insert rs_lastcommit (origin,
            origin_qid, secondary_qid,
            origin_time, commit_time,
            pad1, pad2, pad3, pad4,
            pad5, pad6, pad7, pad8)
            values (@origin, @origin_qid,
            @secondary_qid,@origin_time,
            getdate(), 0x00, 0x00, 0x00,
            0x00, 0x00, 0x00, 0x00, 0x00)
    end
end
```

用法

- **rs_commit** 函数具有函数字符串类作用域。
- Replication Server 会在安装过程中为系统提供的函数字符串类创建一个初始 **rs_commit** 函数字符串。
- 如果使用用户创建的基本函数字符串类，则必须创建 **rs_commit** 函数字符串。
- 在作为函数字符串类主节点的 Replication Server 上创建或自定义 **rs_commit** 函数字符串。
- 在 **rs_commit** 函数字符串中更新 *rs_lastcommit* 系统表。在事务内更新此表可以维护数据完整性。

警告！ 如果没能为提交的每个事务正确地更新 *rs_lastcommit* 系统表，Replication Server 在重新启动后可能会多次应用事务或跳过事务。

另请参见

- alter function string （第 143 页）
- create function string （第 236 页）

- rs_begin （第 399 页）
- rs_get_lastcommit （第 412 页）
- rs_rollback （第 424 页）

rs_datarow_for_writetext

提供与 *text*、*unitext* 或 *image* 列关联的数据行的映像，这些列由 Transact-SQL **writetext** 命令、Client-Library 函数 **ct_send_data** 或 DB-Library™ 函数 **dbwritetext** 和 **dbmoretext** 更新。

示例

- **示例 1** – 执行名为 **capture_datarow** 的存储过程，将 *@au_id* 的值设置为 *au_id* 列的值，并将 *@copy* 的值设置为 *copy* 列的状态值。

```
create function string
 blurbs_rep.rs_datarow_for_writetext
 for sqlserver_derived_class
 output rpc
 'execute capture_datarow
   @au_id = ?au_id!new?,
   @copy = ?copy!text_status?'
```

用法

- Replication Server 先执行 **rs_datarow_for_writetext**，然后将更新的 *text*、*unitext* 或 *image* 数据发送到复制数据服务器。**rs_datarow_for_writetext** 提供行的主键列和可搜索列的值，以便能够处理预订并将数据传送到复制数据库。
- **rs_datarow_for_writetext** 可访问行中除 *text*、*unitext* 和 *image* 列以外的所有列的值。要检索有关 *text*、*unitext* 或 *image* 列的信息，请在函数字符串中包括 *text_status* 修饰符。“text、unitext 和 image 数据的 text_status 值”表中介绍了由 *text_status* 返回的值。
- **rs_datarow_for_writetext** 函数具有复制定义范围。
- 创建复制定义时，Replication Server 为 *rs_sqlserver_function_class* 和 *rs_default_function_class* 生成 **rs_datarow_for_writetext** 函数字符串。
- 如果使用用户创建的基本函数字符串类，则必须为包括 *text*、*unitext* 和 *image* 列的每个复制定义创建 **rs_datarow_for_writetext** 函数字符串。
- 在创建复制定义所在的 Replication Server 上创建或自定义 **rs_datarow_for_writetext** 函数字符串。
- 为 *rs_sqlserver_function_class* 和 *rs_default_function_class* 缺省生成的函数字符串不执行复制数据库中的命令，这是因为行图像中不包含修改的数据。
- 您可以创建新的 **rs_datarow_for_writetext** 函数字符串来收集将传递给网关的主键值。*old* 和 *new* 修饰符都可以提供对列值的访问。

- *text_status* 修饰符检索 *text*、*unitext* 或 *image* 列的状态。表 35. text、unitext 和 image 数据的 text_status 值 （第 403 页）列出 *text_status* 修饰符的可能值。

表 35. `text`、`unitext` 和 `image` 数据的 `text_status` 值

值	说明
0x0001	列的文本指针为空。没有对 *text*、*unitext* 或 *image* 列进行修改。
0x0002	在主数据库中进行了修改，从而导致分配文本指针。Replication Server 执行 **rs_textptr_init** 函数来分配文本指针。
0x0004	后面接当前数据值。Replication Server 执行 **rs_writetext** 函数以修改复制数据库中的 *text*、*unitext* 或 *image* 数据。
0x0008	未复制 *text*、*unitext* 或 *image* 列。不需要在复制数据库中执行任何命令，原因是数据没有更改值，并且 *text*、*unitext* 或 *image* 列的状态为 **replicate_if_changed**。
0x0010	在主数据库中执行操作后，*text*、*unitext* 或 *image* 列包含 null 值。例如，在分配文本指针后，*text* 或 *image* 列中可能包含数据值，而主数据库中的应用程序将这些值设为 null。如果 *text_status* 不是 0x0008，Replication Server 可通过将值设置为 null 来截断复制数据库中的 *text*、*unitext* 或 *image* 列。

另请参见

- rs_get_textptr （第 414 页）
- rs_textptr_init （第 433 页）
- rs_writetext （第 441 页）

rs_delete

删除被复制的表中的行。

示例

- **示例 1** – 为 *titles_rep* 复制定义更改 **rs_delete** 函数字符串，以便它能够执行名为 **del_title** 的存储过程。

```
alter function string titles_rep.rs_delete
 for sqlserver_derived_class
 output rpc
 'execute del_title
   @title=?title!old?'
```

用法

- Replication Server 执行 **rs_delete** 来删除一个表中的单个行。该行由在复制定义中为该表定义的主键列标识。

- **rs_delete** 具有复制定义范围。
- 在创建复制定义时，Replication Server 为系统提供的函数字符串类生成一个 **rs_delete** 函数字符串。
- 如果使用用户创建的基本函数字符串类，则必须为每个复制定义创建 **rs_delete** 函数字符串。
- 请在创建复制定义的位置创建或自定义 **rs_delete** 函数字符串。
- 对于系统提供的类 *rs_sqlserver_function_class* 和 *rs_default_function_class*，**rs_delete** 生成的函数字符串使用 Transact-SQL **delete** 命令语法。要删除的行使用 **where** 子句来标识，该子句指定主键列的删除前值或前像 (before image)。

另请参见

- create function string （第 236 页）
- create replication definition （第 258 页）
- rs_insert （第 418 页）
- rs_update （第 438 页）

rs_dsi_check_thread_lock

确定 DSI 执行程序线程是否持有阻塞复制数据库进程的锁。如果返回值大于 0，表明线程占用另一个数据库进程所需的资源，而且此线程应该回退和重试事务。

示例

- **示例 1** – 创建 **rs_dsi_check_thread_lock** 函数字符串，用于检查当前的 DSI 执行程序线程是否正在阻止另一个复制数据库进程。

```
create function string rs_dsi_check_thread_lock
for sqlserver_derived_class
output language
'select count(*) as seq from master..sysprocesses
where blocked = @@spid and suid = suser_id()'
```

用法

- Replication Server 使用 **rs_dsi_check_thread_lock** 函数检查当前的 DSI 执行程序线程是否正在阻止另一个复制数据库进程。只有在为 **dsi_commit_control** 设置为 on 的连接定义了多个 DSI 线程，而且一个 DSI 执行程序线程已准备好提交但由于不是“下一个”要提交的线程而无法提交，并且经过了为 **dsi_commit_check_locks_intrvl** 指定的时间的情况下，才会执行此函数。
- 函数字符串 **rs_dsi_check_thread_lock** 查询预期会返回单个整数值，即 *seq* 的列名。如果返回值大于 0，表明线程占用另一个数据库进程所需的资源，而且此线程应该回退和重试事务。
- **rs_dsi_check_thread_lock** 具有函数字符串类作用域。

- Replication Server 会在安装过程中为系统提供的函数字符串类创建一个初始 **rs_dsi_check_thread_lock** 函数字符串。
- 如果您要使用自定义基本函数字符串并且想要使用 **dsi_commit_control** 设置为 on 的并行 DSI，则必须为 **rs_dsi_check_thread_lock** 函数字符串创建一个函数字符串。否则，无需为该函数创建函数字符串。
- 在部署在该类的主节点的 Replication Server 上创建或自定义 **rs_dsi_check_thread_lock** 函数字符串。

rs_dumpdb

初始化协调数据库转储。

示例

- **示例 1** – 创建一个 **rs_dumpdb** 函数字符串，该函数字符串将数据库转储到指定的转储设备，并执行一个过程，以更新 *rs_lastcommit* 系统表。如果只有一个复制数据库，或者使用该函数字符串类的所有数据库具有相同的转储设备名称，该函数字符串的使用效果最好。

```
create function string rs_dumpdb
 for sqlserver_derived_class
 output language
 'dump database ?rs_destination_db!sys_raw?
     to pubs2_dmpdb;
 execute rs_update_lastcommit
     ?rs_origin!sys?,
     ?rs_origin_qid!sys?,
     ?rs_secondary_qid!sys?,
     ?rs_origin_commit_time!sys?'
```

- **示例 2** – 此示例比第一个示例更适合多个节点和生产环境。**dumpdb_proc** 可管理复制节点的备份设备。该过程应当选择一个要使用的备份设备，然后将其标记为“已用”，以便后续的转储不会覆盖前一个备份。

```
alter function string rs_dumpdb
 for sqlserver_derived_class
 output rpc
 'execute dumpdb_proc
     ?rs_dump_dbname!sys?,
     ?rs_dump_label!sys?,
     ?rs_dump_timestamp!sys?,
     ?rs_destination_db!sys?,
     ?rs_origin!sys?,
     ?rs_origin_qid!sys?,
     ?rs_secondary_qid!sys?,
     ?rs_origin_commit_time!sys?'
```

该过程使用 *rs_origin*、*rs_origin_qid* 和 *rs_secondary_qid* 来执行 **rs_update_lastcommit**。如果服务器在转储完成后、在更新 *rs_lastcommit* 系统表之前发生故障，则在 Replication Server 恢复运行后将重新启动备份。

注意： 因为 Adaptive Server 不允许 **dump** 命令与其它命令同时包含在一个事务中，所以无法保证转储和 **rs_update_lastcommit** 过程会自动执行。如果未能成功更新 *rs_lastcommit* 系统表，可能会再执行一次转储。

在以下 **dumpdb_proc** 存储过程样本文本中，转储设备是硬编码的。在生产环境中，最好在表中管理这些设备。

```
create proc dumpdb_proc
        @dump_dbname varchar(30),
        @dump_label varchar(30),
        @dump_timestamp varbinary(16),
        @destination_dbname varchar(30),
        @origin int,
        @origin_qid binary(36),
        @secondary_qid binary(36),
        @origin_time datetime
as
print 'Received a dump database command from Replication Server:'
declare @message varchar(255)
select @message = 'dump database ' + @dump_dbname
        + '. Label= '' + @dump_label
        + ''. Dest.db = '' + @destination_dbname
        + ''''
print @message
if @destination_dbname = 'pubs2'
begin
        print 'issuing ''dump database pubs2.'''
        dump database pubs2 to pubs2_dmplog
        update dmp_count set d_count = d_count + 1
        exec pubs2.dbo.rs_update_lastcommit
            @origin, @origin_qid, @secondary_qid,
            @origin_time
end
else if @destination_dbname = 'pubs3'
begin
        print 'issuing ''dump database pubs3.'''
        dump database pubs3 to pubs3_dmplog
        update dmp_count set d_count = d_count + 1
        exec pubs3.dbo.rs_update_lastcommit
            @origin, @origin_qid, @secondary_qid,
            @origin_time
end
```

用法

- Replication Server 通过将 **rs_dumpdb** 函数调用放置在分发到每个复制 Replication Server 的事务流中的同一位置来协调数据库转储。
- **rs_dumpdb** 具有函数字符串类范围。

注意： Replication Server 不会为系统提供的函数字符串类初始化或生成 **rs_dumpdb** 函数字符串。在 Adaptive Server 上使用协调转储之前，您必须创建函数字符串。

- 在作为函数字符串类主节点的 Replication Server 上创建一个 **rs_dumpdb** 函数字符串。
- 要记录多个复制节点上的不同转储设备，请在每个执行数据库转储的复制数据库中创建一个存储过程。然后编写 **rs_dumpdb** 函数字符串，以执行该存储过程。
- *rs_lastcommit* 系统表应当在执行 **rs_dumpdb** 函数字符串时得到更新，这样，重新启动的 Replication Server 就不会执行重复的转储。有关 *rs_lastcommit* 的信息，请参见“**rs_commit**”。

表 36. rs_dumpdb 函数字符串的系统变量

变量名	数据类型	说明
rs_dump_dbname	*varchar(30)*	转储开始的数据库的名称。
rs_dump_label	*varchar(30)*	关于转储的标签信息。对于 Adaptive Server，此变量保存代表转储开始时间的 *datetime* 值。
rs_dump_timestamp	*varbinary(16)*	开始转储时获取的时间戳。

另请参见

- create function string class （第 249 页）
- rs_commit （第 400 页）
- rs_dumptran （第 407 页）
- rs_get_lastcommit （第 412 页）

rs_dumptran

初始化协调事务转储。

示例

- **示例 1** – 创建一个 **rs_dumptran** 函数字符串，以执行名为 **dumptran_proc** 的存储过程。该存储过程管理转储设备，然后执行 **rs_update_lastcommit** 存储过程，向它传递 *rs_origin*、*rs_origin_qid*、*-rs_secondary_qid* 和 *rs_origin_commit_time* 参数。

```
create function string rs_dumptran
 for sqlserver_derived_class
 output rpc
 'execute dumptran_proc
     ?rs_dump_dbname!sys?,
     ?rs_dump_label!sys?,
     ?rs_dump_timestamp!sys?,
     ?rs_dump_status!sys?,
     ?rs_destination_db!sys?,
     ?rs_origin!sys?,
     ?rs_origin_qid!sys?,
     ?rs_secondary_qid!sys?
     ?rs_origin_commit_time!sys?'
```

如果服务器在转储完成之后但在更新 *rs_lastcommit* 系统表之前崩溃，Replication Server 将重新启动备份服务器。

注意： 因为 Adaptive Server 不允许 **dump** 命令与其它命令同时包含在一个事务中，所以无法保证 **rs_update_lastcommit** 过程会自动执行。如果未能成功更新 *rs_lastcommit* 系统表，可能会再执行一次转储。

在下面的 **dumptran_proc** 存储过程示例文本中，转储设备是硬编码的。在生产环境中，最好采用表的形式来管理这些设备：

```
create proc dumptran_proc
   @dump_dbname varchar(30),
   @dump_label varchar(30),
   @dump_timestamp varbinary(16),
   @dump_status int,
   @destination_dbname varchar(30),
   @origin int,
   @origin_qid binary(36),
   @secondary_qid binary(36),
   @origin_time datetime
as
   print 'Received a dump transaction command from Replication
Server:'
   declare @message varchar(255)
   if @dump_status = 0
   begin
      select @message = 'dump transaction ' + @dump_dbname + '.
Label= '''
         + @dump_label + '''' + '. Dest.db = ''' +
@destination_dbname + ''''
   end
   else if @dump_status = 1
   begin
      select @message = 'dump transaction standby '
         + @dump_dbname + '. Label= ''' +
        @dump_label + '''' + '. Dest.db = ''' + @destination_dbname
+ ''''
   end
   print @message
   if @destination_dbname = 'pubs2'
   begin
      print 'issuing ''dump transaction pubs2.'''
      if @dump_status = 0
      begin
         dump transaction pubs2 to pubs2_dmplog
      end
      else if @dump_status = 1
      begin
        dump transaction pubs2 to pubs2_dmplog with standby_access
      end
      update dmp_count set d_count = d_count + 1
      exec pubs2.dbo.rs_update_lastcommit
         @origin, @origin_qid, @secondary_qid,
         @origin_time
   end
```

```
   else if @destination_dbname = 'pubs3'
   begin
      print 'issuing ''dump transaction pubs3.'''
      if @dump_status = 0
      begin
         dump transaction pubs3 to pubs3_dmplog
      end
      else if @dump_status = 1
      begin
        dump transaction pubs3 to pubs3_dmplog with standby_access
      end
      update dmp_count set d_count = d_count + 1
      exec pubs3.dbo.rs_update_lastcommit
         @origin, @origin_qid, @secondary_qid,
         @origin_time
   end
```

- **示例 2** – 请更改在第一个示例中创建的 **rs_dumptran** 函数字符串，使其作为远程过程调用来执行。

```
alter function string rs_dumptran
 for sqlserver_derived_class
 output rpc
 'execute dumptran_proc
     ?rs_dump_dbname!sys?,
     ?rs_dump_label!sys?,
     ?rs_dump_timestamp!sys?,
     ?rs_dump_status!sys?,
     ?rs_destination_db!sys?,
     ?rs_origin!sys?,
     ?rs_origin_qid!sys?,
     ?rs_secondary_qid!sys?,
     ?rs_origin_commit_time!sys?!'
```

用法

- Replication Server 通过将 **rs_dumptran** 函数调用插入分发到所有复制 Replication Server 的事务流中的同一位置来协调事务转储。
- **rs_dumptran** 具有函数字符串类范围。

注意： Replication Server 不会为系统提供的函数字符串类初始化或生成 **rs_dumptran** 函数字符串。在 Adaptive Server 上使用协调转储之前，您必须创建函数字符串。

- 在作为函数字符串类主节点的 Replication Server 上创建一个 **rs_dumptran** 函数字符串。
- *rs_lastcommit* 系统表应当在执行 **rs_dumptran** 函数字符串时得到更新，这样，重新启动的 Replication Server 就不会执行重复的转储。有关 *rs_lastcommit* 的信息，请参见“**rs_commit**”。
- 要记录多个复制节点上的不同转储设备，请在每个执行事务转储的复制数据库中创建一个存储过程，然后编写 **rs_dumptran** 函数字符串，以执行该存储过程。

表 37. rs_dumptran 函数字符串的系统变量

变量名	数据类型	说明
rs_destination_db	*varchar(30)*	从中发送事务的数据库的名称。
rs_dump_dbname	*varchar(30)*	转储开始的数据库的名称。
rs_dump_label	*varchar(30)*	关于转储的标签信息。对于 Adaptive Server，此变量包含代表转储开始时间的 *datetime* 值。
rs_dump_status	*int(4)*	转储状态指示符： • 0 – 表示 dump transaction 命令不包含 **with standby_access** 参数 • 1 – 表示 dump transaction 命令包含 **with standby_access** 参数
rs_dump_timestamp	*varbinary(16)*	转储在源点数据库启动时获取的 Adaptive Server 数据库时间戳。该变量仅用于提供信息。
rs_origin	*int(4)*	事务的源数据库的 ID。
rs_origin_commit_time	*datetime*	在源提交事务的时间。 **注意：** 当 ASE 仍在处理用户数据库恢复时，如果执行 **select getdate()**，**select getdate()** 返回的值可能与 **rs_origin_begin_time** 值不同。
rs_origin_qid	*varbinary(36)*	事务中第一个命令的源队列 ID。
rs_secondary_qid	*varbinary(36)*	事务在预订实现或取消实现队列中的队列 ID。

另请参见

- create function string （第 236 页）
- rs_commit （第 400 页）
- rs_dumpdb （第 405 页）
- rs_get_lastcommit （第 412 页）

rs_get_charset

返回数据服务器使用的字符集。此函数允许 Replication Server 在字符集与预期不符时输出警告消息。

示例

- **示例 1** – 创建一个带输出语言的 **rs_get_charset** 函数字符串，此函数字符串调用 **sp_serverinfo** 系统过程并返回数据服务器的字符集。

```
create function string rs_get_charset
 for rs_sqlserver2_function_class
 output language
 'sp_serverinfo server_csname'
```

用法

- **rs_get_charset** 获取数据服务器使用的字符集的名称。Replication Server 在每次连接到数据服务器时执行此函数。
- **rs_get_charset** 具有函数字符串类范围。
- Replication Server 会在安装过程中为系统提供的函数字符串类创建一个初始 **rs_get_charset** 函数字符串。
- 如果使用用户创建的基本函数字符串类，则必须创建 **rs_get_charset** 函数字符串。
- 在作为函数字符串类主节点的 Replication Server 上创建或自定义 **rs_get_charset** 函数字符串。
- *rs_sqlserver_function_class* 和 *rs_default_function_class* 类的缺省 **rs_get_charset** 函数字符串使用 *server_csname* 参数调用 Adaptive Server 存储过程 **sp_serverinfo**。
- 数据服务器应当返回包含 Sybase 支持的有效字符集的名称的字符串。有效的 Sybase 字符集在 Sybase 版本目录中的 `charsets/charset_name/charset.loc` 中定义，每个 *charset_name* 代表一个支持的字符集的名称。例如，文件 `charsets/iso_1/charset.loc` 定义了 iso_1 字符集。

另请参见

- create function string （第 236 页）
- rs_get_sortorder （第 413 页）

rs_get_errormode

返回本地错误配置，该配置确定是否直接从复制服务器返回本地错误。

示例

- **示例 1 – 为 oth_sql_class** 函数字符串类创建一个可返回本地错误的 rs_get_errormode 函数字符串。

```
create function string rs_get_errormode
for oth_sql_class
output language 'select yes'
```

- **示例 2 – 为 oth_sql_class** 函数字符串类创建一个不返回本地错误的 rs_get_errormode 函数字符串。

```
create function string rs_get_errormode
for oth_sql_class
output language 'select no'
```

用法

- **rs_get_errormode** 函数具有函数字符串类作用域。
- Replication Server 会在安装过程中为系统提供的函数字符串类创建一个初始 **rs_get_errormode** 函数字符串。
- 如果使用用户创建的基本函数字符串类，则必须创建 rs_get_errormode 函数字符串。
- 在作为函数字符串类主节点的 Replication Server 上创建或自定义 rs_get_errormode 函数字符串。
- 函数 **rs_get_errormode** 的预期结果为 `yes` 或 `no`。

rs_get_lastcommit

返回 *rs_lastcommit* 系统表中的行。

示例

- **示例 1** – 创建一个 **rs_get_lastcommit** 函数字符串，该函数字符串执行名为 **rs_get_lastcommit** 的存储过程。以下是该存储过程的文本：

```
create procedure rs_get_lastcommit
 as
 select origin, origin_qid, secondary_qid
   from rs_lastcommit
```

```
create function string rs_get_lastcommit
 for sqlserver_derived_class
 output language
 'execute rs_get_lastcommit'
```

用法

- Replication Server 在为数据库启动 DSI 进程时执行 **rs_get_lastcommit**。该函数返回 *rs_lastcommit* 系统表中的所有行。Replication Server 使用这些信息查找从各个主数据源提交的上一个事务。
- 每次 Replication Server 提交数据库中的事务时，将更新 *rs_lastcommit* 系统表。
- **rs_get_lastcommit** 具有函数字符串类范围。
- Replication Server 会在安装过程中为系统提供的函数字符串类创建一个初始 **rs_get_lastcommit** 函数字符串。
- 如果使用用户创建的基本函数字符串类，则必须创建 **rs_get_lastcommit** 函数字符串。
- 在作为函数字符串类主节点的 Replication Server 上创建或自定义 **rs_get_lastcommit** 函数字符串。

- *rs_sqlserver_function_class* 和 *rs_default_function_class* 类的缺省 **rs_get_lastcommit** 函数字符串通过执行 **rs_commit** 函数字符串中名为 **rs_update_lastcommit** 的存储过程来更新 *rs_lastcommit* 表。
- **rs_get_lastcommit** 必须以正确的顺序为在数据库中复制了其数据的各个主数据库返回列。

表 38. rs_get_lastcommit 返回的列

列名	数据类型	说明
origin	*int*	行所代表的主数据库的 ID 号码
origin_qid	*binary(36)*	标识源数据库中的稳定队列中最后一个提交的事务
secondary_qid	*binary(36)*	如果存在源数据库的预订实现队列，该列中包含该队列中已在复制数据库中提交的最后一个事务

另请参见

- create function string （第 236 页）
- rs_commit （第 400 页）

rs_get_sortorder

获取数据服务器使用的排序顺序。如果排序顺序与 Replication Server 的排序顺序不匹配，并且与预期不符，此函数将返回一条警告消息。

示例

- **示例 1** – 创建一个带输出语言的 **rs_get_sortorder** 函数字符串，此函数字符串调用 **sp_serverinfo** 系统过程并返回数据服务器的排序顺序。

```
create function string rs_get_sortorder
 for rs_sqlserver2_function_class
 output language
 'sp_serverinfo server_soname'
```

用法

- **rs_get_sortorder** 函数获取数据服务器使用的排序顺序的名称。Replication Server 在每次连接到数据服务器时执行此函数。如果排序顺序与 Replication Server 的排序顺序不匹配，将向 Replication Server 错误日志中写入一条警告消息。如果排序顺序匹配，将不会写入任何警告消息。
- **rs_get_sortorder** 函数具有函数字符串类范围。
- Replication Server 会在安装过程中为系统提供的函数字符串类创建一个初始 **rs_get_sortorder** 函数字符串。

- 如果使用用户创建的基本函数字符串类，则必须创建 **rs_get_sortorder** 函数字符串。
- 如果需要创建或自定义 **rs_get_sortorder** 函数字符串，在作为函数字符串类主节点的 Replication Server 上进行相应操作。
- *rs_sqlserver_function_class* 和 *rs_default_function_class* 类的缺省 **rs_get_sortorder** 函数字符串使用 *server_soname* 参数调用 Adaptive Server 存储过程 **sp_serverinfo**。
- **rs_get_sortorder** 函数字符串应返回包含 Sybase 支持的有效排序顺序的名称的字符串。给定字符集的有效 Sybase 排序顺序在 Sybase 版本目录中的 `charsets/charset_name/sortorder.srt` 中定义，其中 *charset_name* 代表支持的字符集的名称，*sortorder* 代表该字符集的支持的排序顺序的名称。例如，文件 `charsets/iso_1/nocase.srt` 定义了 iso_1 字符集的“nocase”排序顺序。

另请参见

- create function string （第 236 页）
- rs_get_charset （第 410 页）

rs_get_textptr

检索 *text*、*unitext* 或 *image* 列的说明。

示例

- **示例 1** – 为 *blurbs* 表中的 *repcopy* 列创建 **rs_get_textptr** 函数字符串。函数字符串名 **copy** 是复制定义中的 *text*、*unitext* 或 *image* 列的名称。

```
create function string
 blurbs_rep.rs_get_textptr;copy
 for sqlserver2_function_class
 output language
 'select repcopy from blurbs
 where au_id = ?au_id!new?'
```

用法

- Replication Server 先调用 **rs_get_textptr** 以检索 *text*、*unitext* 或 *image* 列说明，然后使用 Client-Library 函数 **ct_send_data** 发送数据。
- **rs_get_textptr** 具有复制定义范围。
- 在创建复制定义时，Replication Server 为复制定义中的每个复制 *text*、*unitext* 或 *image* 列的 *rs_sqlserver_function_class* 和 *rs_default_function_class* 类生成一个 **rs_get_textptr** 函数字符串。
- 如果使用用户创建的基本函数字符串类，则必须为复制定义中包括的每个复制 *text*、*unitext* 或 *image* 列创建 **rs_get_textptr** 函数字符串。

- 在创建复制定义所在的 Replication Server 上创建或自定义 **rs_get_textptr** 函数字符串。
- **rs_get_textptr** 必须为指定行中的 *text*、*unitext* 或 *image* 列返回 *text* 或 *unitext* 列说明。*text* 或 *unitext* 列说明必须符合 Open Server 对返回“I/O 描述符结构”的要求。有关此结构的信息，请参见《Open Server Server-Library/C 参考手册》。

另请参见

- rs_datarow_for_writetext （第 402 页）
- rs_textptr_init （第 433 页）
- rs_writetext （第 441 页）

rs_get_thread_seq

返回 *rs_threads* 系统表中指定条目的序列号。

语法

```
rs_get_thread_seq @rs_id
```

参数

- **rs_id** – *int* 数据类型的编号。它代表将被检查的条目的 ID 并与 *rs_threads* 系统表中的 *id* 列的值相匹配。

示例

- **示例 1** – 创建一个 **rs_get_thread_seq** 函数字符串，用来执行 *rs_threads* 表中的 **select** 语句。

```
create function string rs_get_thread_seq
 for sqlserver_derived_class
 output language
 'select seq from rs_threads
   where id = ?rs_id!param?'
```

用法

- Replication Server 执行 **rs_get_thread_seq** 以检查以前的事务的完成状态。只有在为一个连接定义了多个 DSI 线程时才执行它。该函数返回只包含一列 *seq* 的单个行，其中包含指定 ID 的序列号。
- 将阻塞调用该函数的线程，直到最后一个修改指定条目的事务完成时为止。
- **rs_get_thread_seq** 具有函数字符串类范围。
- Replication Server 会在安装过程中为系统提供的函数字符串类创建一个初始 **rs_get_thread_seq** 函数字符串。

- 如果使用用户创建的基本函数字符串类并使用并行 DSI 功能，则必须为 **rs_get_thread_seq** 函数创建函数字符串。如果不使用并行 DSI，无需为该函数创建函数字符串。
- 在作为函数字符串类主节点的 Replication Server 上创建或自定义 **rs_get_thread_seq** 函数字符串。

另请参见

- configure connection （第 180 页）
- rs_initialize_threads （第 417 页）
- rs_set_isolation_level （第 429 页）
- rs_update_threads （第 439 页）

rs_get_thread_seq_noholdlock

使用 **noholdlock** 选项返回 *rs_threads* 系统表中指定条目的序列号。

语法

```
rs_get_thread_seq_noholdlock @rs_id
```

参数

- **rs_id** – *int* 数据类型的编号。它代表将被检查的条目的 ID 并与 *rs_threads* 系统表中的 *id* 列的值相匹配。

示例

- **示例 1** – 创建一个 **rs_get_thread_seq_noholdlock** 函数字符串，用来对 *rs_threads* 表执行 **select** 语句。

```
create function string
   rs_get_thread_seq_noholdlock
for sqlserver_derived_class
output language
'select seq from rs_threads noholdlock
  where id = ?rs_id!param?'
```

用法

- **rs_get_thread_seq_noholdlock** 相当于 **rs_get_thread_seq**，只是当 **dsi_isolation_level** 为 3 时，才使用该函数字符串。只有在为连接定义了多个 DSI 线程时才执行它。行选择是通过 **noholdlock** 选项完成的。该函数返回只包含一列 *seq* 的单个行，其中包含指定 ID 的当前序列号。
- **rs_get_thread_seq_noholdlock** 函数具有函数字符串类范围。

- Replication Server 会在安装过程中为系统提供的函数字符串类创建一个初始 **rs_get_thread_seq_noholdlock** 函数字符串。
- 如果使用用户创建的基本函数字符串类并使用事务隔离级别 3 的并行 DSI 功能，则必须为 **rs_get_thread_seq_noholdlock** 创建函数字符串。
- 在作为函数字符串类主节点的 Replication Server 上创建或自定义 **rs_get_thread_seq_noholdlock** 函数字符串。

另请参见

- alter connection （第 109 页）
- rs_get_thread_seq （第 415 页）
- rs_initialize_threads （第 417 页）
- rs_set_isolation_level （第 429 页）
- rs_update_threads （第 439 页）

rs_initialize_threads

将 *rs_threads* 系统表中每个条目的序列设置为 0。

语法

```
rs_initialize_threads @rs_id
```

参数

- **@rs_id** – 从 1 到 *dsi_num_threads* 的数字，代表将被 Replication Server 设置为 0 的条目的 ID。

示例

- **示例 1** – 创建一个 **rs_initialize_threads** 函数字符串，用来执行名为 **rs_initialize_threads** 的存储过程。以下是该存储过程的文本：

```
create procedure rs_initialize_threads
   @rs_id int
 as
     delete from rs_threads where id = @rs_id
     insert into rs_threads values
       (@rs_id, 0,"", "", "", "")
```

```
create function string rs_initialize_threads
 for sqlserver_derived_class
 output language
 'execute rs_initialize_threads
   @rs_id = ?rs_id!param?'
```

用法

- **rs_initialize_threads** 在初始化连接时执行函数。仅当为该连接定义了多个 DSI 线程时才会执行它。它将 *rs_threads* 系统表中每个条目的序列号设置为 0。
- **rs_initialize_threads** 具有函数字符串类范围。
- Replication Server 会在安装过程中为系统提供的函数字符串类创建一个初始 **rs_initialize_threads** 函数字符串。
- 如果使用用户创建的基本函数字符串类并使用并行 DSI 功能，则应为 **rs_initialize_threads** 创建函数字符串。
- 在作为函数字符串类主节点的 Replication Server 上创建或自定义 **rs_initialize_threads** 函数字符串。

另请参见

- create connection （第 214 页）
- rs_get_thread_seq （第 415 页）
- rs_get_thread_seq_noholdlock （第 416 页）
- rs_set_isolation_level （第 429 页）
- rs_update_threads （第 439 页）

rs_insert

在复制数据库的表中插入单个行。

示例

- **示例 1** – 替换 *publishers* 表的 **rs_insert** 函数字符串。

```
alter function string publishers.rs_insert
 for sqlserver_derived_class
 output language
 'insert into publishers (pub_id, pub_name, city,
  state)
  values (?pub_id!new?, ?pub_name!new?,
     ?city!new?, ?state!new?)'
```

用法

- **rs_insert** 具有复制定义作用域。
- 在创建复制定义时，Replication Server 为系统提供的函数字符串类生成一个 **rs_insert** 函数字符串。
- 如果使用用户创建的基本函数字符串类，则应为每个复制定义创建 **rs_insert** 函数字符串。
- 在创建复制定义所在的 Replication Server 上创建或自定义 **rs_insert** 函数字符串。

- 对于每个复制定义的 *rs_sqlserver_function_class* 和 *rs_default_function_class* 类，**rs_insert** 缺省生成的函数字符串使用 Transact-SQL **insert** 命令语法。
- Replication Server 不能使用 **rs_insert** 将 *text*、*unitext* 或 *image* 数据发送到复制数据库，但可以使用 *text_status* 修饰符报告 *text*、*unitext* 或 *image* 数据的状态。有关 *text_status* 修饰符的说明，请参见 rs_datarow_for_writetext。*text*、*unitext* 或 *image* 数据是使用 **rs_get_textptr**、**rs_textptr_init** 和 **rs_writetext** 发送到复制数据库的。

另请参见

- create function string （第 236 页）
- create replication definition （第 258 页）
- rs_datarow_for_writetext （第 402 页）
- rs_delete （第 403 页）
- rs_get_textptr （第 414 页）
- rs_select （第 424 页）
- rs_select_with_lock （第 426 页）
- rs_textptr_init （第 433 页）
- rs_update （第 438 页）

rs_marker

将其参数作为独立命令传递给 Replication Server。

语法

```
rs_marker @rs_api
```

参数

- **rs_api** – 一个包含用于预订实现的数据的 *varchar(255)* 字符串。

示例

- **示例 1** –

```
create function string rs_marker
 for sqlserver_derived_class
 output language
 'execute rs_marker
     @rs_api = ?rs_api!param?'
```

用法

- **rs_marker** 允许 Replication Server 将数据插入事务日志中，以便 RepAgent 线程可以对其进行检索。

- **rs_marker** 函数具有函数字符串类范围。
- Replication Server 会在安装过程中为系统提供的函数字符串类创建一个初始 **rs_marker** 函数字符串。
- 如果使用用户创建的基本函数字符串类，则应为 **rs_marker** 函数创建函数字符串。
- 在作为函数字符串类主节点的 Replication Server 上创建或自定义 **rs_marker** 函数字符串。
- 在预订实现过程中，Replication Server 使用 **rs_marker**，通过主数据库日志将 **activate subscription** 和 **validate subscription** 命令传递给主 Replication Server。
- 主数据库的 RepAgent 必须能够识别 **rs_marker** 函数执行，并将 *@rs_api* 参数作为命令传递给主 Replication Server。
- 对于 Adaptive Server 数据库，为 Replication Server 设置数据库时，将创建一个名为 **rs_marker** 的 Adaptive Server 复制存储过程。该存储过程使用 **sp_setrepproc** 系统过程标记为“已复制”。
- 当 Adaptive Server RepAgent 在事务日志中遇到 **rs_marker** 执行时，则会将 *@rs_api* 参数作为命令发送到主 Replication Server。

注意： 除非按照《Replication Server 管理指南第一卷》中的说明创建批量预订，否则，不要更改 **rs_marker** 函数字符串或调用 **rs_marker** 存储过程。

另请参见

- activate subscription （第 44 页）
- create subscription （第 280 页）
- sp_setrepproc （第 491 页）
- validate subscription （第 390 页）

rs_non_blocking_commit

请求数据服务器立即发送对 COMMIT 语句的肯定响应，而不必等待将事务写入到磁盘中。

用法

- **rs_non_blocking_commit** 具有函数字符串类作用域。
- 如果 **dsi_non_blocking_commit** 的值在 1 到 60 之间，每次 DSI 连接至复制数据服务器时，**rs_non_blocking_commit** 均会执行。如果 **dsi_non_blocking_commit** 的值为零，则 **rs_non_blocking_commit** 不会执行。
- **rs_non_blocking_commit** 函数映射到 Adaptive Server 15.0 及更高版本中的“**set delayed_commit on**”函数字符串，并映射到 Oracle 10g v2 及更高版本中对应的“**alter session set commit_write = nowait;**”函数字符串。对于其它所有非 Sybase 数据库，**rs_non_blocking_commit** 映射到 null。
- 在启用非阻塞提交的情况下，Replication Server 支持到 Oracle 10g v2 或更高版本的复制，因为 Oracle 10g v2 支持类似延迟提交的功能。

Replication Server 15.2 异构数据类型支持 (HDS) 脚本中具有支持非阻塞提交功能的新函数字符串。面向 Oracle 的 Sybase Enterprise Connect Data Access 支持这些函数字符串。请参见《Replication Server Options 概述指南》。

另请参见

- rs_non_blocking_commit_flush （第 421 页）

rs_non_blocking_commit_flush

将 **insert**、**delete** 或 **update** 命令发送到数据服务器，以便将通过连接（使用 **rs_non_blocking_commit** 进行配置）发送的事务保存到磁盘中。

示例

- **示例 1** – 为 Adaptive Server 创建 **rs_non_blocking_commit_flush** 函数字符串实例：

```
create function string rs_non_blocking_commit_flush
         for sqlserver_derived_class
         output language
         'set delayed_commit off; begin tran; update rs_lastcommit
set
         origin_time = getdate() where origin = 0; commit tran;
         set delayed_commit on'
```

- **示例 2** – 为 Oracle 创建 **rs_non_blocking_commit_flush** 函数字符串实例：

```
create function string rs_non_blocking_commit_flush
         for oracle_derived_class
         output language
         'alter session set commit_write = immediate; begin tran;
         update rs_lastcommit set origin_time = getdate() where
        origin = 0; commit tran; alter session set commit_write =
nowait'
```

用法

- **rs_non_blocking_commit_flush** 具有函数字符串类作用域。
- **rs_non_blocking_commit_flush** 以等于您使用 **dsi_non_blocking_commit** 指定的任意分钟数（1 到 60）的间隔执行。如果 **dsi_non_blocking_commit** 的值为零，**rs_non_blocking_commit_flush** 则不执行。
- **rs_non_blocking_commit_flush** 映射到 Adaptive Server 15.0 及更高版本和 Oracle 10g v2 及更高版本中对应的函数字符串。对于其它所有非 Sybase 数据库，**rs_non_blocking_commit_flush** 映射到 null。
- 在启用非阻塞提交的情况下，Replication Server 支持到 Oracle 10g v2 或更高版本的复制，因为 Oracle 10g v2 支持类似延迟提交的功能。

Replication Server 15.2 异构数据类型支持 (HDS) 脚本中具有支持非阻塞提交功能的新函数字符串。面向 Oracle 的 Sybase Enterprise Connect Data Access 支持这些函数字符串。请参见《Replication Server Options 15.1 概述指南》。

另请参见

- rs_non_blocking_commit （第 420 页）

rs_raw_object_serialization

使 Replication Server 能够以序列化的格式处理 Java 列。

用法

- **rs_raw_object_serialization** 允许 Replication Server 将序列化的数据直接插入复制数据库中。
- **rs_raw_object_serialization** 具有函数字符串类范围。
- Replication Server 会在安装过程中为系统提供的函数字符串类 *rs_sql-server_function_class* 和 *rs_default_function_class* 创建一个初始 **rs_raw_object_serialization** 函数字符串。
- 为连接实现或复制第一个 Java 列时，Replication Server 使用 **rs_raw_object_serialization**，将启用 **rs_raw_object_serialization** 的缺省命令传递给 Adaptive Server。

rs_repl_off

指定是否复制 Adaptive Server 数据库中由维护用户执行的事务。

示例

- **示例 1** – 创建 **rs_repl_off** 函数字符串的实例。

```
create function string rs_repl_off
 for sqlserver_derived_class
 output language
 'set replication off'
```

用法

- 对备用数据库的 DSI 连接执行 **rs_repl_off**。
- **rs_repl_off** 具有函数字符串类范围。
- Replication Server 会在安装过程中为系统提供的函数字符串类创建一个初始 **rs_repl_off** 函数字符串。

- 如果使用用户创建的基本函数字符串类，并且计划以非缺省方式使用，则应为 **rs_repl_off** 创建函数字符串。
- 在作为函数字符串类主节点的 Replication Server 上创建或自定义 **rs_repl_off** 函数字符串。
- 备用数据库连接始终使用系统提供的类 *rs_default_function_class*，该类不能进行修改。因此，如果不使用热备份，就无需为 **rs_repl_off** 创建函数字符串。
- 您可以使用 **alter connection** 或 **configure connection** 设置 **dsi_replication** 配置参数，并指定是否在连接到备用数据库时执行 **rs_repl_off** 函数。将 **dsi_replication** 设置为“off”，以执行 **rs_repl_off**。
- 在热备份应用程序中，对于活动数据库，Replication Server 将 **dsi_replication** 设置为“on”，对于备用数据库，则将其设置为“off”。

另请参见

- create connection （第 214 页）
- create function string （第 236 页）

rs_repl_on

在 Adaptive Server 中将一个或多个数据库连接的复制设置为 on。

示例

- **示例 1** – 创建 **rs_repl_on** 函数字符串的实例：

```
create function string rs_repl_on
 for sqlserver_derived_class
 output language
 'set replication on'
```

用法

- 为数据库的 DSI 连接执行 **rs_repl_on**。
- **rs_repl_on** 具有函数字符串类范围。
- Replication Server 会在安装过程中为系统提供的函数字符串类创建一个初始 **rs_repl_on** 函数字符串。
- 如果使用用户创建的基本函数字符串类，并且计划以非缺省方式使用，则应为 **rs_repl_on** 创建函数字符串。
- 在作为函数字符串类主节点的 Replication Server 上创建或自定义 **rs_repl_on** 函数字符串。

另请参见

- alter connection （第 109 页）
- rs_repl_off （第 422 页）

rs_rollback

回退事务。该函数留作将来使用。

示例

- **示例 1** – 此示例说明了 *rs_sqlserver_function_class* 和 *rs_default_function_class* 类的缺省 **rs_rollback** 函数字符串。

```
create function string rs_rollback
 for sqlserver_derived_class
 output language
 'rollback transaction'
```

用法

- 从主数据库事务日志检索的回退事务不会被分发到复制 Replication Server，因此此函数永远不会执行。
- **rs_rollback** 函数具有函数字符串类范围。
- Replication Server 会在安装过程中为系统提供的函数字符串类创建一个初始 **rs_rollback** 函数字符串。

另请参见

- alter function string （第 143 页）
- create function string （第 236 页）
- rs_begin （第 399 页）
- rs_commit （第 400 页）

rs_select

从被复制的表的主副本中为预订实现选择行，从该表的复制副本中为预订取消实现选择行。

示例

- **示例 1** – 创建 **rs_select** 函数字符串的实例。Replication Server 在预订 **where** 子句为 *au_lname* 列指定了特定的值时使用此函数字符串。

```
create function string
   authors.rs_select;name_select
 for flat_file_class
 scan 'select * from authors
   where au_lname = ?l_name!user?'
```

```
output rpc
'execute name_sel ?l_name!user?, "authors"'
```

用法

- 如果 **create subscription** 命令中包括 **without holdlock**，Replication Server 将执行 **rs_select**，以从主 Replication Server 检索预订实现行。**without holdlock** 用于非原子实现中。用于此操作的函数字符串位于指派给主数据库的类。
- 要在原子实现过程中检索数据，应使用与主数据库连接相关联的函数字符串类和错误类，而不是使用与复制数据库连接相关联的类。
- 如果使用 **incrementally with purge** 删除表复制定义的预订，Replication Server 还会执行 **rs_select** 来标识预订取消实现的行。用于此操作的函数字符串位于指派给复制数据库的类。
- 如果 **create subscription** 不包括 **without holdlock**，Replication Server 将执行 **rs_select_with_lock** 函数，而不是 **rs_select**。
- **rs_select** 具有复制定义范围。
- 在创建复制定义时，Replication Server 为系统提供的函数字符串类生成一个 **rs_select** 函数字符串。
- 如果使用用户创建的基本函数字符串类，则应为每个复制定义创建 **rs_select** 函数字符串，以匹配每个可能的预订 **where** 子句。
- 在创建复制定义所在的 Replication Server 上创建或自定义 **rs_select** 函数字符串。
- 对于每个复制定义的 *rs_sqlserver_function_class* 和 *rs_default_function_class* 类，**rs_select** 缺省生成的函数字符串使用 Transact-SQL **select** 命令语法。
- **rs_select** 的函数字符串具有输入和输出模板。输入模板是带一个 **where** 子句的 SQL **select** 命令，Replication Server 使之与 **create subscription** 命令中的 **where** 子句相匹配。
- 如果 Replication Server 无法将 **select** 操作中的 **where** 子句与某个函数字符串输入模板匹配，它将使用不带输入模板的函数字符串（如果有）。
- 如果 Replication Server 无法找到带有匹配输入模板的函数字符串或者不带输入模板的函数字符串，**rs_select** 函数调用就会失败。

另请参见

- alter function string （第 143 页）
- create function string （第 236 页）
- create subscription （第 280 页）
- rs_delete （第 403 页）
- rs_insert （第 418 页）
- rs_select_with_lock （第 426 页）
- rs_update （第 438 页）

rs_select_with_lock

从被复制的表的主副本中为预订实现选择行，使用锁定保持串行一致性。

示例

- **示例 1** – 创建 **rs_select_with_lock** 函数字符串的实例。Replication Server 在预订 **where** 子句为 *au_lname* 列指定值时使用此函数字符串。

```
create function string
   authors.rs_select_with_lock;name_select
for flat_file_class
scan 'select * from authors
  where au_lname = ?l_name!user?'
output rpc
'execute name_sel_lock ?l_name!user?, "authors"'
```

用法

- 当 **without holdlock** 子句与 **create subscription** 子句一同使用时，Replication Server 执行 **rs_select_with_lock** 函数，从主 Replication Server 中检索初始预订行。**without holdlock** 子句不在原子实现中使用。用于此操作的函数字符串位于指派给主数据库的类。
- 如果使用 **with purge** 删除表复制定义的预订，Replication Server 还会执行 **rs_select_with_lock** 标识预订取消实现的行。用于此操作的函数字符串位于指派给复制数据库的类。
- 如果 **without holdlock** 子句包括在 **create subscription** 中，Replication Server 将执行 **rs_select** 函数，而不是 **rs_select_with_lock**。
- **rs_select_with_lock** 具有复制定义范围。
- 在创建复制定义时，Replication Server 为系统提供的函数字符串类生成一个 **rs_select_with_lock** 函数字符串。
- 如果使用用户创建的基本函数字符串类，则应为每个复制定义创建 **rs_select_with_lock** 函数字符串，以匹配每个可能的预订 **where** 子句。
- 在创建复制定义所在的 Replication Server 上创建或自定义 **rs_select_with_lock** 函数字符串。
- 对于每个复制定义的 *rs_sqlserver_function_class* 和 *rs_default_function_class* 类，**rs_select_with_lock** 缺省生成的函数字符串使用 Transact-SQL **select...holdlock** 命令语法。
- **rs_select_with_lock** 的函数字符串具有输入和输出模板。输入模板是带一个 **where** 子句的 SQL **select** 命令，Replication Server 使之与 **create subscription** 命令中的 **where** 子句相匹配。
- 如果 Replication Server 无法将 **select** 操作中的 **where** 子句与某个函数字符串输入模板匹配，它将使用不带输入模板的函数字符串（如果有）。

- 如果 Replication Server 无法找到带有匹配输入模板的函数字符串或者不带输入模板的函数字符串，**rs_select_with_lock** 函数调用就会失败。

另请参见

- alter function string （第 143 页）
- create function string （第 236 页）
- create subscription （第 280 页）
- rs_delete （第 403 页）
- rs_insert （第 418 页）
- rs_select （第 424 页）
- rs_update （第 438 页）

rs_session_setting

为 Sybase IQ 复制数据库连接的持续时间设置 Sybase IQ 参数和数据库选项。

示例

- **示例 1** – 为 **my_iq_fclass** 函数字符串类创建 **rs_session_setting** 函数字符串，并包括要设置的 Sybase IQ 参数，如 **LOAD_MEMORY_MB**、**MINIMIZE_STORAGE** 和 **JOIN_PREFERENCE** Sybase IQ 数据库选项：

```
create function string rs_session_setting
for my_iq_fclass
output language
'set temporary option Load_Memory_MB=''200''
set temporary option Minimize_Storage=''on''
set temporary option join_preference=5'
go
```

用法

- Sybase IQ 数据库选项的值是使用 **TEMPORARY** 关键字设置的，因此仅适用于当前 Sybase IQ 连接。当重新启动与 Sybase IQ 数据库的连接时，这些值将恢复为缺省值或先前未使用 **TEMPORARY** 关键字设置的值。请参见“Sybase IQ 15.2”>“参考：语句和选项”的“数据库选项”的“数据库选项简介”中的“设置选项”。
- **rs_session_setting** 具有函数字符串类作用域。
- Replication Server 会在安装过程中为系统提供的函数字符串类创建一个初始 **rs_session_setting** 函数字符串。
- 在作为函数字符串类主节点的 Replication Server 上创建或自定义 **rs_session_setting** 函数字符串。
- 为 **rs_session_setting** 函数生成的缺省函数字符串对于：
  - **rs_sqlserver_function_class** 和 **rs_default_function_class** 类为空字符串

- **rs_iq_function_class** 类为：

```
{set temporary option Load_Memory_MB=''200''
set temporary option Minimize_Storage=''on''
set temporary option join_preference=5}
```

- 在 Sybase 15.2 和更高版本中不推荐使用 **LOAD_MEMORY_MB** 数据库选项。请参见“Sybase IQ 15.2”的“新增功能摘要”的“行为更改”中的“数据库选项更改”。

rs_set_ciphertext

允许将加密列复制到 Adaptive Server 表。

示例

- **示例 1** – 修改不支持“**set ciphertext on**”的非 Adaptive Server 数据库的 **rs_set_ciphertext**：

```
alter function string rs_set_ciphertext
    for some_function_string_class
    output language
    ''
```

用法

- 在为任何用户数据库连接执行 **rs_usedb** 后调用 **rs_set_ciphertext**。Replication Server 不会为 Replication Server 连接和 RSSD 连接调用此函数字符串。
- 对于 **rs_default_function_class** 和 **rs_sqlserver_function_class**，**rs_set_ciphertext** 发出“**set ciphertext on**”。对于所有其它函数类，则将 **rs_set_ciphertext** 设置为 null（空字符串）。
- 发生故障时，Replication Server 将继续运行，并且不会向用户报告。这是为了向后兼容不支持“**set ciphertext on**”的早期版本 Adaptive Server。
- 加密列以加密形式的 *varbinary* 到达 Replication Server。对于实现和取消实现，Replication Server 必须对数据库连接执行“**set ciphertext on**”，或者调用 Adaptive Server **ciphertext()** 函数。
- 无论是否有要复制的加密列，或者目标数据库是否接受 ciphertext 属性，Replication Server 始终将 ciphertext 属性设置为 on。
- 不要将加密列指定为可搜索列。Replication Server 不知道 *varbinary* 列是密文还是明文二进制数据，并且无法防止加密列成为搜索列。
- 不要将加密列映射到 *varbinary* 以外的数据类型。Replication Server 不知道列是否为加密列，并且无法防止密文转换为其它数据类型。
- Replication Server 无法加密 *text**unitext* 和 *image* 列。

另请参见

- alter connection （第 109 页）

- alter function string （第 143 页）
- create database replication definition （第 225 页）
- create replication definition （第 258 页）

rs_set_dml_on_computed

允许将实现的计算列作为常规列复制到复制 Adaptive Server 数据库。

用法

- 对于 Adaptive Server 复制数据库，rs_set_dml_on_computed 映射到命令 **set dml_on_computed “on”** 。对于所有非 Sybase 数据库，此函数映射到 null。
- **rs_set_dml_on_computed** 具有函数字符串类范围。
- 在建立连接时，将始终在执行 **use database** 命令后在 DSI 中应用 rs_set_dml_on_computed。
- Adaptive Server version 12.5.*x* 及更早版本的数据库不支持 **set dml_on_computed “on”** 。发生故障时，Replication Server 将继续运行，并且不会向用户报告。

另请参见

- create replication definition （第 258 页）

rs_set_isolation_level

将事务隔离级别传递给复制数据服务器。

示例

- **示例 1** – 创建 **rs_set_isolation_level** 函数字符串的实例。

```
create function string rs_set_isolation_level
for sqlserver_derived_class
output language
'set transaction isolation level?rs_isolation_level!sys_raw?'
```

用法

- 如果已为 **dsi_isolation_level** 设置了值，则 **rs_set_isolation_level** 函数将事务隔离级别传递给复制数据服务器，并在 DSI 每次连接到复制数据服务器时执行。如果 **dsi_isolation_level** 为缺省值，则不会执行 **rs_set_isolation_level**。
- 使用 **alter connection** 或 **create connection**，并将 **set_isolation_level** 选项设置为 *rs_isolation_level* 变量的值。支持的 Adaptive Server 值为 0、1、2 和 3。Replication Server 支持其它数据服务器所支持的所有其它隔离级别值。如果没有为 *rs_isolation_level* 提供任何值，Replication Server 将使用目标数据服务器的隔离值。

- 在执行 **rs_usedb** 函数字符串命令后，Replication Server 将立即执行 **rs_set_isolation_level**。
- **rs_set_isolation_level** 函数具有函数字符串类范围。
- Replication Server 会在安装过程中为 Adaptive Server 和缺省函数字符串类创建一个初始 **rs_set_isolation_level** 函数字符串。
- 如果使用非缺省函数字符串类并使用并行 DSI 功能，则应为 **rs_set_isolation_level** 函数创建函数字符串。修改的函数字符串必须包含 *rs_isolation_level* 变量。
- 在作为函数字符串类主节点的 Replication Server 上创建或自定义 **rs_set_isolation_level** 函数字符串。

另请参见

- create connection （第 214 页）
- rs_get_thread_seq （第 415 页）
- rs_initialize_threads （第 417 页）
- rs_update_threads （第 439 页）

rs_set_quoted_identifier

配置数据服务器连接以接受带引号的标识符。

注意： 在长度、特殊字符以及支持的保留字方面，Adaptive Server、SQL Anywhere、Microsoft SQL Server、Universal Database (UDB) 以及 Oracle 等数据服务器处理带引号的标识符的方式并不相同。在异构环境中，您必须确保所复制的带引号标识符在主数据服务器和复制数据服务器上均有效。

用法

- **rs_set_quoted_identifier** 将添加到缺省函数字符串类中，并且具有函数字符串类范围。
- 当 **dsi_quoted_identifier** 为 on 时，Replication Server 将 **rs_set_quoted_identifier** 发送到复制数据服务器，表明数据服务器期望收到带引号的标识符。如果复制数据服务器为 Adaptive Server、SQL Anywhere 或 Microsoft SQL Server，则将 **rs_set_quoted_identifier** 设置为 **set quoted_identifiers on** 命令。否则，**rs_set_quoted_identifier** 设置为 ""。

另请参见

- create connection （第 214 页）
- create replication definition （第 258 页）
- alter connection （第 109 页）
- alter replication definition （第 152 页）

rs_set_timestamp_insert

允许将时间戳列复制到 Adaptive Server 表。

示例

- **示例 1** – 修改不支持 **set timestamp_insert on** 的非 Adaptive Server 数据库的 **rs_set_timestamp_insert**：

```
alter function string rs_set_timestamp_insert
    for some_function_string_class
    output language
    ''
```

用法

- 在为任何用户数据库连接执行 **rs_usedb** 后调用 **rs_set_timestamp_insert**。Replication Server 不会为 RSSD 连接调用此函数字符串。
- **rs_set_timestamp_insert** 具有函数字符串类范围。
- **rs_set_timestamp_insert** 映射到 Adaptive Server 复制数据库的 **set timestamp_insert on**。对于所有非 Adaptive Server 数据库，**rs_set_timestamp_insert** 映射到 null。
- Adaptive Server 15.0.1 和更低版本不支持 **set timestamp_insert on**。
- 如果执行 **rs_set_timestamp_insert** 失败，Replication Server 将继续运行，并且不会向用户报告。

另请参见

- alter function string （第 143 页）
- create replication definition （第 258 页）

rs_setproxy

更改数据服务器中的登录名。

用法

- **rs_setproxy** 具有函数字符串类作用域。
- Replication Server 会在安装过程中为 **rs_sqlserver_function_class** 函数字符串类创建 **rs_setproxy** 函数字符串。缺省值为：

  set session authorization “?rs_destination_user!sys”

  生成的字符串使用 Adaptive Server **set proxy** 命令的语法。使用 **alter function string** 可以替换缺省的函数字符串。

- 如果数据服务器不支持网络安全服务，或者没有对应的 **set proxy** 命令，可以将 **unified_login** 设置为“not required”，或者创建一个空的 **rs_setproxy** 函数字符串。
- 函数字符串变量修饰符 *sys* 包含数据服务器的登录名。该登录名通常是维护用户或预订用户的登录名。

另请参见

- alter function string （第 143 页）
- create function string （第 236 页）

rs_sqldml

用于将 SQLDML 传送到 Replication Server 的复制函数。

示例

- **示例 1** – 将 SQLDML 作为名为 **rs_sqldml** 的存储过程发送到 Replication Server：

```
create proc rs_sqldml
   @rs_operator char(1),
   @rs_status int,
   @rs_insert_column varchar(16384),
   @rs_from varchar(16384),
   @rs_where varchar(16384),
   @rs_set varchar(16384),
   @rs_select varchar(16384),
   @rs_owner varchar(255),
   @rs_object varchar(255),
   @rs_rowcount int
```

其中：

- *rs_operator* – 任何以下操作：
  - U – **update**
  - D – **delete**
  - I – **insert select**
  - S – **select into**
- *rs_object* – 操作的表名
- *rs_owner* – 操作的表所有者。如果表所有者状态为 off，则所有者名称为 null。
- *rs_category* – SQLDML 类别：
  - C1 – 可以应用于任何复制数据库并生成完全相同的结果集的语句。
  - C2 – 只能应用于热备份数据库或 MSA 数据库以生成完全相同的结果集的语句。
- *rs_status* – SQLDML 状态。
- *rs_set* – **UPDATE** 语句中的 **set** 子句
- *rs_where* – **where** 子句

- *rs_select* – **INSERT SELECT** 或 **SELECT INTO** 语句中的 **select** 子句
- *rs_from* – **INSERT SELECT** 或 **SELECT INTO** 语句中的 **from** 子句
- *rs_insert_column* – **INSERT SELECT** 语句的列列表
- *rs_rowcount* – 受影响的行数，它仅在 **rs_sqldml** 结尾处可用。

用法

- **rs_sqldml** 将作为复制的函数发送到 Replication Server。如果 SQLDML 没有 **responding** 子句，则将该参数设置为 null。
- **SELECT INTO** 无法在用户定义的事务内执行，它将作为系统事务进行复制。
- RepAgent 将 **rs_sqldml** 及其受影响的行日志记录发送到 Replication Server，Replication Server 决定是否将 SQLDML 或受影响的行应用于目标。
- Adaptive Server 记录 **execbegin rs_sqldml** 以指示 SQLDML 开头，并记录 **execend rs_sqldml** 以指示 SQLDML 结尾。SQLDML 将打包在 **execbegin** 命令内；*@rs_rowcount* 打包在 execend 命令内。
- 若要防止记录更改的行数少于 SQLDML 复制阈值的 SQLDML，Adaptive Server 将对 **execbegin** 执行延迟记录。在 SQLDML 更改的行数超过阈值后，它才会执行 execbegin。RepAgent 将标记 SQLDML 的第一个日志记录。
- SQLDML 延迟记录并不是必需的。例如，非 Adaptive Server 复制代理可能无法执行延迟记录。

rs_textptr_init

为 *text*、*unitext* 或 *image* 列分配文本指针。

示例

- **示例 1** – 为 *blurbs* 表中的 *copy* 列创建 **rs_textptr_init** 函数字符串。

```
create function string blurbs_rep.rs_textptr_init;copy
   for sqlserver2_function_class
   output language
      'update blurbs set copy = NULL
           where au_id = ?au_id!new?'
```

用法

- Replication Server 在行到达时执行 **rs_textptr_init**，指示主数据库中进行了修改，从而导致为 *text*、*unitext* 或 *image* 列分配文本指针。当 Replication Server 需要在复制数据库中执行 **writetext** 操作并且尚未分配文本指针时，也会执行该函数字符串。
- **rs_textptr_init** 函数具有复制定义范围。

- 在创建复制定义时，Replication Server 为复制定义中的每个复制 *text*、*unitext* 或 *image* 列的 *rs_sqlserver_function_class* 和 *rs_default_function_class* 类生成一个 **rs_textptr_init** 函数字符串。
- 如果使用用户创建的基本函数字符串类，则应为复制定义中包括的每个复制 *text*、*unitext* 或 *image* 列创建 **rs_textptr_init** 函数字符串。
- 在创建复制定义所在的 Replication Server 上创建或自定义 **rs_textptr_init** 函数字符串。

另请参见

- rs_get_textptr （第 414 页）
- rs_datarow_for_writetext （第 402 页）
- rs_writetext （第 441 页）

rs_ticket_report

将票据插入 *rs_ticket_history* 表中。

示例

- **示例 1** – 自定义的 **rs_ticket_report** 示例：

```
alter function string rs_ticket_report
   for rs_sqlserver_function_class
   output language
      'insert rs_ticket_history(h1,h2,h3,h4,
         pdb,prs,rrs,rdb,pdb_t,exec_t, dist_t,rsi_t,
         dsi_t,exec_b,rsi_b,dsi_tnx,dsi_cmd,ticket)
      values(?h1!param?, ?h2!param?, ?h3!param?,
         ?h4!param?, ?rs_origin_db!sys?, ?prs!param?,
         ?rrs!param?, ?rs_destination_db!sys?,
         ?pdb!param?, ?exec!param?, ?dist!param?,
         ?rsi!param?, ?dsi!param?, ?b!param?,
         ?rsi_b!param?, ?dsi_t!param?, ?dsi_c!param?,
         ?rs_ticket_param!param?)'
```

用法

- **rs_ticket_report** 具有函数字符串类作用域。
- **rs_ticket_report** 将 **rs_ticket** 信息写入 *rs_ticket_history* 表。不过，您可以自定义 **rs_ticket_report**，以便根据需要使用 **rs_ticket** 信息。
  有关 *rs_ticket_history* 参数的信息，请参见“**rs_ticket_history**”。
- 若要禁用 **rs_ticket_report**，请将连接配置参数 **dsi_rs_ticket_report** 设置为 off。

另请参见

- rs_ticket （第 548 页）
- rs_ticket_history （第 625 页）

rs_triggers_reset

关闭 Adaptive Server 和 Oracle 中的触发器。

示例

- **示例 1** – 为 Adaptive Server 的用户创建基本函数字符串类创建 **rs_triggers_reset** 函数字符串的实例。

```
create function string rs_triggers_reset
 for sqlserver2_function_class
 output language
 'set triggers off'
```

- **示例 2** – 为 Oracle 的用户创建基本函数字符串类创建 **rs_triggers_reset** 函数字符串的实例。

```
create function string rs_triggers_reset
 for oracle_function_class
 output language
 'BEGIN rs_trigger_control.enable();; END;;'
```

注意： 与具有 **set triggers off** 命令的 Adaptive Server 不同，Oracle 不发布会话级触发器控制。因此，您需要在复制 Oracle 数据库中使用 **create connection using profile** 安装 RS_TRIGGER_CONTROL 软件包，以便能够使用 **rs_triggers_reset** 函数字符串。请参见《Replication Server 异构复制指南》中的“Oracle 复制数据服务器问题”。

用法

- 缺省情况下，对备用数据库的 DSI 连接执行 **rs_triggers_reset** 函数，而对任何其它 DSI 连接不执行该函数。
- **rs_triggers_reset** 具有函数字符串类范围。
- Replication Server 会在安装过程中为系统提供的函数字符串类创建一个初始 **rs_triggers_reset** 函数字符串。
- 备用数据库连接始终使用系统提供的类 *rs_default_function_class*，该类不能进行修改。对于任何其它数据库连接，无需为 **rs_triggers_reset** 函数创建函数字符串，除非：
  - 数据库连接使用用户创建的基本函数字符串类，并且
  - 要将该连接的 **dsi_keep_triggers** 配置参数设置为“off”。
- 在作为函数字符串类主节点的 Replication Server 上创建一个 **rs_triggers_reset** 函数字符串。
- 将数据库连接的 **dsi_keep_triggers** 设置为“off”，以便在建立连接时执行 **rs_triggers_reset**。对于备用数据库，**dsi_keep_triggers** 的缺省值为“off”，对于

复制数据库，则为“on”。可以使用 **alter connection** 或 **configure connection** 命令来更改此设置。

另请参见

- create connection （第 214 页）
- create function string （第 236 页）

rs_truncate

在复制数据库中截断表或表分区。

示例

- **示例 1** – 使用一个执行 Transact-SQL **delete** 命令（记录所有删除）而不是 **truncate table** 命令（不记录删除）的函数字符串来替换 *authors* 表的现有 **rs_truncate** 函数字符串。

```
alter function string authors.rs_truncate
 for sqlserver_derived_class
 output language
 'delete authors'
```

  如果出现以下情况，需要为 *authors* 表自定义 **rs_truncate** 函数字符串：

  - 复制数据库不支持 Transact-SQL **truncate table** 命令，或者
  - 需要在复制数据库上记录删除。

- **示例 2** – 替换 *publisher* 表的现有 **rs_truncate** 函数字符串，以将 **truncate table partition** 作为 **delete** 命令进行复制：

```
alter function string publisher.rs_truncate
  for rs_sqlserver_function_class
  output language
  'begin transaction
    if (?1!param? = '''') /* No parameter */
      delete publisher
    if (?1!param? = ''A'')
      delete publisher where c1 < 1000
    if (?1!param? = ''B'')
      delete publisher where c1 >= 1000
commit transaction'
```

- **示例 3** – 将函数字符串更改为在具有参数时不执行任何操作，以便在复制中不截断表分区：

```
alter function string publisher.rs_truncate
  for rs_sqlserver_function_class
  output language
  'if(?1!param? = '''') delete publisher'
```

用法

- **rs_truncate** 具有复制定义范围。Replication Server 执行该函数以截断表或者一个或多个表分区。
- 在创建复制定义时，Replication Server 为系统提供的函数字符串类生成一个 **rs_truncate** 函数字符串。
- 如果使用用户创建的基本函数字符串类，则应为每个复制定义创建 **rs_truncate** 函数字符串。
- 在创建复制定义所在的 Replication Server 上创建或自定义 **rs_truncate** 函数字符串。
- 对于每个复制定义的 *rs_sqlserver_function_class* 和 *rs_default_function_class* 类，**rs_truncate** 缺省生成的函数字符串使用 Transact-SQL **truncate table** 命令语法。它将删除表中所有的行，而不会记录各个行的删除。
- Replication Server 将重建在主节点上执行的相同命令。此命令要求复制节点具有相同的分区名称。否则，DSI 将关闭。
- 分区名称作为参数传递给 **rs_truncate** 函数。**rs_truncate** 函数字符串接受基于位置的函数字符串参数。以下是基于位置的变量：

  ```
  ?n!param?
  ```

  函数字符串变量 **?1!param?** 对应于 **rs_truncate** 函数中的第一个参数。
- 如果函数字符串包含基于位置的函数字符串变量，则函数字符串的最低版本为 1500。如果复制定义包含 1500 版函数字符串，则其最低版本至少为 1500。

表 39. 函数字符串变量修饰符

修饰符	说明
new、*new_raw*	对要插入或更新的行中某一列的新值的引用
old、*old_raw*	对要更新或删除的行中某一列的现有值的引用
user、*user_raw*	对 **rs_select** 或 **rs_select_with_lock** 函数字符串的输入模板中定义的变量的引用。
sys、*sys_raw*	对系统定义变量的引用
param、*param_raw*	对函数参数的引用
text_status	对函数参数的引用。如果此参数不是通过函数复制定义或用户定义的函数 (**create function**) 定义的，则参数名称处必须有一个介于 1 和 99 之间的数字（无前导 0），用以说明此参数在 LTL 命令中的函数中所处的位置。

另请参见

- alter function string （第 143 页）
- rs_datarow_for_writetext （第 402 页）
- rs_get_textptr （第 414 页）

- rs_insert （第 418 页）
- rs_delete （第 403 页）
- rs_textptr_init （第 433 页）
- rs_writetext （第 441 页）
- set （第 328 页）

rs_update

更新复制数据库的表中的单个行。

示例

- **示例 1** – 使用与 Replication Server 为系统提供的函数字符串类生成的缺省函数字符串类似的函数字符串替换 *authors* 表的现有 **rs_update** 函数字符串。

```
alter function string authors.rs_update
 for sqlserver_derived_class
 output language
 'update authors set au_id = ?au_id!new?,
     au_lname = ?au_lname!new?,
     au_fname = ?au_fname!new?,
     phone = ?phone!new?,
     address = ?address!new?,
     city = ?city!new?,
     state = ?state!new?,
     country = ?country!new?,
     postalcode = ?postalcode!new?
   where au_id = ?au_id!old?'
```

用法

- Replication Server 执行 **rs_update** 以更新表中的单个行。该行由在复制定义中为该表定义的主键列标识。
- **rs_update** 函数具有复制定义范围。
- 在创建复制定义时，Replication Server 为系统提供的函数字符串类生成一个 **rs_update** 函数字符串。
- 如果使用用户创建的基本函数字符串类，则应为每个复制定义创建 **rs_update** 函数字符串。
- 在创建复制定义所在的 Replication Server 上创建或自定义 **rs_update** 函数字符串。
- 对于每个复制定义的 *rs_sqlserver_function_class* 和 *rs_default_function_class* 类，**rs_update** 缺省生成的函数字符串使用 Transact-SQL **update** 命令语法。它将替换行中的所有列，并使用指定主键列的预更新值或前映像的 **where** 子句标识该行。
- 当 **set autocorrection** 为 **on** 时，Replication Server 不使用 **rs_update**，而是调用 **rs_delete** 来删除现有的行，调用 **rs_insert** 来插入行。
- Replication Server 不能使用 **rs_update** 发送 *text*、*unitext* 或 *image* 数据，但可以使用 *text_status* 修饰符报告 *text*、*unitext* 或 *image* 数据的状态。有关 *text_status* 修饰符

的说明，请参见 rs_datarow_for_writetext。*text*、*unitext* 或 *image* 类型的数据是使用 **rs_get_textptr**、**rs_textptr_init**、**rs_datarow_for_writetext** 和 **rs_writetext** 函数发送到复制数据库的。

另请参见

- alter function string （第 143 页）
- rs_datarow_for_writetext （第 402 页）
- rs_get_textptr （第 414 页）
- rs_insert （第 418 页）
- rs_delete （第 403 页）
- rs_textptr_init （第 433 页）
- rs_writetext （第 441 页）
- set （第 328 页）

rs_update_threads

更新 *rs_threads* 系统表中指定条目的序列号。

语法

```
rs_update_threads @rs_id, @rs_seq
```

参数

- **rs_id** – *int* 数据类型的编号，代表要更新的条目的 ID。
- **rs_seq** – *int* 数据类型的编号，代表条目的新序列号。

示例

- **示例 1** – 创建 **rs_update_threads** 函数字符串，用来执行名为 **rs_update_threads** 的存储过程。以下是该存储过程的文本：

```
create function string rs_update_threads
  for sqlserver_derived_class
  output language
  'execute rs_update_threads
  @rs_seq = ?rs_seq!param?,
  @rs_id = ?rs_id!param?'
```

```
create procedure rs_update_threads
  @rs_id int,
  @rs_seq int
  as
  update rs_threads set seq = @rs_seq
    where id = @rs_id
```

用法

- 如果为一个连接定义了多个 DSI 线程，在每个事务启动时，将执行 **rs_update_threads** 函数。只有在为一个连接定义了多个 DSI 线程时才执行它。
- **rs_update_threads** 函数具有函数字符串类范围。
- Replication Server 会在安装过程中为系统提供的函数字符串类创建一个初始 **rs_update_threads** 函数字符串。
- 如果使用用户创建的基本函数字符串类和并行 DSI 功能，则应为 **rs_update_threads** 函数创建函数字符串。
- 在作为函数字符串类主节点的 Replication Server 上创建或自定义 **rs_update_threads** 函数字符串。

另请参见

- create connection （第 214 页）
- rs_get_thread_seq （第 415 页）
- rs_initialize_threads （第 417 页）
- rs_set_isolation_level （第 429 页）

rs_usedb

更改数据服务器中的数据库上下文。

示例

- **示例 1** – 将现有的 **rs_usedb** 函数字符串更改为与 Replication Server 为系统提供的函数字符串类生成的缺省函数字符串类似的函数字符串。

```
alter function string rs_usedb
 for sqlserver_derived_class
 output language
 'use ?rs_destination_db!sys_raw?'
```

- **示例 2** – 对于不支持多个数据库的数据服务器，使用空字符串为输出模板创建 **rs_usedb** 函数字符串。

```
create function string rs_usedb
 for TOKYO_DS
 output language ''
```

用法

- Replication Server DSI 在第一次连接到数据服务器时执行此函数。
- **rs_usedb** 具有函数字符串类范围。
- Replication Server 会在安装过程中为系统提供的函数字符串类创建一个初始 **rs_usedb** 函数字符串。

- 如果使用用户创建的基本函数字符串类，则应为 **rs_usedb** 函数创建函数字符串。
- 在作为函数字符串类主节点的 Replication Server 上创建或自定义 **rs_usedb** 函数字符串。
- 对于 *rs_sqlserver_function_class* 和 *rs_default_function_class* 类，**rs_usedb** 函数缺省生成的函数字符串具有 Transact-SQL **use** 命令的语法。
- 如果数据服务器不支持多个数据库或数据库上下文，输出模板可以为空字符串 (' ')。

另请参见

- alter function string （第 143 页）
- create function string （第 236 页）

rs_writetext

修改复制数据库中的 *text*、*unitext* 或 *image* 数据。

示例

- **示例 1** – 创建 **rs_writetext** 函数字符串，该函数字符串使用 RPC 方法更新 *blurbs* 表中的 *copy* 列。

```
create function string
   blurbs_rep.rs_writetext;copy
 for gw_function_class
 output rpc
 'execute update_blurbs_copy
   @copy_chunk = ?copy!new?,
   @au_id = ?au_id!new?,
   @last_chunk = ?rs_last_text_chunk!sys?,
   @writetext_log = ?rs_writetext_log!sys?'
```

- **示例 2** – 创建 **rs_writetext** 函数字符串，该函数字符串使用 **writetext** 方法更新 *copy* 列。Replication Server 通过使用对 *copy* 列执行 **rs_get_textptr** 函数时返回的 I/O 描述符来修改 *copy* 列。

```
create function string
   blurbs_rep.rs_writetext;copy
 for rs_sqlserver2_function_class
 output writetext
 use primary log
```

例如，如果您具有 **rs_get_textptr** 的函数字符串，**rs_writetext** 函数就按如下方式修改 *blurbs* 表中的 *repcopy* 列：

```
create function string
   blurbs_rep.rs_get_textptr;copy
 for sqlserver2_function_class
 output language
```

```
 'select repcopy from blurbs
  where au_id = ?au_id!new?'
```

- **示例 3** – 创建 **rs_writetext** 函数字符串，该函数字符串使用 **none** 方法指定不应更新 *copy* 列。

```
create function string
   blurbs_rep.rs_writetext;copy
 for rs_sqlserver2_function_class
 output none
```

用法

- **rs_writetext** 具有复制定义作用域。
- 在创建复制定义时，Replication Server 为复制定义中的每个复制 *text*、*unitext* 或 *image* 列的 *rs_sqlserver_function_class* 和 *rs_default_function_class* 类生成一个 **rs_writetext** 函数字符串。
- 如果使用用户创建的函数字符串类，则应为复制定义中包括的每个复制 *text*、*unitext* 或 *image* 列创建 **rs_writetext** 函数字符串。
- 在创建复制定义所在的 Replication Server 上创建或自定义 **rs_writetext** 函数字符串。
- Replication Server 支持三种创建 **rs_writetext** 函数字符串的输出格式：RPC、**writetext** 和 **none**。

使用 RPC 方法

使用 RPC 方法创建 **rs_writetext** 函数字符串时，Replication Server 反复执行远程过程调用，每执行一次过程，最多可以提供 255 个字节的 *text*、*unitext* 或 *image* 值。

在 RPC 中传递数据时，*text* 或 *unitext* 数据使用 *varchar* 参数；*image* 数据使用 *varbinary* 参数。Replication Server 可确保数据块在 *text* 或 *unitext* 列的字符界限上分区。如果正在使用单字节的字符集，将以 255 个字节的大块发送数据。

每次 Replication Server 执行 RPC 时，如果后面还有数据，就将 *rs_last_text_chunk* 系统变量 (*int*) 设置为 0，如果是最后一次对该 *text* 列执行 RPC，则设置为 1。

- 如果主数据库中使用了 **writetext** 记录选项，另一个 *int* 系统变量 *rs_writetext_log* 将被设置为 1；如果主数据库中未使用该记录选项，则设置为 0。
- 可以通过使用 *new* 或 *old* 修饰符访问数据行中其它列的值。如果在主数据库中使用了 Transact-SQL **insert** 命令，则必须使用 *new* 修饰符。
- 可以使用 *text_status* 修饰符来检索 *text*、*unitext* 或 *image* 列的状态。有关 *text_status* 修饰符的说明，请参见“**rs_datarow_for_writetext**”。

使用 **writetext** 方法

用来创建 **rs_writetext** 函数字符串的 **writetext** 方法提供了下表中所示的选项，这些选项用于指定复制数据库中的记录行为。

表 40. writetext 记录选项

记录选项	说明
use primary log	如果在主数据库事务日志中指定了记录选项，则在复制数据库事务日志中记录数据。如果主数据库事务日志中未指定记录，则不记录。
with log	在复制数据库事务日志中记录数据。
no log	不在复制数据库事务日志中记录数据。

rs_sqlserver_function_class 的缺省函数字符串使用 **use primary log** 选项。

使用 **none** 方法

rs_writetext 函数字符串的 **none** 输出模板选项指示 Replication Server 不要使用 Client-Library 函数 **ct_send_data** 来更新 *text*、*unitext* 或 *image* 列值。此选项为在异构环境中使用 *text*、*unitext* 或 *image* 列提供了必要的灵活性。

有关详细信息，请参见《Replication Server 管理指南第二卷》。

另请参见

- rs_get_textptr （第 414 页）
- rs_textptr_init （第 433 页）
- rs_datarow_for_writetext （第 402 页）

Adaptive Server 命令和系统过程

列出与 Replication Server 一起使用的 Adaptive Server 命令和系统过程。

dbcc dbrepair

Transact-SQL 命令，清除脱机复制数据库的辅助截断点。

语法

```
dbcc dbrepair(database_name, ltmignore)
```

参数

- **database_name** – 要为其清除辅助截断点的数据库的名称。
- **ltmignore** – 停用指定数据库中的辅助截断点。

用法

- **dbcc dbrepair** 可清除脱机数据库的辅助截断点；带有 **ignore** 选项的 **dbcc settrunc** 可清除联机数据库的辅助截断点。
- Sybase 建议您在开始升级之前清除事务日志和复制数据库的辅助截断点。如果尚未执行这两项任务，Adaptive Server 不允许在升级后将数据库联机。
- 如果在升级之前没有清除事务日志和辅助截断点，请使用 **dbcc dbrepair** 以使 Adaptive Server 能够将数据库联机。
  在运行 **dbcc dbrepair** 之前：
  1. 在脱机数据库上启动 RepAgent 线程。
  2. 清除事务日志。

  如果在运行 **dbcc dbrepair** 之前没有清除事务日志，日志中的所有事务将会丢失。

另请参见

- dbcc settrunc （第 447 页）

dbcc gettrunc

Transact-SQL 命令，检索有关 Adaptive Server 数据库的当前 RepAgent 信息。

语法

```
dbcc gettrunc
```

用法

- **dbcc gettrunc** 用于启用了 RepAgent 的数据库。
- **dbcc gettrunc** 命令返回一行，包含如下所示的各列：

表 41. dbcc gettrunc 返回的列

列名 **RepAgent**	内容
secondary trunc page	在数据库日志中不被截断的第一页
secondary trunc state	值为以下一项： • 1 – Adaptive Server 不会将位于截断页上或截断页后的日志截断 • 0 – Adaptive Server 忽略截断页
db rep stat	屏蔽，构成方式如下： • 0x01 – 辅助截断页有效 • 0x02 – 数据库至少包含一个显式复制的表 • 0x04 – 数据库包含复制存储过程 • 0x08 – 将所有内容都复制到备用数据库 • 0x10 – 将 L1 复制到备用数据库 • 0x80 – 在执行 HA 故障切换后，Replication Agent 自动重新启动 仅限于 RepAgent： • 0x20 – RepAgent 已启用 • 0x40 – 自动启动 RepAgent
generation id	数据库生成 ID
database id	数据库的 Adaptive Server ID 号
database name	数据库名
ltl version	RepAgent：日志传送语言 (LTL) 版本

注意： 因为迄今为止仅在 Adaptive Server 12.0 版和更高版本中实现了支持级别 L1，所以将 L1 复制到备用数据库与将所有内容复制到备用数据库并无区别。有关详细信息，请参见 **sp_reptostandby**。

另请参见

- admin get_generation （第 52 页）
- dbcc settrunc （第 447 页）

dbcc settrunc

Transact-SQL 命令，修改 Adaptive Server 数据库的辅助截断点信息。

语法

```
dbcc settrunc('ltm', {'valid' | 'ignore'})
dbcc settrunc('ltm', 'gen_id', db_generation)
dbcc settrunc('ltm', {'begin' | 'end',)
```

参数

- **有效的** – 指示 Adaptive Server 考虑辅助截断点。此选项可防止 Adaptive Server 截断尚未传送到 Replication Server 的事务日志记录。
- **ignore** – 指示 Adaptive Server 忽略辅助截断点。这样，Adaptive Server 就可以截断 RepAgent 尚未传送到 Replication Server 的日志记录。
- **gen_id** – 指示 Adaptive Server 重置日志中的数据库生成号。
- **db_generation** – 新的数据库生成号。恢复转储后，该数值会递增，以防止 Replication Server 将新事务作为重复事务而拒绝。

  警告！ 如果 RepAgent 正在运行，则不能执行 **dbcc settrunc**。
- **begin** – 将辅助截断点 (STP) 设置到日志的开头。
- **end** – 将 STP 设置到日志的末尾。

用法

- **dbcc settrunc** 用于启用了 RepAgent 的数据库。
- 对于包含要复制的主数据的 Adaptive Server 数据库或存储复制存储过程的数据库，辅助截断点必须是 **valid**。
- 如果辅助截断点是 **valid**，Adaptive Server 不会截断 Replication Server 尚未从 RepAgent 接收到的日志记录。
- 如果在很长时间内未修改辅助截断点，日志可能会被填满，应用程序将无法继续运行。关闭 Replication Server 和 RepAgent 之后，可以将辅助截断点更改为 **ignore**，这样就可以截断日志，从而使应用程序可以继续运行。然后，使用 **rs_zeroltm** 过程将定位符值重置为零 (0)。不过，请注意以下警告：

  警告！ 如果您将辅助截断点设置为 **ignore**，然后截断日志，则复制数据将是不正确的。您必须重新创建预订，通过执行 **rs_subcmp** 调和预订，或者装载数据库和事务转储并重放丢失的事务。有关重放丢失事务的说明，请参见《Replication Server 管理指南第二卷》。在恢复协调的转储后，应递增数据库生成号。使用 **admin get_generation** 可查找当前的生成号。

有关运行此存储过程的详细信息，请参见 **rs_zeroltm**。

- 恢复后递增数据库生成号可防止 Replication Server 拒绝新的日志记录。有关重装协调的转储的信息，请参见《Replication Server 管理指南第二卷》。
- 如果主 Replication Server 不能接受事务，并且主数据库事务日志已满而必须截断，您可能需要关闭辅助截断点并截断日志，才能使 Adaptive Server 事务可以继续。在这种情况下，可以使用 **dbcc settrunc('ltm', 'ignore')** 关闭 Replication Agent，然后关闭数据库中的辅助截断点。

  使用 **dbcc settrunc** 后，您必须使用 **rs_zeroltm** 存储过程将数据库的定位符值重置为 0。否则，存储在 *rs_locater* 系统表中的日志页可能变得无效。这样，启动 RepAgent 可能导致 Adaptive Server 注册数据损坏，并产生诸如 605 和 813 之类的错误。
- 在您关闭辅助截断点后执行的事务不会被传送到 Replication Server。因此，主数据库和复制数据库可能不同步。

  由于这个原因，在截断日志和 Replication Server 成功启动之后，您可能必须更改复制定义、删除并重新创建预订，然后重新实现复制数据库中的数据。在重新实现数据之前，新列将为 null。

  如果相对来说只有少量事务没有传送到 Replication Server，您可以选择使用 **rs_subcmp** 程序来协调主数据库和复制数据库。

另请参见

- admin get_generation （第 52 页）
- dbcc dbrepair （第 445 页）
- rs_subcmp （第 559 页）
- rs_zeroltm （第 550 页）
- sp_config_rep_agent （第 456 页）

set replication

Transact-SQL 命令，允许或禁止将数据定义语言 (DDL) 和/或数据操纵语言 (DML) 命令复制到当前 **isql** 会话的备用数据库。

语法

```
set replication [on | force_ddl | default | off]
```

参数

- **on** – 如果 **sp_reptostandby** 设置为“none”，则为标记 **sp_setreptable** 的表启用 DML 命令的复制。如果 **sp_reptostandby** 设置为“L1”或“all”，则对备用数据库启用 DML 和 DDL 命令的复制。这是缺省设置。
- **force_ddl** – 始终允许复制当前会话的 DDL 命令。如果 **sp_reptostandby** 设置为“L1”或“all”，则为所有用户表复制 DML 命令。如果 **sp_reptostandby** 设置为“none”，则为标有 **sp_setreptable** 的表复制 DML 命令。

注意： 从 Replication Server 12.0 版开始，在命令 **set replication force_ddl** 中使用的 **force_ddl** 不再是保留字。这并不影响 **set replication force_ddl** 的功能；只是在其它对象名中使用 **force_ddl** 时，不再需要使用双引号。

- **default** – 关闭 **force_ddl** 并使 **set replication** 的状态恢复为“on”（缺省值）。
- **off** – 关闭当前会话的已标记表和用户存储过程的复制。不将任何 DML 命令和 DDL 命令复制到备用数据库或复制数据库。

用法

- **set replication** 要求 Adaptive Server 11.5 版或更高版本的数据库。

权限

set replication 要求“sa”或“dbo”权限以及 **replication_role**。

另请参见

- sp_reptostandby （第 477 页）
- sp_setreptable （第 493 页）

set repmode

在会话级允许或禁止将 **update**、**delete**、**insert select** 或 **select into** 作为 SQL 语句进行复制。

语法

```
set repmode { “on” SQLDML_option | “never” | “off” |
‘threshold’, ‘value’ }
```

```
SQLDML_option ::= { U | D | I | S }
```

参数

- **SQLDML_option** – 以下 DML 操作的任意组合：
  - U – **update**
  - D – **delete**
  - I – **insert select**
  - S – **select into**

  使用 **set repmode** 定义的 SQL 复制设置将覆盖使用 **sp_setrepdbmode** 或 **sp_setrepdefmode** 定义的 SQL 复制设置。
- **on** – 启用指定的 DML 操作的 SQL 复制。
- **off** – 删除 SQL 语句的会话级复制设置，并恢复为数据库级或表级设置。

- **never** – 指定不复制 SQL 语句。

示例

- **示例 1** – 若要在会话期间仅将 **select into** 和 **delete** 作为 SQL 语句进行复制，请使用：

```
set repmode on 'DS'
```

- **示例 2** – 若要在会话期间禁用 SQL 语句复制，而无论数据库或表级设置是什么，请使用：

```
set repmode never
```

- **示例 3** – 此示例说明了会话级设置如何覆盖对象级设置。此示例使用 SQL 语句仅复制 **update** 语句：

```
set repmode on 'U'
go
sp_setrepdefmode tabname, on, 'UDI'
go
```

- **示例 4** – 此示例显示如何在会话级将阈值定义为 1000 行：

```
set repmode ‘threshold’, ‘1000’
go
```

用法

- 可以在登录时（使用“登录触发器”）或批处理开始时设置会话级选项。会话设置将覆盖表或数据库设置。
- 会话级设置仅在会话期间处于活动状态。如果在存储过程或触发器内设置选项，在存储过程或触发器执行终止时，这些设置将恢复为表级或数据库级设置。

另请参见

- sp_setrepdbmode （第 486 页）
- sp_setrepdefmode （第 488 页）

set repthreshold

指定复制的 SQL 语句必须至少影响多少行才会为会话激活 SQL 语句复制。

语法

```
set repthreshold value
```

参数

- **value** – 指定复制的 SQL 语句必须至少影响多少行才会为会话激活 SQL 语句复制。

示例

- **示例 1** – 此示例显示如何在未在数据库级和表级设置任何阈值的情况下在会话级将阈值定义为 23，或者覆盖表级和数据库级的阈值设置：

```
set repthreshold 23
go
```

- **示例 2** – 此示例显示如何在会话级将阈值重置为缺省值 50：

```
set repthreshold 0
go
```

- **示例 3** – 您可以在 Adaptive Server 存储过程内调用 **set repthreshold**。此示例显示如何创建 **set_rep_threshold_23** 存储过程并在 **my_proc** 存储过程中进行调用：

  1. 创建 **set_rep_threshold_23** 存储过程：

```
create procedure set_rep_threshold_23
as
set repthreshold 23
update my_table set my_col = 2  (statement 2)
go
```

  2. 创建 **my_proc** 存储过程：

```
create procedure my_proc
as
update my_table set my_col = 1   (statement 1)
exec set_rep_threshold_23
update my_table set my_col = 3   (statement 3)
go
```

  3. 执行 **my_proc** 以调用 **set_repthreshold_23**：

```
exec my_proc
go
```

  在 **my_proc** 存储过程内，首先执行语句 1，阈值为 50。然后执行语句 2，阈值为 23。接下来执行语句 3，阈值为 50，这是因为 **set repthreshold 23** 命令仅在执行 **set_rep_threshold_23** 过程时有效。

- **示例 4** – 此示例显示如何使会话级阈值可以导出。因此，您可以将某个过程的 **export_options** 设置设为“on”，然后设置 SQL 语句复制阈值，使外部作用域中的过程使用由该存储过程设置的 SQL 语句复制阈值。

  1. 创建 **set_repthreshold_23** 存储过程并将 **export_options** 设置为 on：

```
create procedure set_repthreshold_23
as
set repthreshold 23             (statement 4)
set export_options on
```

```
update my_table set my_col = 2  (statement 2)
go
```

2. 创建 **my_proc** 存储过程：

```
create procedure my_proc
as
update my_table set my_col = 1   (statement 1)
exec set_rep_threshold_23
update my_table set my_col = 3   (statement 3)
go
```

3. 执行 **my_proc** 以调用 **set_repthreshold_23**：

```
exec my_proc
go
```

首先执行语句 1，阈值为 50。然后执行语句 2，阈值为 23。接下来执行语句 3，阈值为 50，这是因为 **set repthreshold 23** 命令的作用域是会话的作用域。

- **示例 5** – 您可以创建登录触发器，以便自动为特定的登录 ID 设置复制阈值。
  - 创建阈值设置为 23 的 **threshold** 存储过程并启用导出：

```
create proc threshold
as
set repthreshold 23
set export_options on
go
```

  - 指示 Adaptive Server 当用户“Bob”登录时自动运行 **threshold** 存储过程：

```
sp_modifylogin Bob, 'login script', threshold
go
```

当 Bob 登录到 Adaptive Server 时，会话的 SQL 语句复制阈值设置为 23。

用法

- 缺省阈值为 50 行，这表示如果 DML 语句至少影响 51 行，Adaptive Server 将使用 SQL 语句复制。若要使用缺省阈值，请将 **threshold** 参数设置为 0。**threshold** 参数的范围是 0 到 10,000。
- 您可以在 Adaptive Server 存储过程内调用 **set repthreshold**。
- 会话级阈值是可以导出的。因此，您可以将某个过程的 **export_options** 设置设为“on”，然后设置 SQL 语句复制阈值，使外部作用域中的过程使用由该存储过程设置的 SQL 语句复制阈值。
- 可以在登录时（使用“登录触发器”）或批处理开始时设置会话级阈值。会话设置将覆盖表或数据库设置。
- 会话级阈值仅在会话期间处于活动状态。如果在存储过程或触发器内设置阈值，在存储过程或触发器执行终止时，这些设置将恢复为表级或数据库级设置。
- 在会话级设置的阈值覆盖表级和数据库级的阈值，并且为任何表设置的阈值覆盖在数据库级设置的阈值。

另请参见

- sp_setrepdbmode （第 486 页）

- sp_setrepdefmode （第 488 页）
- set repmode （第 449 页）

sp_configure 'enable rep agent threads'

在 Adaptive Server 中启用或禁用 RepAgent 线程集成。

语法

```
sp_configure 'enable rep agent threads'[, 1 | 0]
```

参数

- **1** – 为数据服务器启用 RepAgent 集成。
- **0** – 为数据服务器禁用 RepAgent 集成。

用法

- 使用 **sp_configure** '**enable rep agent threads**' 为 Adaptive Server 12.0 版或更高版本的数据库启用 RepAgent。
- 使用不带选项的 **sp_configure 'enable rep agent threads'** 可以显示当前值、缺省值和最近更改过的值。
- 按如下顺序启用 RepAgent：
  - **sp_addserver** – 为 RepAgent 标识 Adaptive Server。此操作只需执行一次。
  - **sp_configure** '**enable rep agent threads**' – 为 RepAgent 启用数据服务器。此操作只需执行一次。
  - **sp_config_rep_agent** – 为 RepAgent 启用数据库。

  有关 **sp_addserver** 的详细信息，请参见《Adaptive Server Enterprise 参考手册》。

权限

sp_configure 要求“sa”或“sso”权限以修改配置参数。

任何用户都可以执行 **sp_configure** 以显示有关参数及其值的信息。

另请参见

- sp_config_rep_agent （第 456 页）

sp_configure 'Rep Agent Thread administration'

显示当前 RepAgent 线程池大小和其它 RepAgent 线程参数的设置。

语法

```
sp_configure 'Rep Agent Thread administration'
```

示例

- **示例 1** – 输入:

  ```
  sp_configure 'Rep Agent Thread administration'
  ```

  您会看到:

  ```
  Group: Rep Agent Thread Administration

  Parameter       Default  Memory  Config  Run     Unit      Type
  Name                     Used    Value   Value

  enable rep      0        0       1       1       switch    dynamic
  agent threads

  replication     4096     8194    4096    4096    memory    dynamic
  agent memory                                     pages(2k)
  size
  ```

  此示例显示，**enable rep agent threads** 是可以打开或关闭的动态参数。对动态参数进行的更改不需要重新启动 RepAgent。

用法

在使用 **sp_configure 'replication agent memory size'** 增加多线程 RepAgent 的内存之前，使用 **sp_configure 'Rep Agent Thread administration'** 检查当前分配给 RepAgent 池的内存。

有关 **sp_configure** 的详细信息，请参见《Adaptive Server Enterprise 参考手册》。

权限

sp_configure 要求“sa”或“sso”权限以修改配置参数。

任何用户都可以执行 **sp_configure** 以显示有关参数及其值的信息。

sp_configure 'replication agent memory size'

更改 Adaptive Server 分配给多线程 RepAgent 的 RepAgent 线程池的内存。

语法

```
sp_configure 'replication agent memory size', repagent_mem_size
```

参数

- **repagent_mem_size** – 要分配给 RepAgent 线程池的内存页数。

示例

- **示例 1** – 要将 RepAgent 线程池大小设置为 8194 页，请输入：

  ```
  sp_configure 'replication agent memory size', 8194
  ```

  您会看到：

  ```
  Group: Rep Agent Thread Administration

  Parameter       Default  Memory  Config  Run     Unit      Type
  Name                     Used    Value   Value

  replication     4096     16430   8194    8194    memory    dynamic
  agent memory                                     pages(2k)
  size
  (1 row affected)
  Configuration option changed. ASE need not be rebooted since the
  option is dynamic.
  Changing the value of 'replication agent memory size' to '8194'
  increases the amount of memory ASE uses by 8236 K.
  ```

用法

在使用 **sp_configure 'replication agent memory size'** 增加多线程 RepAgent 的内存之前，使用 **sp_configure 'Rep Agent Thread administration'** 检查当前分配给 RepAgent 池的内存。

请参见《Replication Server 管理指南第二卷》的“性能调优”的“Multi-Path Replication”的“Multiple Primary Replication Paths”（多个主复制路径）中的“Enabling Multithreaded RepAgent and Multiple Paths for RepAgent”（启用多线程 RepAgent 和 RepAgent 的多个路径）。

有关 **sp_configure** 的详细信息，请参见《Adaptive Server Enterprise 参考手册》。

权限

sp_configure 要求“sa”或“sso”权限以修改配置参数。

任何用户都可以执行 **sp_configure** 以显示有关参数及其值的信息。

sp_config_rep_agent

为 Adaptive Server 数据库的 RepAgent 线程更改或显示配置参数。

语法

```
sp_config_rep_agent [dbname
 [, {'enable', 'repserver_name', 'repserver_username',
'repserver_password'} |
'disable'[, 'preserve secondary truncpt'] |
'rs servername'[, 'repserver_name'] |
'rs username'[, 'repserver_username'] |
'rs password'[, 'repserver_password'] |
'scan batch size'[,'no_of_qualifying_log_records'] |
'scan timeout'[, 'scan_timeout_in_seconds'] |
'retry timeout'[, 'retry_timeout_in_seconds'] |
'skip ltl errors'[, 'true' | 'false'] |
'batch ltl'[, 'true' | 'false'] |
'send warm standby xacts'[, 'true' | 'false'] |
'send buffer size'[, '2K' | '4K' |'8K' | '16K'] |
'connect dataserver'[, 'connect_dataserver_name'] |
'connect database'[, 'connect_database_name'] |
'send maint xacts to replicate'[, 'true' | 'false'] |
'send structured oqids'[, 'true' | 'false'] |
'short ltl keywords'[, 'true' | 'false'] |
'security mechanism'[, 'mechanism_name'] |
'unified login'[, 'true' | 'false'] |
'mutual authentication'[, 'true' | 'false'] |
'msg confidentiality'[, 'true' | 'false'] |
'msg integrity'[, 'true' | 'false'] |
'msg replay detection'[, 'true' | 'false'] |
'msg origin check'[, 'true' | 'false'] |
'msg out-of-sequence check'[, 'true' | 'false'] |
'skip unsupported features'[, 'true' | 'false'] |
'schema cache growth factor'[, 'growth_factor_value'] |
'ha failover'[, 'true' |'false'] |
'data limits filter mode'[, 'off' | 'stop' | 'skip' |'truncate'] |
'priority'[, 'priority_value'] |
'startup delay'[, 'delay_value'] |
'net password encryption'[, 'true' | 'false'] |
'cluster instance name'[, 'coordinator' | 'instance_name'] |
'bind to engine'[, engine_number] |
'ltl batch size'[, ltl_batch_size] |
'ltl metadata reduction', {'true' |'false'}
'multithread rep agent', {'true' | 'false'} |
'multipath distribution model {'connection'|'object'} |
'number of send buffers', {'num_of_send_buffers'} |
'max number replication paths', {'max number replication paths value'}
'ddl path for unbound objects', {'all' | 'default'} |
'auto start', {'true' | 'false'}]
```

参数

- ***dbname*** – 要为其配置 RepAgent 的数据库的名称。
- ***enable*** – 将数据库标记为使用 RepAgent，并将辅助截断点设置为有效。

  此命令对 Replication Server 口令进行编码，并将 Replication Server 名称、Replication Server 用户和已编码的口令插入指定数据库的 `sysattributes` 表中。
- ***repserver_name*** – RepAgent 所连接并向其传送日志事务的 Replication Server 的名称。
- ***repserver_username*** – RepAgent 线程用于与 Replication Server 连接的用户名。
- ***repserver_password*** – RepAgent 用于与 Replication Server 连接的口令。

  如果启用了基于网络的安全性，并且您要建立 **unified login**，在数据库中启用 RepAgent 时，必须将 *repserver_password* 指定为 NULL。
- ***rs servername [, repserver_name]*** – RepAgent 所连接并向其传送日志事务的 Replication Server 的新名称或现有名称。
- ***rs username[, repserver_username]*** – RepAgent 线程用于与 Replication Server 连接的新用户名或现有用户名。
- ***rs password[, repserver_password]*** – RepAgent 线程用于与 Replication Server 连接的新口令或现有口令。
- ***disable*** – 取消数据库使用 RepAgent 的标记。使用 **preserve secondary truncpt** 可保留辅助截断点。缺省情况下将辅助截断点设置为 IGNORE；即，禁用它。

  只有在将 Replication Server 降级为更低版本或将主数据库更改为另一状态时，才可使用 **disable**。此命令可截断 `sysattributes` 表中的所有 RepAgent 条目。
- ***scan batch size[, 'no_of_qualifying_records'']*** – 指定每批可发送给 Replication Server 的日志记录的最大数目。达到记录的最大数目时，RepAgent 会向 Replication Server 要求新的辅助截断点。缺省值为 1000 条记录。

  在多路径复制中，RepAgent 要求提供辅助截断点的频率取决于 **scan batch size** 和 **ltl batch size** 的组合。RepAgent 会将辅助截断点请求放入队列，以便在批处理中的日志记录数达到 **scan batch size** 的值时进行处理。但 RepAgent 在将 **ltl batch size** 中指定数字数的 LTL 数据发送到 Replication Server 时，RepAgent 发送器线程只处理队列中的辅助截断点请求。尽管增加 **ltl batch size** 可改善复制性能，但应考虑对辅助截断点请求数的影响，以避免出现主数据库日志由于辅助截断点移动速度不够而变满的情况。
- ***scan timeout[, 'scan_timeout_in_seconds']*** – 指定在 RepAgent 扫描并处理了事务日志中的所有记录，并且 Replication Server 尚未通过发送新的辅助截断点来确认以前发送的记录的情况下，RepAgent 休眠的秒数。经过 **scan timeout** 秒后，RepAgent 再次向 Replication Server 要求辅助截断点。缺省值为 15 秒。

  在 Replication Server 通过发送新的辅助截断点或扩展事务日志来确认以前发送的记录之前，RepAgent 会一直查询 Replication Server。

  如果 Replication Server 已经确认所有记录，并且没有新的事务记录到达日志，则 RepAgent 会进入休眠状态，直到事务日志被扩展。

- ***retry timeout[, 'retry_timeout_in_seconds']*** – 指定发生可重试错误或关闭 Replication Server 之后，RepAgent 尝试重新连接到 Replication Server 之前 RepAgent 休眠的秒数。缺省值为 60 秒。
- ***skip ltl errors*** – 指定 RepAgent 是否忽略 LTL 命令中的错误。此选项通常用于恢复模式。如果设置为 true，RepAgent 会先记录然后跳过 Replication Server 返回的有关 **distribute** 命令的错误。如果设置为 false，RepAgent 将在发生这些错误时关闭。缺省值为 false。
- ***batch ltl*** – 指定 RepAgent 是成批地将 LTL 命令发送到 Replication Server，还是每次发送一个命令。如果设置为 true，命令将成批发送。缺省值为 false。
- ***send warm standby xacts*** – 指定 RepAgent 是否将维护用户事务、模式更改和系统事务发送到热备份数据库。只应将此选项用于热备份配置中的当前活动数据库的 RepAgent。缺省值为 false。
- ***send buffer size[, '2K', '4K', '8K', '16K']*** – 控制 RepAgent 用于与 Replication Server 通信的发送缓冲区的大小。增大发送缓冲区的大小将减少 RepAgent 与 Replication Server 的通信次数，但是将增大使用的内存量。

  缺省值为 2K。
- ***connect dataserver[, 'connect_dataserver_name']*** – 指定当以恢复模式连接到 Replication Server 时 RepAgent 使用的数据服务器的名称。这是 RepAgent 用于 **connect source** 命令的数据服务器名称；它通常是主数据库的数据服务器。
- ***connect database[, 'connect_database_name']*** – 指定当以恢复模式连接到 Replication Server 时 RepAgent 使用的临时数据库的名称。这是 RepAgent 用于 **connect source** 命令的数据库的名称；它通常是主数据库。
- ***send maint xacts to replicate*** – 指定 RepAgent 是否应该将维护用户的记录发送到 Replication Server，以便分发到预订节点。缺省值为 false。
- ***send structured oqids*** – 指定 RepAgent 以结构化标识形式（可节省 LTL 中的空间，因而可增加吞吐量）还是以二进制字符串形式发送原始队列 ID (OQID)。缺省值为 false。
- ***short ltl keywords*** – 指定 RepAgent 是否向 Replication Server 发送 LTL 的缩写形式，从而节省空间并减少发送的数据量。缺省值为 false。
- ***security mechanism [, 'mechanism_name']*** – 指定 RepAgent 连接到 Replication Server 时所用的基于网络的安全性机制。
- ***unified login*** – 如果启用基于网络的安全系统，则指定 RepAgent 是使用安全认证还是使用口令来连接其它服务器。缺省值为 false。
- ***mutual authentication*** – 指定 RepAgent 在连接到 Replication Server 时是否应该要求相互鉴定检查。缺省值为 false。此选项尚未实现。
- ***msg confidentiality*** – 指定是否对所有发送到 Replication Server 的消息进行加密。缺省值为 false。
- ***msg integrity*** – 指定是否应该检查与 Replication Server 交换的所有消息是否被篡改。缺省值为 false。
- ***msg replay detection*** – 指定是否应该检查从 Replication Server 接收的消息，以确保它们未被截取和重放。缺省值为 false。

- ***msg origin check*** – 指定是否检查从 Replication Server 接收的每条消息的来源。缺省值为 false。
- ***msg out-of-sequence check*** – 指定是否检查从 Replication Server 接收的消息的顺序。缺省值为 false。
- ***skip unsupported features*** – 指示 RepAgent 跳过 Replication Server 不支持的 Adaptive Server 功能的日志记录。如果 Replication Server 的版本低于 Adaptive Server，则通常使用此选项。缺省值为 false。
- ***schema cache growth factor[, 'growth_factor_value']*** – 控制表模式或存储过程模式在失效前可以在 RepAgent 模式高速缓存中驻留的时间。值越大，驻留时间越长，需要的内存也就越多。范围是从 1 到 10。缺省值为 1。
- ***ha failover*** – 指定在已安装 Sybase 故障切换的情况下，RepAgent 是否在服务器故障切换后自动启动。缺省值为 true。
- ***data limits filter mode[, 'off' | 'stop' | 'skip' | 'truncate']*** – 指定 RepAgent 先如何处理包含新的、更宽的列和参数或更大的列和参数计数的日志记录，然后再尝试将其发送到 Replication Server。

  - **off** – RepAgent 允许所有日志记录通过。
  - **stop** – RepAgent 在遇到包含宽数据的日志记录时关闭。
  - **skip** – RepAgent 跳过包含宽数据的日志记录，并在错误日志中张贴一条消息。
  - **truncate** – RepAgent 截断宽数据，使其符合 Replication Server 能够处理的最大限制。

  警告！ Sybase 建议不要将 **data_limits_filter_mode, off** 设置用于 Replication Server 12.1 版或更低版本，因为这样可能会导致 RepAgent 跳过或截断宽数据或者停止运行。

  data limits filter mode 的缺省值取决于 Replication Server 的版本号。对于 Replication Server 12.1 版及更低版本，缺省值为 stop。对于 Replication Server 12.5 版及更高版本，缺省值为 off。
- ***priority[, 'priority_value']*** – 设置个别 RepAgent 的相对优先级值。**priority** 值的范围是 0 到 7，其中值为 0 表示最高优先级。缺省值为 5。

  注意： Sybase 建议您不要将 **priority** 的值设置为 0，因为这样做可能会降低性能。
- ***startup delay[, 'delay_value']*** – 这会将 RepAgent 自动启动延迟指定的一段时间，从而使 Replication Server 能够在 RepAgent 尝试与其连接之前继续运行。缺省情况下，RepAgent 在自动启动过程中不会出现任何延迟。设置一个值（以秒为单位）以使 RepAgent 启动延迟指定的秒数。缺省值为 0 秒。
- ***net password encryption*** – 指定与远程服务器的连接是使用客户端口令加密握手启动的，还是使用常规未加密口令握手序列启动的。缺省值为 true。
- ***cluster instance name[, 'coordinator' | 'instance_name']*** – 控制在其中启动 RepAgent 的实例。缺省情况下，RepAgent 会在充当协调器角色的实例中启动。但是，您可以将 RepAgent 配置为在集群中的任意已声明实例中启动。
- ***bind to engine[, engine_number]*** – 将 RepAgent 执行限定为指定的引擎号。可通过在专用引擎或未充分利用的引擎上运行 RepAgent 来提高其性能。**engine_number**

值的范围是 -1 到 (**max online engines** - 1)。缺省值为 -1，表示 RepAgent 可以在任何引擎上运行。

注意： **bind to engine** 子句不会将其它用户任务或系统任务限定在指定引擎号上运行。

- ***ltl batch size[, ltl_batch_size]*** – 设置 RepAgent 可发送到 Replication Server 的一批 **LTL** 数据的最大大小（以字节为单位）。**ltl_batch_size** 值的范围是 16,384 到 2,147,483,647 个字节。缺省值为 16,384 个字节。

  可通过将 LTL 批次大小增加到较大的数字来提高 RepAgent 的性能。在每个 LTL 批次结束时，RepAgent 将检查上一批次中是否有错误。通过增加 LTL 批次大小，将会减少 RepAgent 检查 LTL 错误的次数。
- ***ltl metadata reduction*** – 设置为 true 可在 RepAgent 中启用表元数据减少并在 Replication Server 中自动启用执行程序命令高速缓存。缺省值为 false。
- ***multithread rep agent*** – 设置为 true 可启用多线程 RepAgent，多线程 RepAgent 将对 RepAgent 扫描程序和发送器活动使用单独的线程，这是构建多个主复制路径的前提条件。缺省值为 false。
- ***multipath distribution model*** – 设置 RepAgent 的多路径分配模式，其中：
  - **connection** – 按连接分布。RepAgent 按照唯一的系统进程 ID (spid) 和可用复制路径数通过某个复制路径分配事务。
  - **object** – （缺省值）按对象绑定分布。RepAgent 将对象（如表和存储过程）绑定到特定复制路径以并行启用这些对象的复制。

注意： 如果您将按对象绑定分布更改为按连接分布，RepAgent 将忽略所有对象绑定并显示警告。如果恢复为按对象绑定分布并重新启动 RepAgent，则会保留对象绑定。

- ***max number replication paths*** – 设置您允许 RepAgent 用于通过多个复制路径从主数据库中复制数据的最大路径数。RepAgent 会为每个 RepAgent 路径生成一个 RepAgent 发送器线程。

  有效值的范围：1 到 MAXINT 的值，MAXINT 的值为 2,147,483,647 个路径。缺省值为 1。

  对于绑定到路径的复制对象，如果 **max number replication paths** 小于路径数，RepAgent 会报告错误并终止。

  要构建多个主复制路径，请使用 **multithread rep agent** RepAgent 参数启用多线程 RepAgent。
- ***number of send buffers*** – 设置您在配置多路径复制时，多线程 RepAgent 的扫描程序和发送器任务可以使用的最大发送缓冲区数量。

  有效值的范围：50 到 MAXINT 的值，MAXINT 的值为 2,147,483,647 个缓冲区。缺省值为 50 个缓冲区。

要构建多个主复制路径，请使用 **multithread rep agent** RepAgent 参数启用多线程 RepAgent。

- ***ddl path for unbound objects*** – 在多路径复制环境中通过所有路径或缺省路径发送出站对象的 SQL 和 DDL 语句。缺省设置为 all。
- ***auto start*** – 指定 Adaptive Server 重新启动并恢复数据库时，RepAgent 是否自动启动。设置为 true 以便 RepAgent 在您重新启动 Adaptive Server 时自动启动。缺省值为 false。

示例

- **示例 1** – 为 pubs2 数据库启用 RepAgent。RepAgent 使用“repusr1”和口令“reppwd1”连接到“repsvr1”：

```
sp_config_rep_agent pubs2, 'enable', 'repsvr1',
  'repusr1', 'reppwd1'
```

- **示例 2** – 显示 pubs2 数据库的配置信息：

```
sp_config_rep_agent pubs2
```

```
Parameter Name                Default    Config Value     Run Value
----------------------      ---------  ---------------  ---------
----
priority                        5            5                5
trace flags                     0            0                0
scan timeout                    15           15               15
retry timeout                   60           60               60
rs username                    n/a         rs1_user         rs1_user
batch ltl                       true         true             true
rs servername                   n/a          rs1              rs1
send buffer size                2k           4k               4k
trace log file                  n/a          n/a              n/a
connect database                n/a          n/a              pdb1
connect dataserver              n/a          n/a              pds1
scan batch size                 1000         1000             1000
security mechanism              n/a          n/a              n/a
msg integrity                  false        false            false
unified login                  false        false            false
schema cache growth factor      1            1                1
skip ltl errors                false        false            false
msg origin check               false        false            false
short ltl keywords             false        false            false
msg confidentiality            false        false            false
data limits filter mode         stop         stop             stop
msg replay detection           false        false            false
mutual authentication          false        false            false
send structured oqids          false        false            false
send warm standby xacts        false        false            false
msg out-of-sequence check      false        false            false
skip unsupported features      false        false            false
send maint xacts to replicate  false        false            false
net password encryption         true         true             true
startup delay                   0            5                5
cluster instance name     coordinator  coordinator     coordinator
```

```
bind to engine                   -1           2                2
ltl batch size                  16384        16384            16384
```

- **示例 3** – 显示特定参数的值：

```
sp_config_rep_agent pubs2, 'scan batch size'
```

```
Parameter Name              Default      Config Value     Run Value
--------------------------  ---------    ---------------  ---------
scan batch size               1000       1000             1000
```

- **示例 4** – 为 pubs2 数据库将 **scan_timeout** 设置为 60 秒：

```
sp_config_rep_agent pubs2, 'scan timeout', '60'
```

- **示例 5** – 将 RepAgent 配置为等待 50 秒后再启动：

```
sp_config_rep_agent pubs2, 'startup delay', '50'
```

- **示例 6** – 在 ASE1 上启动已禁用的 RepAgent：

```
1> sp_config_rep_agent pdb,
    'cluster instance name','ASE1'
2> go
```

```
Parameter Name              Default       Config Value     Run Value
--------------------------  -----------   ---------------  -------
cluster instance name       coordinator   ASE1             ASE1
```

用法

- 使用 **sp_config_rep_agent** 为 Adaptive Server 数据库配置 RepAgent。
- 按以下方式启用 RepAgent：
  - **sp_addserver** – 为 RepAgent 标识 Adaptive Server。此操作只需每屏执行一次。
  - **sp_configure 'enable rep agent thread'** – 为 RepAgent 配置数据服务器。此操作只需每屏执行一次。
  - **sp_config_rep_agent** – 为 RepAgent 配置数据库。

  有关 **sp_addserver** 的详细信息，请参见《Adaptive Server Enterprise 参考手册》。
- 在使用 **sp_config_rep_agent** 配置参数之后，必须使用 **sp_start_rep_agent** 重新启动 RepAgent 才能使新的参数生效。
- 如果您执行不带参数的 **sp_config_rep_agent**，Adaptive Server 将显示已为 RepAgent 启用的所有数据库的缺省值、配置值和运行期值。

  如果您只输入 *dbname*，Adaptive Server 将显示指定数据库的缺省值、配置值和运行期值。
- 由 **sp_config_rep_agent** 指定的属性存储在数据库的 *sysattributes* 表中，具有 *RA* 特性类。
- 在数据服务器上使用 **sp_configure** 启用 RepAgent 后，使用 **sp_config_rep_agent** 设置 RepAgent 配置参数。
- repserver_user 必须具有 **connect source** 权限。

配置基于网络的安全性

注意： 可以在 Adaptive Server 上使用 **sp_configure** 为 RepAgent 启用基于网络的安全性。有关详细信息，请参见《Adaptive Server Enterprise 系统管理指南》。

- 一种安全机制可能无法支持所有的安全性属性。在 Replication Server 上执行 **admin security_property** 可以检验安全性机制的属性。有关详细信息，请参见 **admin security_property**。
- 为 RepAgent 启用的安全性机制必须与为 Replication Server 启用的安全性机制相同。RepAgent 和 Replication Server 上的安全设置必须兼容。

如果 RepAgent 设置为	Replication Server 上的设置可以为
true	• required 或 • not required
false	not required

- 如果 **unified_login** 设置为 true，在数据库上启用 RepAgent 时，您必须将 **rs_password** 参数指定为 NULL。
- 如果您指定一个或多个安全设置，但不指定安全性机制，则 Adaptive Server 将初始化缺省机制，即 `$SYBASE/$SYBASE_ASE/config/libtcl.cfg` 中 SECURITY 部分的第一个条目。

权限

sp_config_rep_agent 需要“sa”或“dbo”权限或者 replication_role。

另请参见

- sp_configure 'enable rep agent threads' （第 453 页）
- sp_help_rep_agent （第 463 页）
- sp_start_rep_agent （第 495 页）
- sp_stop_rep_agent （第 497 页）

sp_help_rep_agent

显示关于 RepAgent 线程的静态和动态信息。

语法

```
sp_help_rep_agent [dbname[, 'recovery' | 'process' | 'config' |
'scan' | 'security' | 'send' |'all']]
```

参数

- **dbname** – 您要了解其信息的 RepAgent 所在数据库的名称。
- **recovery** – 显示关于 RepAgent 的恢复状态信息。

- **process** – 为单个和多个复制路径显示有关 RepAgent 进程的信息。
- **config** – 为单个和多个复制路径显示有关 RepAgent 的配置信息。
- **scan** – 显示关于 RepAgent 的日志扫描信息。
- **security** – 显示基于网络的安全性机制的当前设置。
- **send** – 显示有关已分配给 RepAgent 的发送缓冲区数量的信息。
- **all** – 显示关于连接到指定数据库的 RepAgent 的所有上述信息。

示例

- **示例 1** – 显示恢复信息。

```
sp_help_rep_agent pubs2, 'recovery'
```

  您会看到:

```
Replication  Agent Recovery Status

dbname   connect     connect   status    rs servername  rs username
         dataserver  database
------   ----------  --------  ------    -------------  -----------
pubs2    sqlserver1   pubs2    scanning  repsvr1        repusr1
```

- **示例 2** – 显示进程信息。

```
sp_help_rep_agent pubs2, 'process'
```

  您会看到:

```
Replication Agent Process Status

dbname  spid   sleep status  retry count   last error
-----   ----   ------------  ----------    ----------
pubs2   40     not sleeping  0             0
```

- **示例 3**

  显示多线程 RepAgent 的进程信息。

```
sp_help_rep_agent pubs2, 'process'
```

  您会看到:

```
Replication Agent Process Status

dbname  spid   sleep status   state     retry count   last error
-----   ----   ------------   --------  ----------    ----------
pubs2   12     not sleeping   sleeping  0             0


Replication Agent (sender) Process status

dbname  spid   sleep status   state     retry count   last error
-----   ----   -------------  -----     ----------    ----------
pubs2   13     connect retry  sleeping  3             0
pubs2   14     connect retry  sleeping  3             0
```

- **示例 4**

  显示有关由扫描程序任务生成的发送缓冲区和由发送器任务发送给 Replication Server 的发送缓冲区的信息。仅当启用 **sp_config_rep_agent** 的 **multithread rep agent** 参数时才会显示该信息。

```
sp_help_rep_agent pubs2, 'send'
```

  您会看到：

```
Replication Agent Send Status

dbname  sender_spid   total_send_buffers   send_buffers_used
------  -----------   ------------------   -----------------
pubs2   13            40                   0
```

- **示例 5** – 显示扫描信息。

```
sp_help_rep_agent pubs2, 'scan'
```

  您会看到：

```
Replication Agent Scan status

dbname  start_marker  end_marker  current_marker
------  ------------  ----------  --------------
pubs2   (472675,13)   (278622,0)  (265736,16)

log_recs_scanned   oldest_trans.
-----------------  -------------
0                  (-1,0)
```

- **示例 6**

  显示有关由扫描程序任务生成的发送缓冲区和由发送器任务发送给 Replication Server 的发送缓冲区的信息。仅当启用 **sp_config_rep_agent** 的 **multithread rep agent** 参数时才会显示该信息。

```
sp_help_rep_agent pubs2, 'send'
```

  您会看到：

```
Replication Agent Send Status

dbname  sender_spid   total_send_buffers   send_buffers_used
------  -----------   ------------------   -----------------
pubs2   13            40                   0
```

用法

- **sp_help_rep_agent** 用于启用了 RepAgent 的数据库。
- 如果执行不带参数的 **sp_help_rep_agent**，Adaptive Server 将显示启用了 RepAgent 的所有数据库的有关信息。
- 表 42. sp_help_rep_agent 'recovery' 输出的列说明 （第 466 页） 介绍了 **sp_help_rep_agent'recovery**' 系统过程的输出。

表 42. sp_help_rep_agent 'recovery' 输出的列说明

列	说明
dbname	包含存档日志的数据库的名称，其数据在恢复期间被传送到 Replication Server。
connect data-server	原始数据服务器的名称，该数据服务器所使用的数据库的事务日志已按正常模式传送到 Replication Server。传递到 Replication Server 的 LTL **connect source** 命令中包含此信息。
connect database	原始数据库的名称，其事务日志已按正常模式传送到 Replication Server。传递到 Replication Server 的 LTL **connect source** 命令中包含此信息。
status	表示 RepAgent 活动。状态值有: • “not running” – RepAgent 没有运行。 • “not active” – RepAgent 不在恢复模式中。 • “initial” – RepAgent 正在恢复模式中进行初始化。 • “end of log” – RepAgent 正处于恢复模式下，并已到达事务日志的结尾。 • “unknown” -- 不是以上任何一种。
rs servername	RepAgent 正向其传送信息的 Replication Server 的名称。使用此选项可覆盖 `sysattributes` 设置。
rs username	RepAgent 用于登录到 Replication Server 的登录名。使用此选项可覆盖 `sysattributes` 设置。

- 表 43. sp_help_rep_agent 'config' 输出的列说明 （第 466 页） 介绍了 **sp_help_rep_agent 'config'** 系统过程的输出。

表 43. sp_help_rep_agent 'config' 输出的列说明

列	说明
dbname	包含存档日志的数据库的名称，其数据在恢复期间被传送到 Replication Server。
auto start	如果 RepAgent 在服务器启动期间自动启动，则值为“true”。否则，值为“false”。
rs servername	RepAgent 正向其传输日志事务的 Replication Server 的名称。
rs username	RepAgent 线程用于登录 Replication Server 的登录名。在 Replication Server 中该登录名必须已被授予 **connect source** 权限。
scan batch size	每批发送到 Replication Server 的日志记录的最大数目。 缺省值为 1000。

列	说明
scan timeout	在 RepAgent 扫描并处理了事务日志中的所有记录，并且 Replication Server 尚未通过发送新的辅助截断点来确认以前发送的记录的情况下，RepAgent 休眠的秒数。 缺省值为 15 秒。
retry timeout	发生可重试错误或 Replication Server 关闭后，RepAgent 在尝试重新连接到 Replication Server 之前休眠的秒数。 缺省值为 60 秒。
skip ltl errors	如果 RepAgent 忽略 LTL 命令中的错误，则值为“true”。如果发生这些错误后 RepAgent 关闭，则值为“false”。在恢复模式下，**skip ltl errors** 通常设置为“true”。 缺省值为“false”。
batch ltl	如果 RepAgent 批处理 LTL 命令并将它们发送到 Replication Server，则值为“true”。如果 LTL 命令在格式化后立即被发送到 Replication Server，则值为“false”。 缺省值为“false”。
send warm standby xacts	如果 RepAgent 将模式、系统 xact 和所有的更新（包括维护用户所做的更新）提交给 Replication Server 以应用于热备份应用中的备用数据库，则值为“true”。如果 RepAgent 未将更新提交给备用数据库，则值为“false”。 缺省值为“false”。
connect dataserver	当 RepAgent 在恢复模式下运行时，由 RepAgent 连接到 Replication Server 的数据服务器的名称。如果 RepAgent 不是在恢复模式下运行，则包含 *dbname* 数据库的数据服务器名称。
connect database	以恢复模式运行时，RepAgent 将其连接到 Replication Server 的数据库的名称。如果 RepAgent 不是在恢复模式下运行，则包含 *dbname* 数据库名。
send maint commands to replicate	如果 RepAgent 将维护用户的记录发送到复制数据库，则值为“true”。如果 RepAgent 不将维护用户的记录发送到复制数据库，则值为“false”。 缺省值为“false”。
ha failover	指定在安装有 Sybase 故障切换时，RepAgent 是否在发生服务器故障切换后自动启动。 缺省值为“true”。

列	说明
skip unsupported features	指示 RepAgent 跳过 Replication Server 不支持的 Adaptive Server 功能的日志记录。如果 Replication Server 的版本低于 Adaptive Server，则通常使用此选项。 缺省值为“false”。
short ltl keywords	指定 RepAgent 是否向 Replication Server 发送 LTL 的缩写形式，从而节省空间并减少发送的数据量。 缺省值为“false”。
send buffer size	控制 RepAgent 用于与 Replication Server 通信的发送缓冲区的大小。增大发送缓冲区的大小将减少 RepAgent 与 Replication Server 的通信次数，但是将增大使用的内存量。值为“2K”、“4K”、“8K”和“16K”。 缺省值为“2K”。
priority	设置个别 RepAgent 的相对优先级值。**priority** 值的范围是 0 到 7，其中值为 0 表示最高优先级。缺省值为 5。 **注意：** Sybase 建议您不要将 **priority** 值设置为 0。
send structured oqids	指定 RepAgent 以结构化标识形式（可节省 LTL 中的空间，因而可增加吞吐量）还是以二进制字符串形式发送原始队列 ID (OQID)。 缺省值为“false”。
data limits filter mode	指定 RepAgent 先如何处理包含新的、更宽的列和参数或更大的列和参数计数的日志记录，然后再尝试将其发送到 Replication Server。 • **off** – RepAgent 允许所有日志记录通过。 • **stop** – RepAgent 在遇到包含宽数据的日志记录时关闭。 • **skip** – RepAgent 跳过包含宽数据的日志记录，并在错误日志中张贴一条消息。 **data_limits_filter_mode** 的缺省值取决于 Replication Server 版本号。对于 Replication Server 12.1 和更低版本，缺省值为“stop”；对于 Replication Server 12.5 和更高版本，缺省值为“off”。
schema cache growth factor	控制表模式或存储过程模式在失效前可以在 RepAgent 模式高速缓存中驻留的时间。值越大，驻留时间越长，需要的内存也就越多。范围是从 1 到 10。 缺省值为 1。
startup delay	RepAgent 启动所延迟的秒数。缺省值为 0。
cluster instance name	在其中启动 RepAgent 的集群实例的名称。缺省值为**“coordinator”**。

列	说明
bind to engine	指定在上面执行 RepAgent 的引擎号。范围是 -1 到 (**max online engines** - 1)，其中 **max online engines** 是 Adaptive Server 配置参数。缺省值为 -1，表示 RepAgent 可以在任何引擎上运行。
ltl batch size	在给定批次中，RepAgent 可以发送到 Replication Server 的 LTL 数据的最大大小（以字节为单位）。最小值和缺省值均为 16,384 个字节。最大值为 2,147,483,647 个字节。
multithread_ rep_agent	指定是否启用多线程 RepAgent。多线程 RepAgent 对 RepAgent 扫描程序和发送器活动使用单独的线程，这是构建多个主复制路径的前提条件。 缺省值为 false。
number_of_ send_buffers	多线程 RepAgent 的扫描程序和发送器任务可以使用的最大发送缓冲区数量。 有效值的范围：1 到 MAXINT 的值，MAXINT 的值为 2,147,483,647 个缓冲区。缺省值为 50 个缓冲区。

- 表 44. sp_help_rep_agent 'process' 输出的列说明 （第 469 页） 介绍了 **sp_help_rep_agent 'process'** 系统过程的输出。

表 44. sp_help_rep_agent 'process' 输出的列说明

列	说明
dbname	包含存档日志的数据库的名称，其数据在恢复期间被传送到 Replication Server。
sleep status	休眠状态值有： • “waiting for rewrite” – RepAgent 正在等待提交两阶段提交事务。 • “end of log” – RepAgent 在日志的末尾，等待日志扩展。 • “connect retry” – RepAgent 正在等待，然后会尝试连接到 Replication Server。 • “not sleeping” – 不是以上任何一种。RepAgent 处于活动状态。
state	状态值有： • “Sleeping” – 挂起了 RepAgent 发送器任务并正在等待活动。 • “Awake” – RepAgent 发送器任务处于活动状态。
retry count	上一次成功连接 Replication Server 以来，RepAgent 尝试连接失败的次数。
spid	数据服务器中的 PID。
last error	上一个 Replication Server 或连接错误的错误号。

- 表 44. sp_help_rep_agent 'process' 输出的列说明 （第 469 页） 介绍了 **sp_help_rep_agent 'send'** 系统过程的输出。

表 45. sp_help_rep_agent 'scan' 输出的列说明

列	说明
dbname	包含存档日志的数据库的名称，其数据在恢复期间被传送到 Replication Server。
sender_spid	RepAgent 发送器任务的 PID。
total_send_buffers	分配给每个发送器任务的发送缓冲区数。
send_buffers_used	发送器任务使用的发送缓冲区数。

- 表 46. sp_help_rep_agent 'scan' 输出的列说明 （第 470 页） 介绍了 **sp_help_rep_agent 'scan'** 系统过程的输出。

表 46. sp_help_rep_agent 'scan' 输出的列说明

列	说明
dbname	包含存档日志的数据库的名称，其数据在恢复期间被传送到 Replication Server。
start marker	标识当前批处理中扫描的第一条记录。
end marker	标识当前批处理中将要扫描的最后一条记录。
current marker	标识当前正被扫描的记录。
log recs scanned	RepAgent 在当前批处理中已扫描的日志记录数目。
oldest transaction	标识当前正被扫描的批处理中最早的一个事务。

- 表 47. sp_help_rep_agent 'security' 输出的列说明 （第 470 页） 介绍了 **sp_help_rep_agent 'security'** 存储过程的输出。

表 47. sp_help_rep_agent 'security' 输出的列说明

列	说明
dbname	包含存档日志的数据库的名称，其数据在恢复期间被传送到 Replication Server。
security mechanism	已启用的安全性机制的名称。
unified login	指定 RepAgent 是尝试使用证书（“true”）还是使用口令（“false”）连接 Replication Server。缺省值为“false”。
mutual authentication	指定 RepAgent 在连接到 Replication Server 时是否执行相互鉴定检查。缺省值为“false”。
msg confidentiality	指定 RepAgent 是否对所有发送到 Replication Server 的数据进行消息加密。缺省值为“false”。

列	说明
msg integrity	指定 RepAgent 是否对所有与 Replication Server 交换的数据执行消息完整性检查。缺省值为“false”。
msg replay detection	指定 RepAgent 是否检查以检测入侵者是否已捕获和重放数据。缺省值为“false”。
msg origin check	指定 RepAgent 是否检验从 Replication Server 发送的数据的来源。缺省值为“false”。
msg out-of-sequence	指定 RepAgent 是否检验来自 Replication Server 的消息的接收顺序是否与其发送顺序相同。缺省值为“false”。
net password encryption	指出是否使用客户端口令加密握手来启动到 Replication Server 的连接。缺省值为“true”。

权限

sp_help_rep_agent 要求“sa”或“dbo”权限或者 **replication_role**。

另请参见

- sp_config_rep_agent （第 456 页）
- sp_start_rep_agent （第 495 页）
- sp_stop_rep_agent （第 497 页）

sp_replication_path

配置和管理主数据库和 Replication Server 之间的替代复制路径。

语法

```
sp_replication_path "dbname", {
'add' "physical_path", "repserver_name", "rs_username",
"rs_password" |
'add', 'logical', 'logical_path',  "physical_path" |
'drop', "physical_path" |
'drop', 'logical', 'logical_path', [,"physical_path"] |
'bind', "object_type", "[table_owner].object_name", "path_name" |
'unbind', "object_type", "object_name", {"path_name" | all} |
'config', "path_name",  "config_parameter", "config_value" |
'list', ['object_type'], ['object_name']
```

参数

- **dbname** – 要为其配置 RepAgent 的数据库的名称。
- **add** – 添加从 *dbname* 到 Replication Server 的替代物理 RepAgent 路径。

  - *physical_path* – 替代 RepAgent 路径的名称。
  - *repserver* – 要从 *dbname* 进行连接的复制的名称。
  - *rs_username* – 具有连接到 *repserver* 的相应权限的用户名。一般情况下，就是维护用户。
  - *rs_password* – *rs_username* 的口令
- **add, logical** – 添加可用于将绑定到物理路径的数据和对象分发到 Replication Server 的逻辑 RepAgent 路径。
  - *logical_path* – 逻辑路径的名称。
- **drop** – 从非缺省主复制路径的物理复制路径中删除作为目标的 Replication Server。
- **drop, logical** – 从逻辑复制路径（例如物理路径）中删除元素。
- **bind** – 将对象与物理或逻辑主复制路径相关联。在复制过程中，绑定对象始终遵循相同的路径。
  - *object_type* – 指定对象类型，对象类型可以是 **table** 或 **sproc**（存储过程）。
  - *[table_owner].object_name* – 表名或存储过程名。

    注意： 如果不指定表所有者，则当对象是表时，绑定仅适用于数据库所有者 `dbo` 所有的表。
  - *path_name* – 物理路径名或逻辑路径名。
- **unbind** – 删除绑定对象与物理或逻辑复制路径之间的关联。
  - *object_type* – 指定对象的类型：**path**、**table** 或 **sproc**（存储过程）。
  - *[table_owner.]object_name* – 要取消绑定的表、存储过程或路径的名称。

    注意： 如果不指定表所有者，则当对象是表时，绑定仅适用于数据库所有者 `dbo` 所有的表。
  - *path_name* | **all** – 指定物理或逻辑路径名或所有路径。如果指定 **path** 作为 *object_type*，则提供路径名作为 *object_name* 并指定 **all** 选项，Replication Agent 会从指定的路径名取消绑定所有对象。
- **config** – 在替代复制路径中设置参数值。
  - *config_parameter* – 为 **rs username** 或 **rs password**
  - *config_value* – 对于 **rs username** 为 *rs_username*，对于 **rs password** 为 *rs_password*。
- **list** – 显示有关绑定和复制对象的信息。
  - *object_type* – 指定对象的类型：**path**、**table** 或 **sproc**（存储过程）。
  - *object_name* – 显示特定对象的绑定关系。当想要指定对象的名称时，必须指定 *object_type*。

示例

- **示例 1** – 创建替代物理复制路径。
  - 使用 RS2 用户 ID 和 RS2_password 创建 PDS 数据服务器中的 pdb 数据库和 RS2 Replication Server 之间的 pdb_1 替代物理复制路径。在 PDS 中，输入：

    ```
    sp_replication_path "pdb", 'add', "pdb_1", "RS2", "RS2_user",
    "RS2_password"
    ```
  - 使用 RS1 用户 ID 和 RS1_password 创建 PDS 数据服务器中的 pdb 数据库和 RS1 Replication Server 之间的 pdb_2 替代物理复制路径。在 PDS 中，输入：

    ```
    sp_replication_path "pdb", 'add', "pdb_2", "RS1", "RS1_user",
    "RS1_password"
    ```

  现在与 pdb 之间有三个物理复制路径：pdb_1、pdb_2 和在创建替代物理复制路径之前必须创建的到 RS1 或 RS2 的现有缺省路径复制路径。
- **示例 2** – 创建 pdb_1 物理路径支持的 logical_1 逻辑路径。在 PDS 中，输入：

  ```
  sp_replication_path 'pdb', 'add', 'logical', 'logical_1', 'pdb_1'
  ```
- **示例 3** – 添加 pdb_2 物理路径以支持现有的 logical_1 逻辑路径。

  ```
  sp_replication_path 'pdb, 'add', 'logical', 'logical_1', 'pdb_2'
  ```
- **示例 4** – 删除 RS1 Replication Server 作为物理路径的目标：

  ```
  sp_replication_path pdb, 'drop', "RS1"
  ```
- **示例 5** – 从逻辑路径中删除物理路径。
  - 从 logical_1 中删除 pdb_1：

    ```
    sp_replication_path 'pdb', 'drop', 'logical', 'logical_1',
    'pdb_1'
    ```
  - 从 logical_1 中删除 pdb_2：

    ```
    sp_replication_path 'pdb', 'drop', 'logical', 'logical_1',
    'pdb_2'
    ```
- **示例 6** – 删除 logical_1 逻辑路径：

  ```
  sp_replication_path 'pdb', 'drop', 'logical', 'logical_1'
  ```
- **示例 7** – 将对象绑定到物理或逻辑复制路径。

  要将：
  - t1 表绑定到 pdb_2 复制路径，请输入：

    ```
    sp_replication_path pdb, 'bind', "table", "t1", "pdb_2"
    ```
  - 属于 owner1 的 t2 表绑定到 pdb_2 复制路径，请输入：

    ```
    sp_replication_path pdb, 'bind', "table", "owner1.t2", "pdb_2"
    ```
  - **sproc1** 存储过程绑定到 pdb_2 复制路径，请输入：

    ```
    sp_replication_path pdb, 'bind', "sproc", "sproc1", "pdb_2"
    ```
  - dt1 维度表对象绑定到 everywhere 逻辑路径，请输入：

```
sp_replication_path pdb, 'bind', "table", "dt1", "everywhere"
```

或者，可以在 *object_name* 中使用星号“*”或百分号“%”通配符或两者的组合来指定要绑定到某一路径的一定范围的名称或匹配字符。例如，要将名称与各种通配符组合相匹配的表绑定到 pdb_2 复制路径，请输入：

- `sp_replication_path pdb, 'bind', 'table', 'a*', "pdb_2"`
- `sp_replication_path pdb, 'bind', 'table', 'au%rs', "pdb_2"`
- `sp_replication_path pdb, 'bind', 'table', 'a*th%s', "pdb_2"`
- `sp_replication_path pdb, 'bind', 'table', 'authors%', "pdb_2"`

- **示例 8** – 从复制路径取消绑定对象。

  要删除：

  - `t1` 表到 pdb_2 复制路径的绑定，请输入：

    ```
    sp_replication_path pdb, 'unbind', 'table', "t1", "pdb_2"
    ```
  - `t1` 表上的所有绑定，请输入：

    ```
    sp_replication_path pdb, 'unbind', 'table', "t1", "all"
    ```
  - 所有对象到 pdb_2 复制路径的绑定，请输入：

    ```
    sp_replication_path pdb, 'unbind', 'path', 'pdb_2', "all"
    ```

- **示例 9** – 更改替代复制路径的口令和用户 ID。

  要将：

  - `pdb_1` 替代复制路径用于连接到 RS1 的用户名更改为“RS1_user”，请输入：

    ```
    sp_replication_path pdb, 'config', "pdb_1", "rs_username",
    “RS1_user”
    ```
  - `pdb_1` 用于连接到 RS1 的口令更改为“january”，请输入：

    ```
    sp_replication_path pdb, 'config', "pdb_1", "rs password",
    “january”
    ```

- **示例 10** – 显示所有绑定对象的路径关系：

```
sp_replication_path 'pdb','list'
go
```

您会看到：

```
Binding              Type     Path
-------------------- -------- --------------------
dbo.dt1              T        everywhere
dbo.sproc1           P        pdb_1
dbo.sproc1           P        pdb_2
dbo.t1               T        pdb_2
dbo.t2               T        pdb_1

(5 rows affected)

Logical Path                  Physical Path
```

```
---------------------------- ---------------------
everywhere                   pdb_1
everywhere                   pdb_2

(2 rows affected)
Physical Path                  Destination
---------------------------- -------------------
pdb_1                          RS2
pdb_2                          RS1

(2 rows affected)
(return status = 0)
```

- **示例 11** – 显示有关所有绑定表的信息：

```
sp_replication_path 'pdb','list','table'
go
```

您会看到：

```
Binding              Type     Path
-------------------- -------- --------------------
dbo.dt1              T        everywhere
dbo.t1               T        pdb_2
dbo.t2               T        pdb_1

(3 rows affected)
(return status = 0)
```

- **示例 12** – 显示有关所有存储过程的信息：

```
sp_replication_path 'pdb','list','sproc'
go
```

您会看到：

```
Binding              Type     Path
-------------------- -------- --------------------
dbo.sproc1           P        pdb_2
dbo.sproc1           P        pdb_1
dbo.sproc2           P        pdb_1

(3 rows affected)
(return status = 0)
```

- **示例 13** – 仅显示有关 **sproc1** 存储过程的信息：

```
sp_replication_path 'pdb','list','sproc','sproc1'
go
```

您会看到：

```
Binding              Type     Path
-------------------- -------- --------------------
dbo.sproc1           P        pdb_2
dbo.sproc1           P        pdb_1

(2 rows affected)
(return status = 0)
```

- **示例 14** – 显示有关所有复制路径的信息：

```
sp_replication_path 'pdb','list','path'
go
```

您会看到：

```
Path                 Type     Binding
-------------------- -------- ----------------------
everywhere           T        dbo.dt1
pdb_1                P        dbo.sproc1
pdb_1                T        dbo.t2
pdb_2                P        dbo.sproc1
pdb_2                T        dbo.t1

(5 rows affected)
Logical Path                   Physical Path
------------------------------ ------------------------
everywhere                     pdb_1
everywhere                     pdb_2

(2 rows affected)
Physical Path                  Destination
------------------------------ ----------------------
pdb_1                          RS2
pdb_2                          RS1

(2 rows affected)
(return status = 0)
```

- **示例 15** – 仅显示有关 pdb_1 物理路径的信息：

```
sp_replication_path 'pdb','list','path','pdb_1'
go
```

您会看到：

```
Path                 Type     Binding
-------------------- -------- --------------------
pdb_1                P        dbo.sproc1
pdb_1                T        dbo.t2

(2 rows affected)
Physical Path                   Destination
------------------------------ -------------------
pdb_1                           RS2

(1 rows affected)
(return status = 0)
```

- **示例 16** – 仅显示有关“logical_1”逻辑复制路径的信息：

```
sp_replication_path 'pdb','list','path','logical_1'
go
```

您会看到：

```
Path                 Type     Binding
-------------------- -------- -------------------------
logical_1           T        dbo.dt1
```

```
(1 rows affected)
Logical Path                   Physical Path
------------------------------ --------------------------
logical_1                      pdb_1
logical_1                      pdb_2

(2 rows affected)
Physical Path                  Destination
------------------------------ --------------------------
pdb_1                          RS2
pdb_2                          RS1

(2 rows affected)
(return status = 0)
```

注意： 您还会看到作为逻辑路径基础的物理路径。

用法

- 必须在主数据库和 Replication Server 之间创建替代主连接并在将对象绑定到路径之前将该连接与主数据库到 Replication Server 的替代 RepAgent 复制路径相关联。请参见“Replication Server”的“性能调优”中的“Multi-Path Replication”。
- 可以将表和存储过程绑定到可为多路径复制创建的替代物理或逻辑路径。
- 在复制过程中，绑定到路径的所有对象始终遵循相同的路径。

权限

sp_replication_path 需要“sa”或“dbo”权限或者 **replication_role**。

sp_reptostandby

标记或取消标记复制到备用数据库的数据库。对于受支持的用户表模式更改和数据更改，允许复制这些更改。

语法

```
sp_reptostandby dbname [, 'L1' | 'all' | 'none'] [, use_index]
```

参数

- **dbname** – 活动数据库的名称。
- **L1** – 将模式复制功能集支持级别设置为 Adaptive Server 12.0 版中引入的支持级别 1。如果将 Adaptive Server 升级为实现更高支持级别（即，**L2**、**L3** 等）的更高版本，则支持级别仍保持为 Adaptive Server 12.0 版支持级别。目前，Adaptive Server 12.0 版和更高版本中仅实现了支持级别 **L1**。

- **all** – 将模式复制功能集支持级别设置为当前 Adaptive Server 实现的最高支持级别。如果将 Adaptive Server 升级到更高版本，则会自动启用较高版本实现的最高支持级别。
- **none** – 取消要复制的所有数据库表的标记，并停止将数据和模式复制到备用数据库。

  注意: 如果您使用带有 **none** 关键字的 **sp_reptostandby** 关闭复制，Adaptive Server 将以排它模式锁定所有用户表，然后写入所有被取消标记为复制的表的日志记录。如果数据库中有许多用户表，这个过程可能非常耗时。
- **use_index** – 将数据库标记为使用 *text*、*unitext*、*image* 或 *rawobjects* 列上的索引，并在未显式标记为要复制的那些表中创建内部索引。

示例

- **示例 1** – 将 *pubs2* 的复制状态设置为 **all** 并对文本和图像指针创建全局索引：

```
sp_reptostandby pubs2,'all','use_index'
```

- **示例 2** – 在数据库级别显示 SQL 语句复制状态：

```
1> sp_reptostandby pubs2
2> go

The replication status for database 'pubs2' is 'ALL'.
The replication mode for database 'pubs2' is ' udis'.
(return status = 0)
```

用法

- **sp_reptostandby** 用于 Adaptive Server 11.5 版或更高版本的数据库。您还必须在活动数据库和备用数据库上启用 RepAgent。
- 将数据操纵语言 (DML) 命令、支持的数据定义语言 (DDL) 命令以及支持的系统过程复制到备用数据库。
- 如果数据库为主数据库，则用户数据库中支持复制的 DDL 命令和系统过程在主数据库中不受支持。

  如果 DDL 命令或系统过程包含口令信息，则使用源 ASE 系统表中存储的密文口令值通过复制环境发送此口令信息。
- **sp_reptostandby** 标记要复制到热备份数据库的数据库。它不启用到复制数据库的复制。
- 在执行 **sp_reptostandby** 并启用热备份后，可通过将个别数据库表的复制状态设置为 **never**，有选择性地关闭这些表的复制。可以使用 **set replication** 命令控制 **isql** 会话的 DDL 命令、DML 命令以及过程的复制。
- 缺省情况下，**sp_reptostandby** 将 *text*、*unitext* 或 *image* 数据标记为 **replicate_if_changed**。不能将其状态更改为 **always_replicate** 或 **do_not_replicate**。
- 如果热备份应用程序包括常规复制，则可能会将 *text*、*unitext* 或 *image* 数据列作为 **always_replicate** 或 **replicate_if_changed** 处理。

    - 如果将 **sp_setreptable** 标记的 *text*、*unitext* 或 *image* 列指定为 **always_replicate**（缺省值），则会将所有 *text*、*unitext* 或 *image* 列作为 **always_replicate** 处理。
    - 如果 **sp_setrepcol** 将 *text*、*unitext* 或 *image* 列指定为 **do_not_replicate** 或 **replicate_if_changed**，则会将所有 *text*、*unitext* 或 *image* 列作为 **replicate_if_changed** 处理。
- 如果数据库包含一个或多个含 *text*、*unitext*、*image* 或 *rawobject* 列的大表，则 **sp_reptostandby** 执行的内部进程可能会花费很长时间。要加快进程的速度，可以使用 **use_index**，它会为未显式标记为要进行复制的表的每个 *text*、*unitext*、*image* 或 *rawobject* 列创建全局性的非聚簇索引。
- 如果使用 **use_index**，在创建非聚簇索引的同时将保持共享表锁。
- 如果在运行 **sp_reptostandby** 时指定了 **none** 选项，并且最初将数据库标记为使用索引进行复制，则会删除为复制创建的所有这些索引。

限制和要求

- 备用数据库的版本级别必须与活动数据库的版本级别相同或更高。两个数据库必须具有相同的磁盘分配、段名和角色。有关详细信息，请参见《Adaptive Server Enterprise 系统管理指南》。
- 不会将登录信息复制到备用数据库。
- 如果备用服务器中没有指定的数据库，则复制包含另一数据库名称的命令或过程将会失败。
- 受支持的 DDL 命令（如 **create table**）不得包含局部变量。
- 有些命令不复制到备用数据库：
    - **select into** 和 **update statistics**
    - 数据库或配置选项，例如 **sp_dboption** 和 **sp_configure**
- 如果数据库是主数据库：
    - 不会复制用户表和用户存储过程。
    - 无法使用 **dump** 或 **load** 实现目标数据库。请使用其它方法，如 **bcp**，可以在其中对数据进行处理以解决不一致问题。
    - 源 ASE 服务器和目标 ASE 服务器必须支持主数据库复制功能。
    - 源 ASE 服务器和目标 ASE 服务器必须具有相同的硬件体系结构类型（32 位版本和 64 位版本可以兼容）和相同的操作系统（不同版本可以兼容）。
- 如果复制了主数据库，则必须在主数据库中执行以下系统过程：
    - **sp_addlogin**
    - **sp_defaultdb**
    - **sp_defaultlanguage**
    - **sp_displaylevel**
    - **sp_droplogin**
    - **sp_locklogin**
    - **sp_modifylogin**

- 不能使用 **drop index** 手动删除为 *text*、*unitext*、*image* 或 *rawobject* 复制创建的索引。只能使用支持的复制存储过程 **sp_reptostandby**、**sp_setreptable** 和 **sp_setrepcol** 来更改复制索引状态。

权限

sp_reptostandby 要求“sa”或“dbo”权限或者 **replication_role**。

另请参见

- set replication （第 448 页）
- sp_setrepcol （第 483 页）
- sp_setreptable （第 493 页）
- sp_setreplicate （第 490 页）
- sp_setrepproc （第 491 页）

支持的 DDL 命令和系统过程

使用 **sp_reptostandby** 启用复制时 Replication Server 在备用数据库上复制的 DDL 命令、Transact-SQL 命令和 Adaptive Server 系统过程。

有一部分命令和存储过程标有星号，这表示它们的复制在 Adaptive Server 12.5 及更高版本中受支持。

支持的 DDL 命令有:

- **alter encryption key**
- **alter key**
- **alter login**
- **alter login profile**
- **alter...modify owner** – Replication Server 将具有不同所有者的表视为不同表。如果您使用 **alter...modify owner** 更改 Adaptive Server 复制表的所有者，则必须对表复制定义进行相关更改。请参见《Replication Server 管理指南第一卷》的“管理复制表”的“修改复制定义”的“更改复制定义”的“可以对复制定义进行的更改”中的“更改表所有者”。
- **alter table**
- **create default**
- **create encryption key**
- **create function**
- **create index**
- **create key**
- **create login**
- **create login profile**
- **create plan***

- **create procedure**
- **create rule**
- **create schema***
- **create table**
- **create trigger**
- **create view**
- **drop default**
- **drop function**
- **drop login**
- **drop login profile**
- **drop index**
- **drop procedure**
- **drop rule**
- **drop table**
- **drop trigger**
- **drop view**
- **grant**
- **installjava*** – MSA 环境不支持复制 **installjava**。
- **remove java***
- **revoke**

受支持的系统过程有:

- **sp_add_qpgroup***
- **sp_addalias**
- **sp_addgroup**
- **sp_addmessage**
- **sp_addtype**
- **sp_adduser**
- **sp_bindefault**
- **sp_bindmsg**
- **sp_bindrule**
- **sp_cachestrategy**
- **sp_changegroup**
- **sp_chgattribute**
- **sp_commonkey**
- **sp_config_rep_agent**
- **sp_drop_all_qplans***
- **sp_drop_qpgroup***
- **sp_dropalias**

- **sp_dropgroup**
- **sp_dropkey**
- **sp_dropmessage**
- **sp_droptype**
- **sp_dropuser**
- **sp_encryption**
- **sp_export_qpgroup***
- **sp_foreignkey**
- **sp_hidetext**
- **sp_import_qpgroup***
- **sp_primarykey**
- **sp_procxmode**
- **sp_recompile**
- **sp_rename**
- **sp_rename_qpgroup***
- **sp_replication_path**
- **sp_setrepcol**
- **sp_setrepdefmode**
- **sp_setrepproc**
- **sp_setreplicate**
- **sp_setreptable**
- **sp_unbindefault**
- **sp_unbindmsg**
- **sp_unbindrule**

支持在主数据库中进行复制的 DDL 命令和系统过程集不同于支持在用户数据库中进行复制的 DDL 命令和系统过程集。

如果数据库是主数据库，则支持的 DDL 命令有：

- **alter role**
- **create role**
- **drop role**
- **grant role**
- **revoke role**

如果数据库是主数据库，则支持的系统过程为：

- **sp_addexternlogin**
- **sp_addlogin**
- **sp_addremotelogin**
- **sp_addserver**

- **sp_defaultdb**
- **sp_defaultlanguage**
- **sp_displaylevel**
- **sp_dropexternlogin**
- **sp_droplogin**
- **sp_dropremotelogin**
- **sp_dropserver**
- **sp_locklogin**
- **sp_maplogin**
- **sp_modifylogin**
- **sp_password**
- **sp_passwordpolicy** – 为 **allow password downgrade** 以外的所有选项复制。
- **sp_role**

sp_setrepcol

设置或显示 *text*、*unitext* 或 *image* 列的复制状态。

语法

```
sp_setrepcol table_name [, {column_name | null}
    [, {do_not_replicate | always_replicate |
replicate_if_changed}]]
             [, use_index]
```

参数

- **table_name** – 复制表的名称。在执行 **sp_setrepcol** 前，您必须使用 **sp_setreptable** 启用表的复制。
- **column_name** – 表中 *text*、*unitext* 或 *image* 列的名称。如果将列名指定为 **null**，则可以设置表中所有 *text*、*unitext* 或 *image* 列的复制状态。
- **do_not_replicate** – 禁止 Adaptive Server 记录 *text*、*unitext* 或 *image* 列的复制信息。如果以前将列标记为使用索引进行复制，则设置 **do_not_replicate** 可删除该索引。
- **always_replicate** – 只要行中的任何列发生更改，Adaptive Server 就会记录 *text*、*unitext* 或 *image* 列的复制信息。这种状态会因为复制没有更改的 *text*、*unitext* 或 *image* 列而增加开销；然而，它可以防止在非原子实现期间由于行迁移或行更改而产生的数据不一致问题。
- **replicate_if_changed** – 只有在 *text*、*unitext* 或 *image* 列数据发生更改时，Adaptive Server 才会记录 *text*、*unitext* 或 *image* 列的复制信息。这种状态可以减少开销，但它可能导致在非原子实现期间由于行迁移或行更改而产生的数据不一致问题。

- **use_index** – 将列标记为使用 *text*、*unitext*、*image* 或 *rawobjects* 列的索引进行复制。

示例

- **示例 1** – 显示 *au_pix* 表中所有 *text*、*unitext* 或 *image* 列的复制状态。必须使用 **sp_setreptable** 将 *au_pix* 标记为要进行复制。

```
sp_setrepcol au_pix
```

- **示例 2** – 显示 *au_pix* 表中的 *pic* 列的复制状态。pic 必须是 *text*、*unitext* 或 *image* 数据类型列。

```
sp_setrepcol au_pix, pic
```

- **示例 3** – 指定 *au_pix* 表中的 *pic*（*image* 数据类型）应处于 **replicate_if_changed** 状态。（在 *pubs2* 数据库中的这一特定表中，没有其它 *text*、*unitext* 或 *image* 列。）

```
sp_setrepcol au_pix, pic, replicate_if_changed
```

- **示例 4** – 指定 *au_pix* 表中所有 *text*、*unitext* 或 *image* 列应处于 **replicate_if_changed** 状态。

```
sp_setrepcol au_pix, null, replicate_if_changed
```

- **示例 5** – 将列 *t*（*text* 数据类型）标记为 **replicate_if_changed** 并使用索引进行复制：

```
sp_setrepcol t1, t, replicate_if_changed, use_index
```

- **示例 6** – 禁用压缩的 LOB 列复制：

```
sp_setrepcol table_name, lob_column_name, 'do_not_replicate'
```

用法

- 使用 **sp_setreptable** 启用表的复制后，使用 **sp_setrepcol** 指定 *text*、*unitext* 或 *image* 列的复制方式。
- 您还可以执行带有表名的 **sp_setrepcol** 以显示表中所有 *text*、*unitext* 或 *image* 列的复制状态，或执行带有表名和 *text*、*unitext* 或 *image* 列名的该命令以显示指定列的复制状态。
- 使用 **replicate_if_changed** 选项可以减少 *text*、*unitext* 或 *image* 列的复制开销。但需要考虑以下限制和注意事项：
  - 如果您为某列指定 **replicate_if_changed** 状态，则任何包含该列的复制定义必须也处于 **replicate_if_changed** 状态。
  - 如果您将任一列的复制状态设置为 **replicate_if_changed**，就不能将任何包含该列的复制定义的自动更正设置为“on”。
  - 如果使用非基本预订实现并且已将任一 *text*、*unitext* 或 *image* 列的复制状态设置为 **replicate_if_changed**，则 Replication Server 将在错误日志文件中显示一条

消息。如果在预订实现期间应用程序修改了主表，此消息将警告您数据可能不一致。

- 如果您的应用程序允许将行迁移到预订中，并且您已将任一 *text*、*unitext* 或 *image* 列的复制状态设置为 **replicate_if_changed**，则在行迁移到预订时，如果 *text* 或 *image* 数据丢失，Replication Server 将在错误日志中显示一条警告消息。

  如果在主表中执行更新操作时未更改状态为 **replicate_if_changed** 的 *text*、*unitext* 或 *image* 列，且更新操作导致行迁移到预订，则在复制表中插入的行将会丢失 *text*、*unitext* 或 *image* 数据。运行 **rs_subcmp** 程序可以使复制表和主表中的数据一致。

  当预订带有 **where** 子句时，可以进行行迁移。如果更新预订 **where** 子句中指定的列，可以使行对该预订有效或迁移到该预订中。

  当发生这种情况时，Replication Server 必须在复制数据库中执行 **insert**。**insert** 要求所有列的值，包括主数据库中未更改的 *text*、*unitext* 或 *image* 列。

- 如果将表标记为 **sp_reptostandby**，则不能使用 **sp_setrepcol** 更改 *text*、*unitext* 或 *image* 列的复制状态；*text*、*unitext* 和 *image* 列始终被作为 **replicate_if_changed** 处理。
- 如果热备份应用程序包括常规复制，并且您已用 **sp_reptostandby** 和 **sp_setreptable** 标记表，则可能会将 *text*、*unitext* 或 *image* 数据列作为 **always_replicate** 或 **replicate_if_changed** 处理。
  - 如果将 **sp_setreptable** 标记的 *text*、*unitext* 或 *image* 列指定为 **always_replicate**（缺省值），则会将所有 *text*、*unitext* 和 *image* 列作为 **always_replicate 处理。**
  - 如果 **sp_setrepcol** 将 *text*、*unitext* 或 *image* 列指定为 **do_not_replicate** 或 **replicate_if_changed**，则会将所有 *text*、*unitext* 或 *image* 列作为 **replicate_if_changed** 处理。
- 索引状态的优先顺序依次是：列、表、数据库。如果将表标记为使用 *text*、*unitext*、*image* 或 *rawobject* 列的索引，但您不想使用这些列中任意一个列的索引，则列状态将覆盖表状态。
- 不能使用 **drop index** 手动删除为 *text*、*unitext*、*image* 或 *rawobject* 复制创建的索引。只能使用支持的复制存储过程 **sp_reptostandby**、**sp_setreptable** 和 **sp_setrepcol** 来更改复制索引状态。

权限

sp_setrepcol 要求“sa”或“dbo”权限或者 **replication_role**。

另请参见

- sp_reptostandby （第 477 页）
- sp_setreplicate （第 490 页）
- sp_setreptable （第 493 页）

sp_setrepdbmode

在数据库级为一种或多种特定 DML 操作类型启用或禁用 SQL 语句复制。

语法

```
sp_setrepdbmode dbname [, “option [option [...]]” [, “on” | “off”]
   [ ‘threshold’, ‘value’ ]
```

```
option ::= { U | D | I | S }
```

参数

- **dbname** – 要启用 SQL 语句复制的数据库的名称。
- **option** – 以下 DML 操作的任意组合：
  - U – **update**
  - D – **delete**
  - I – **insert select**
  - S – **select into**

  如果将数据库复制模式设置为任意 **UDIS** 组合，RepAgent 将会发送各个日志记录以及 Replication Server 生成 SQL 语句所需的信息。
- **on** – 启用指定的 DML 操作的 SQL 复制。
- **off** – 在数据库级为所有类型的 DML 操作禁用 SQL 语句复制，而不管在 *option* 中指定了什么操作。
- **'threshold', 'value'** – 指定复制的 SQL 语句必须至少影响多少行才会激活 SQL 语句复制。将 *value* 重置为“0”，表示缺省阈值为 50 行。

示例

- **示例 1** – 复制 **delete** 和 **select into** 语句：

  ```
  sp_setrepdbmode pdb, 'DS', 'on'
  ```
- **示例 2** – 显示当前 SQL 复制设置：

  ```
  1> sp_setrepdbmode pdb1
  2> go
  ```

  ```
  The replication mode for database 'pdb1' is 'us'.
  (return status = 0)
  ```
- **示例 3** – 若要在数据库级禁用所有 SQL 语句的复制，请使用：

  ```
  sp_setrepdbmode pdb, 'D', 'off'
  ```
- **示例 4** – 将阈值设置为 100 行：

```
sp_setrepdbmode pubs2, 'threshold', '100'
go
```

- **示例 5** – 此示例显示在数据库级和表级为 *pubs2* 数据库和 *table1* 表设置不同的阈值：

  1. 在数据库级将阈值重置为缺省值 50 行：

     ```
     sp_setrepdbmode pubs2, 'threshold', '0'
     go
     ```

  2. 启用 *pubs2* 的 **update**、**delete**、**insert** 和 **select into** 操作的 SQL 语句复制：

     ```
     sp_setrepdbmode pubs2, 'udis', 'on'
     go
     ```

  3. 仅当在 *table1* 上执行 **update**、**delete**、**insert** 和 **select into** 操作并影响 1,000 以上的行时，才为 *pubs2* 中的 *table1* 触发 SQL 语句复制：

     ```
     sp_setrepdefmode table1, 'threshold', '1000'
     go
     ```

- **示例 6** – 此示例显示如何在数据库级为 *pubs2* 定义阈值，同时为 *table1* 和 *table2* 等表定义不同的操作：

  1. 在数据库级设置阈值，以便当数据操作语言 (DML) 语句影响 100 行以上时触发 SQL 语句复制：

     ```
     sp_setrepdbmode pubs2, 'threshold', '100'
     go
     ```

  2. 为两个特定表定义一组不同的操作，并在其中使用 SQL 语句复制来复制操作。**Update**、**delete** 和 **insert** 操作针对的是 *table1*，delete 操作针对的是 *table2*：

     ```
     sp_setrepdefmode table1, 'udi', 'on'
     go
     sp_setrepdefmode table2, 'd' 'on'
     go
     ```

  在此示例中，在对 *table2* 执行 **delete** 操作时，或在 *table1* 上执行任何 DML 时，如果达到您在数据库级指定的阈值 100 行，将会触发 SQL 语句复制。

用法

- 只有在将数据库标记为要进行复制时（通过将 **sp_reptostandby** 设置为 **ALL** 或 **L1**），您才能在数据库级别设置 SQL 语句复制。
- 缺省阈值为 50 行，这表示如果 DML 语句至少影响 51 行，Adaptive Server 将使用 SQL 语句复制。若要使用缺省阈值，请将 **threshold** 参数设置为 0。**threshold** 参数的范围是 0 到 10,000。
- 您可以在数据库级配置复制，同时在数据库级为 SQL 语句复制设置阈值。例如：

```
sp_reptostandby pubs2, 'none'
go
sp_setrepdbmode pubs2, 'threshold'
go
```

但您不能在数据库级配置复制并在数据库级定义操作，因为数据库级的 SQL 语句复制要求复制整个数据库，您不能只复制操作。例如，您不能执行：

```
sp_reptostandby pubs2, 'none'
go
sp_setrepdbmode pubs2, 'udis', 'on'
go
```

- 在会话级设置的阈值覆盖表级和数据库级的阈值，并且为任何表设置的阈值覆盖在数据库级设置的阈值。

另请参见

- set repmode （第 449 页）
- sp_setrepdefmode （第 488 页）
- set repthreshold （第 450 页）

sp_setrepdefmode

更改或显示标记为要进行复制的表的所有者状态，并为特定 DML 操作启用或禁用表级 SQL 语句复制。

语法

```
sp_setrepdefmode table_name [, 'owner_on' | 'owner_off' |
    'SQLDML_option [SQLDML_option [ ...]]' [, 'on' | 'off' |
'never' ] |
    'threshold', 'value']
```

```
SQLDML_option ::= { U | D | I }
```

参数

- **table_name** – 当前数据库中已用 **sp_setreptable** 标记为要复制的表的名称。
- **owner_on** – 更改表的所有者状态，这样当该表标记为要复制时，会考虑表名和所有者的名称。如果多个表的名称相同，但所有者不同，可以启用这些表的复制。
- **owner_off** – 更改表的所有者状态，这样当该表标记为要复制时，只考虑表名。
- **SQLDML_option** – 任何以下 DML 操作：
  - U – **update**
  - D – **delete**
  - I – **insert select**

  如果将表复制模式设置为任意 **UDI** 组合，RepAgent 将会发送其它信息，以便为指定的 DML 操作启用 SQL 语句复制。
- **on** – 启用指定的 DML 操作的 SQL 复制。

- **off** – 删除 SQL 语句的表级复制设置（无论是否在 *option* 中指定这些语句），并采用数据库级复制设置。
- **never** – 禁用 SQL 语句复制，而无论数据库设置是什么以及是否指定了 UDI 参数。
- **'threshold', 'value'** – 指定复制的 SQL 语句必须至少影响多少行才会激活 SQL 语句复制。

示例

- **示例 1** – 在 *t* 表上启用 **update**、**delete** 和 **insert select** 操作的 SQL 语句复制：

```
1> sp_setrepdefmode t, 'UDI', 'on'
2> go
```

- **示例 2** – 将阈值设置为 10。如果 DML 语句至少影响 11 行，Adaptive Server 将在 *t* 表上使用 SQL 复制：

```
sp_setrepdefmode t, 'threshold', '10'
```

- **示例 3** – 显示 *rs_ticket_history* 表的 SQL 复制设置和所有者状态：

```
1> sp_setrepdefmode rs_ticket_history,'udi'
2> go
```

```
The replication status for 'rs_ticket_history' is
currently  owner_off, 'udi'.
The replication threshold for table 'rs_ticket_history'
is '0'.
(return status = 0)
```

- **示例 4** – 将阈值设置为缺省值：

```
sp_setrepdbmode t, 'threshold', '0'
```

用法

- **sp_setrepdefmode** 用于启用了 RepAgent 的 Adaptive Server 数据库。
- 如果执行 **sp_setrepdefmode** 时只带有表名，它将显示该表的 SQL 复制设置和所有者状态。
- 使用 **sp_setrepdefmode** 更改表的模式。您不能用 **sp_setreptable** 更改表的所有者模式。
- 如果提供 **owner_off** 选项并且表的当前模式为“owner on”，**sp_setrepdefmode** 将检查表名在所有处于 **owner off** 模式的复制表中是否唯一。如果名称唯一，则 **sp_setrepdefmode** 会将表模式更改为 **owner off**。如果名称不唯一，此过程将失败。
- 缺省阈值为 50 行，这表示如果 DML 语句至少影响 51 行，Adaptive Server 将使用 SQL 语句复制。若要使用缺省阈值，请将 **threshold** 参数设置为 0。**threshold** 参数的范围是 0 到 10,000。

权限

sp_setrepdefmode 要求“sa”或“dbo”权限或者 **replication_role**。

另请参见

- set repmode （第 449 页）
- sp_setreptable （第 493 页）
- sp_setrepdbmode （第 486 页）
- set repthreshold （第 450 页）

sp_setreplicate

启用或禁用 Adaptive Server 表或存储过程的复制。它还显示表或存储过程的当前复制状态。

注意： 此系统过程仍受支持，但它的功能已被并入系统过程 **sp_setreptable** 和 **sp_setrepproc** 中。**sp_setreplicate** 将 *text*、*unitext* 或 *image* 数据类型列的复制状态设置为 **do_not_replicate**。要复制 *text*、*unitext* 或 *image* 列，请使用 **sp_setreptable** 系统过程，而不是 **sp_setreplicate**。若要指定要复制的各个 *text*、*unitext* 或 *image* 列，请在使用 **sp_setreplicate** 或 **sp_setreptable** 之后使用 **sp_setrepcol**。

语法

```
sp_setreplicate [object_name [, {'true' | 'false'}]]
```

参数

- **object_name** – 是当前数据库中表或存储过程的名称。
- **true** – 启用该表或存储过程的复制。
- **false** – 禁用该表或存储过程的复制。

示例

- **示例 1** – 显示当前数据库中所有表和存储过程的复制状态。

  ```
  sp_setreplicate
  ```

- **示例 2** – 显示 *publishers* 表的复制状态。

  ```
  sp_setreplicate publishers
  ```

- **示例 3** – 启用 *publishers* 表的复制。

  ```
  sp_setreplicate publishers, 'true'
  ```

用法

- 使用函数复制定义时，可以使用 **sp_setrepproc** 启用或禁用存储过程的复制。使用表复制定义时，可以使用 **sp_setrepproc** 或 **sp_setreplicate** 启用或禁用存储过程的复制。

- 使用不带参数的 **sp_setreplicate** 可显示数据库中复制表或存储过程的列表。
- 使用不带 **true** 或 **false** 的 **sp_setreplicate** *object_name* 可显示表或存储过程的当前复制状态。
- 如果您使用 **sp_reptostandby** 将一个表标记为要隐式复制到备用数据库，则会将由 **sp_setreplicate** 或 **sp_setrepcol** 设置为 **do_not_replicate** 的 *text*、*unitext* 或 *image* 列作为 **replicate_if_changed** 处理。而设置为 **always_replicate** 或 **replicate_if_changed** 的列将被作为已标记的列处理。
- 因为 Adaptive Server Enterprise 会启动一个事务来执行复制存储过程，所以在设计过程时牢记以下几点非常重要：
  - 如果复制存储过程包含 DDL 命令（例如 **create table**），则 Adaptive Server Enterprise 会生成错误，除非数据库上启用了数据库选项“DDL-in-Tran”。
  - 如果复制存储过程包含事务和回退事务的回退命令，那么回退命令将回退整个过程的执行。
  - 由于存在外部事务，Adaptive Server Enterprise 将保持所有锁定，直到过程执行完毕。

另请参见

- sp_setrepcol （第 483 页）
- sp_setrepproc （第 491 页）
- sp_setreptable （第 493 页）

sp_setrepproc

启用或禁用存储过程的复制，或显示存储过程的当前复制状态。

语法

```
sp_setrepproc [proc_name [,'false' | 'table' |
      'function' [,'log_current' | 'log_sproc']]]
```

参数

- **proc_name** – 当前数据库中存储过程的名称。
- **false** – 禁用存储过程的复制。
- **table** – 启用与表复制定义相关的存储过程的复制。此选项等效于对过程执行 **sp_setreplicate**。
- **函数** – 启用与函数复制定义相关的存储过程的复制。
- **log_current** – 记录您在当前数据库（而不是复制的存储过程所在的数据库）中复制的存储过程的执行情况。
- **log_sproc** – 记录您在存储过程所在的数据库（而不是当前数据库）中复制的存储过程的执行情况。缺省值为 **log_sproc**。

示例

- **示例 1** – 显示当前数据库中所有存储过程的复制状态。对于每一个过程，将指明是完全启用复制、使用函数复制定义启用还是使用表复制定义启用。

```
sp_setrepproc
```

- **示例 2** – 显示 **upd_pubs** 存储过程的复制状态。指明存储过程是完全启用复制、使用函数复制定义启用还是使用表复制定义启用。

```
sp_setrepproc upd_pubs
```

- **示例 3** – 启用 **upd_pubs** 存储过程的复制以用于函数复制定义。对 **upd_pubs** 的执行将记录在 **upd_pubs** 所在的数据库中。

```
sp_setrepproc upd_pubs, 'function'
```

- **示例 4** – 启用 **upd_pubs** 存储过程的复制以用于表复制定义。对 **upd_pubs** 的执行将记录在 **upd_pubs** 所在的数据库中。

```
sp_setrepproc upd_pubs, 'table'
```

- **示例 5** – 启用 upd_pubs 存储过程的复制以用于函数复制定义。对 **upd_pubs** 的执行将记录在当前数据库中。

```
sp_setrepproc upd_pubs, 'function', 'log_current'
```

- **示例 6** – 启用 **upd_pubs** 存储过程的复制以用于函数复制定义。对 **upd_pubs** 的执行将记录在 **upd_pubs** 所在的数据库中。

```
sp_setrepproc upd_pubs, 'function', 'log_sproc'
```

用法

- 使用不带参数的 **sp_setrepproc** 可以显示数据库中的所有复制存储过程。
- 使用不带其它参数的 **sp_setrepproc** *proc_name* 可以显示存储过程的当前复制状态。
- 如果您使用的是 Adaptive Server 11.5 版或更高版本，并且用 **sp_setrepproc** 启用过程复制，则在用户存储过程内部执行的受支持的 DDL 命令和存储过程将被复制到备用数据库。

  如果没有用 **sp_setrepproc** 启用过程复制，则在用户存储过程内部执行的受支持的 DDL 命令和存储过程将不复制到备用数据库。
- 因为 Adaptive Server 会启动一个事务来执行复制存储过程，所以在设计过程时请记住以下几点：
  - 如果复制存储过程包含 DDL 命令（例如 **create table**），则 Adaptive Server Enterprise 会生成错误，除非数据库上启用了数据库选项“DDL-in-Tran”。
  - 如果复制存储过程包含事务和回退事务的回退命令，那么回退命令将回退整个过程的执行。
  - 由于存在外部事务，Adaptive Server 将保持所有锁定，直到过程执行完毕。

另请参见

- sp_reptostandby （第 477 页）
- sp_setreplicate （第 490 页）
- sp_setreptable （第 493 页）

sp_setreptable

启用或禁用 Adaptive Server 表的复制或显示表的当前复制状态。

语法

```
sp_setreptable [table_name [, {'true' | 'false' | 'never'}
     [, {owner_on | owner_off | null}] [, use_index]]]
```

参数

- **table_name** – 标记为要复制的表的名称。
- **true** – 将表显式标记为要进行复制，而无论是否将数据库标记为要进行复制。
- **false** – 在以前已启用复制的表上禁用复制状态。
- **never** – 在表上禁用复制，而无论数据库复制设置是什么。
- **owner_on** – 设置表的模式，这样当该表标记为要复制时，将考虑表名和所有者的名称。如果多个表的名称相同，但所有者不同，可以启用这些表的复制。此选项用于 Adaptive Server 11.5 版和更高版本的数据库。
- **owner_off** – 设置表的模式，这样当该表标记为要复制时，将只考虑表名。这是缺省值。它确保每一个标记为要复制的表的名称都是唯一的。此选项用于 Adaptive Server 11.5 版和更高版本的数据库。
- **null** – 将其传递给 owner 参数时设置缺省值 **owner_off**。
- **use_index** – 将表标记为使用 *text*、*unitext*、*image* 或 *rawobjects* 列的索引进行复制。

示例

- **示例 1** – 显示当前数据库中您使用 **sp_setreptable** 标记为要进行复制的所有表的复制状态：

  ```
  sp_setreptable
  ```

- **示例 2** – 显示 *publishers* 表的复制状态：

  ```
  sp_setreptable publishers
  ```

- **示例 3** – 启用 *publishers* 表的复制：

  ```
  sp_setreptable publishers, 'true'
  ```

- **示例 4** – 允许复制多个名称都为 *publishers* 但所有者不同的表：

```
sp_setreptable publishers, 'true', owner_on
```

- **示例 5** – 复制名为 *publishers*、属于所有者 *dbo* 且存储在数据库 *pubs2* 中的表：

```
sp_setreptable 'pubs2.dbo.publishers', 'true', owner_on
```

- **示例 6** – 将要复制的表标记为使用 *text*、*unitext*、*image* 和 *rawobject* 列的索引，并将所有者状态设置为“off”：

```
sp_setreptable t1, true, null, use_index
```

- **示例 7** – 如果最初将 *t1* 标记为使用索引进行复制，则删除表 *t1* 的复制状态并删除复制索引：

```
sp_setreptable t1, 'false'
```

- **示例 8** – 若要在 *pdb* 数据库中的 *tnever* 表上禁用复制，请使用：

```
sp_reptostandby pdb, 'ALL'
go
sp_setreptable tnever, 'never'
go
```

用法

- 使用不带参数的 **sp_setreptable** 可以显示数据库中复制表的列表。
- 使用不带 **true** 或 **false** 的 **sp_setreptable** *table_name* 可以显示表的当前复制状态。
- 如果包括 **owner_on** 选项，表名相同但所有者不同的多个表可以被复制到复制数据库和热备份数据库中。确保表上的复制定义也包括所有者信息，否则复制可能失败。
- 如果已使用 **sp_setreptable** 将某个表标记为要复制，则可以使用 **sp_setrepdefmode** 系统过程更改所有者模式。
- 复制索引状态的优先顺序依次是：列、表、数据库。例如，在标记为要使用索引进行复制的数据库中，表的状态会覆盖索引的状态：
- 如果将包含一个或多个 *text*、*unitext*、*image* 或 *rawobject* 列的大表标记为要进行复制，则会在一个事务中执行内部进程，并且执行该进程可能会花费很长时间。要加快进程的速度，请使用 **use_index** 选项为每个 *text*、*unitext*、*image* 或 *rawobject* 列创建全局性的非聚簇索引。
- 如果使用 **use_index**，在创建全局非聚簇索引的同时将保持共享表锁。
- 不能使用 **drop index** 手动删除为 *text*、*unitext*、*image* 或 *rawobject* 复制创建的索引。只能使用支持的复制存储过程 **sp_reptostandby**、**sp_setreptable** 和 **sp_setrepcol** 来更改复制索引状态。

权限

sp_setreptable 要求“sa”或“dbo”权限或者 **replication_role**。

另请参见

- sp_reptostandby （第 477 页）
- sp_setrepcol （第 483 页）

- sp_setrepdefmode （第 488 页）
- sp_setreplicate （第 490 页）
- sp_setrepproc （第 491 页）

sp_start_rep_agent

为指定数据库启动 RepAgent 线程。

语法

```
sp_start_rep_agent dbname[, {'recovery' | 'recovery_foreground' |
'resync' | 'resync purge'
                                                                  |
'resync init'} [, 'connect_dataserver',
              'connect_database'[, 'repserver_name',
repserver_username',
             'repserver_password']]]
```

参数

- **dbname** – 要为其启动 RepAgent 的数据库的名称。
- **recovery** – 以恢复模式启动 RepAgent，该模式用于启动恢复操作。恢复模式用于在队列丢失后重建队列。

  您也可以在恢复模式下指定 Replication Server 名称、用户名和口令。指定这些参数可覆盖 *sysattributes* 设置。
- **recovery_foreground** – **recovery_foreground** 具有与 **recovery** 相同的功能。但是，它在屏幕上显示恢复进度信息，而不是在 Adaptive Server 错误日志中显示。如果恢复进度信息显示结束并显示命令提示符，则说明恢复已完成。
- **resync** –

  当截断点没有更改时，可发送不带任何选项的 resync database 标记，例外情况是 RepAgent 应从它处理的最后一个点继续处理事务日志。
- **resync purge** –

  发送带有 purge 选项的 resync database 标记，以指示 Replication Server 在接收任何新入站事务之前，清除入站队列中所有打开的事务并重置重复检测。
- **resync init** –

  发送带有 init 选项的 resync database 标记，以指示 Replication Server 清除入站队列中所有打开的事务、重置重复检测和挂起出站 DSI。
- **connect_dataserver** – 用于恢复脱机日志的数据服务器的名称。
- **connect_database** – 用于恢复脱机日志的数据库的名称。
- **repserver_name** – RepAgent 所连接的 Replication Server 的名称。
- **repserver_user_name** – RepAgent 用于与 Replication Server 连接的用户名。

- **repserver_password** – RepAgent 用于与 Replication Server 连接的口令。

示例

- **示例 1** – 为 *pubs2* 数据库启动集成的 RepAgent。RepAgent 与 **sp_config_rep_agent** 中指定的 Replication Server 连接。它将开始扫描事务日志并将已设置格式的 LTL 命令发送到 Replication Server。

```
sp_start_rep_agent pubs2
```

- **示例 2** – 在恢复模式下为连接到 *svr2* 数据服务器的 *pdb2* 数据库启动 RepAgent。

```
sp_start_rep_agent pubs2 for_recovery, svr2, pdb2
```

- **示例 3** – 将 RepAgent 配置为将数据库 *db2* 的恢复显示到客户端：

```
sp_start_rep_agent db2, recovery_foreground, ds, db1
```

```
RepAgent(5). Starting recovery, processing log records
   between (1018, 0) and (2355, 2).
RepAgent(5). Processed 1000 log records.
RepAgent(5). Processed 2000 log records.
RepAgent(5). Processed 3000 log records.
RepAgent(5). Processed 4000 log records.
RepAgent(5). Processed 5000 log records.
RepAgent(5). Processed 6000 log records.
RepAgent(5). Processed 7000 log records.
RepAgent(5). Processed 8000 log records.
RepAgent(5). Processed 9000 log records.
RepAgent(5). Processed 10000 log records.
RepAgent(5). Processed 11000 log records.
RepAgent(5). Processed 12000 log records.
RepAgent(5). Processed 13000 log records.
RepAgent(5). Processed 14000 log records.
RepAgent(5). Processed 15000 log records.
RepAgent(5). Processed 16000 log records.
RepAgent(5). Processed 17000 log records.
RepAgent(5). Processed 18000 log records.
RepAgent(5). Processed 19000 log records.
RepAgent(5). Processed 20000 log records.
RepAgent(5). Processed 20084 log records, recovery
   complete.
Replication Agent thread is started for database 'db2'.
(return status = 0)
```

用法

- **sp_start_rep_agent** 用于启用了 RepAgent 的数据库。
- 使用 **sp_config_rep_agent** 启用 RepAgent 后，请使用 **sp_start_rep_agent** 命令启动 RepAgent。使用 **sp_start_rep_agent** 启动 RepAgent 之后，在服务器启动期间，它将在数据服务器恢复后自动启动。
- 使用 **sp_stop_rep_agent** 关闭 RepAgent 后，将禁用自动启动。可使用 **sp_start_rep_agent** 将其重新启用。

- 对于脱机恢复，存档事务日志可能被转储到临时恢复数据库中。然后，您就可以将临时恢复数据库的事务日志中的记录传送到复制数据库。可以使用 **recovery** 或 **recovery_foreground** 执行 **sp_start_rep_agent**（使用临时数据服务器和数据库名称）来扫描临时事务日志。

  在恢复过程中，RepAgent 扫描完事务日志后就会关闭。在下一个事务转储装载完毕后，执行带有先前指定的选项的 **sp_start_rep_agent** 即可重新启动 RepAgent。

权限

sp_start_rep_agent 要求“sa”或“dbo”权限或者 **replication_role**。

另请参见

- sp_help_rep_agent （第 463 页）
- sp_stop_rep_agent （第 497 页）

sp_stop_rep_agent

为指定数据库关闭 RepAgent 线程。

语法

```
sp_stop_rep_agent dbname[, 'nowait']
```

参数

- **dbname** – 要为其关闭 RepAgent 的数据库的名称。
- **nowait** – 立即关闭 RepAgent，而无需等待正在执行的操作完成。

  缺省设置是等到当前批处理结束后再关闭 RepAgent。

示例

- **示例 1** – 为 *pubs2* 数据库关闭集成的 RepAgent。缺省关闭选项允许 RepAgent 完成处理当前的批处理。

  ```
  sp_stop_rep_agent pubs2
  ```

用法

- **sp_stop_rep_agent** 用于启用了 RepAgent 的数据库。
- 使用 **sp_stop_rep_agent** 关闭 RepAgent 后，在服务器启动期间该数据库联机后，它不会自动启动。要重新启用自动启动，请执行 **sp_start_rep_agent** 过程。
- **sp_stop_rep_agent** 是异步的，执行它可能要花些时间。使用 **sp_who** 可检查 RepAgent 的状态。

权限

sp_start_rep_agent 要求“sa”或“dbo”权限或者 **replication_role**。

另请参见

- sp_config_rep_agent （第 456 页）
- sp_help_rep_agent （第 463 页）
- sp_start_rep_agent （第 495 页）

RSSD 存储过程

列出用于 Replication Server 的 RSSD 存储过程。

rs_capacity

帮助您估计稳定队列的大小要求。与 **rs_fillcaptable** 存储过程一起使用。

语法

```
rs_capacity TranDuration, FailDuration, SaveInterval, MatRows
```

参数

- **TranDuration** – 最长事务的持续时间（以秒计）。缺省值为最大 5 秒。
- **FailDuration** – 故障期间队列必须保留信息的时间长度（以分钟计）。缺省值为 60 分钟。
- **SaveInterval** – 在确认消息已接收之后应保留这些消息的时间长度（以分钟计）。缺省值为 1 分钟。
- **MatRows** – 预订中要实现的行数。缺省值为 1000 行。

示例

- **示例 1** – 在 **rs_fillcaptable** 存储过程的示例中，使用带有以下参数的 **rs_capacity**。

```
rs_capacity
 60,    /* TranDuration maximum 60 seconds */
 360,   /* FailDuration 6 hours */
 10,    /* SaveInterval 10 minutes */
 3500   /* Materialize 3500 rows */
```

rs_capacity 返回每一队列所需的队列大小的估计值。它还根据复制定义和要实现的行数给出所需的预订实现队列大小的估计值。

用法

- **rs_capacity** 使用 *rs_captable* 表（使用 **rs_fillcaptable** 存储过程创建）中的数据来计算稳定队列大小要求的估计值。请在使用 **rs_fillcaptable** 说明复制定义之后执行 **rs_capacity**。

另请参见

- rs_fillcaptable （第 506 页）

rs_delexception

删除例外日志中的事务。

语法

```
rs_delexception [transaction_id]
```

参数

- **transaction_id** – 要删除的事务的号码。

示例

- **示例 1** – 从例外日志中删除号码为 1234 的事务。

  ```
  rs_delexception 1234
  ```

用法

- 如果不指定任何参数，**rs_delexception** 将在例外日志中显示事务摘要。
- 如果您提供了有效的 *transaction_id*，**rs_delexception** 会删除事务。使用不带参数的 **rs_helpexception** 或 **rs_delexception** 即可查找事务的 *transaction_id*。

另请参见

- rs_helpexception （第 519 页）
- rs_delexception_date （第 500 页）
- rs_delexception_id （第 501 页）
- rs_delexception_range （第 502 页）

rs_delexception_date

删除 `rs_exceptscmd`、`rs_exceptshdr` 和 `rs_systext` 系统表中由例外日志中的事务日期标识的一组事务。

语法

```
rs_delexception_date transaction_date_start [,transaction_date_end]
```

参数

- **transaction_date_start** – 要删除的范围中最早事务的起始日期。用双引号将日期括起来。

- **transaction_date_end** – 要删除的范围中最晚事务的起始日期。指定日期范围中最晚事务起始日期为可选日期。用双引号将日期括起来。

示例

- **示例 1** – 从例外日志中删除起始日期为 2010 年 10 月 1 日的事务。

```
rs_delexception_date "10/01/2010"
```

- **示例 2** – 从例外日志中删除起始日期在 2010 年 10 月 1 日和 2010 年 10 月 31 日之间（含）的所有事务。

```
rs_delexception_date "10/01/2010", "10/31/2010"
```

用法

- 可以采用承载 RSSD 或 SQL Anywhere 数据库（即 ERSSD）的 Adaptive Server 支持的不同格式，为 *transaction_date_end* 和 *transaction_date_end* 输入日期。有关可接受的日期和时间格式的信息，请参见：
  - 《Adaptive Server Enterprise 参考手册：构件块》的“系统数据类型和用户定义的数据类型”的“日期和时间数据类型”中的“输入日期和时间数据”
  - 《SQL Anywhere Server - SQL 参考》的“SQL 数据类型”的“日期和时间数据类型”中的“向数据库发送日期和时间”。
- **rs_delexception_date** 从例外表中删除 *transaction_date_start* 和 *transaction_date_end* 之间范围的事务，不包括 *transaction_date_start* 和 *transaction_date_end*。
- 如果您没有指定任何参数，**rs_delexception_date** 会显示一条错误消息。在执行不带参数的 **rs_helpexception** 或 **rs_delexception** 以获取例外日志中有效的事务和起始日期的最新列表时，可查看 `"org date"` 列。
- 如果您仅为 *transaction_date_start* 指定了有效日期，而没有在 *transaction_date_end* 中指定第二个有效日期，则 **rs_delexception_date** 只删除您在 *transaction_date_start* 中指定的事务。
- 如果您输入的命令没有导致删除任何事务，**rs_delexception_date** 将显示错误消息。

另请参见

- rs_delexception （第 500 页）

rs_delexception_id

删除 `rs_exceptscmd`、`rs_exceptshdr` 和 `rs_systext` 系统表中由例外日志中的事务 ID 标识的一组事务。

语法

```
rs_delexception_id transaction_id_start [,transaction_id_end]
```

参数

- **transaction_id_start** – 要删除的范围中第一个事务的 ID 号。
- **transaction_id_end** – 要删除的范围中最后一个事务的 ID 号。指定范围中的最后一个事务为可选事务。

示例

- **示例 1** – 从例外日志中删除 ID 号为 1234 的事务。您还可以使用 **rs_delexception** 删除单个事务。

```
rs_delexception 1234
```

- **示例 2** – 从例外日志中删除 ID 号在 1234 和 9800 之间（含）的所有事务。

```
rs_delexception 1234, 9800
```

用法

- **rs_delexception_id** 从例外表中删除 *transaction_id_start* 和 *transaction_id_end* 之间范围的事务，不包括 *transaction_id_start* 和 *transaction_id_end*。
- 如果您没有指定任何参数，**rs_delexception_id** 会显示一条错误消息。使用不带参数的 **rs_helpexception** 或 **rs_delexception** 可以获取例外日志中有效事务的最新列表。
- 如果您在 *transaction_id_start* 中为事务 ID 指定了单个有效值，而没有在 *transaction_id_end* 中指定第二个事务 ID 号，则 **rs_delexception_id** 只删除您在 *transaction_id_start* 中指定的事务。
- 如果您输入 0（零）作为事务 ID 号，并且没有输入第二个事务 ID 号，**rs_delexception_id** 将删除例外日志中的所有事务。
- 如果您输入浮点数（例如 123.456），并且您要使用：
  - **ERSSD** – **rs_delexception_id** 将只处理整数 (123) 而忽略小数点后面的数字
  - **RSSD** – **rs_delexception_id** 将返回一条错误消息，您可以重新输入命令
- 如果您输入的命令没有导致删除任何事务，**rs_delexception_id** 将显示错误消息。

另请参见

- rs_delexception （第 500 页）

rs_delexception_range

删除 `rs_exceptscmd`、`rs_exceptshdr` 和 `rs_systext` 系统表中由例外日志中的源节点或用户或目标节点标识的一组事务。

语法

```
rs_delexception_range
{{"origin"|"org"}, "origin_data_server.origin_database" |
```

```
, {"destination"|"dest"},
"destination_data_server.destination_database" |
, "user", "origin_user"}
```

参数

- **"origin"/"org", "*origin_data_server.origin_database*"** – 输入 **"origin"** 或简写形式 **"org"**，并指定发起您要从例外日志中删除的事务的数据服务器和数据库。用双引号将这些参数括起来，并使用逗号分隔各个参数。
- **"destination"/"dest", "*destination_data_server.destination_database*"** – 输入 **destination** 或简写形式 **"dest"**，并指定接收您要从例外日志中删除的事务的数据服务器和数据库。用双引号将这些参数括起来，并使用逗号分隔各个参数。
- **"user", "*origin_user*"** – 输入 **"user"**，并指定发起您要从例外日志中删除的事务的用户。用双引号将这些参数括起来，并使用逗号分隔各个参数。

示例

- **示例 1** – 从例外日志中删除源自 SYDNEY_DS 数据服务器的 `south_db` 数据库的事务。

  ```
  rs_delexception_range "org", "SYDNEY_DS.south_db"
  ```

- **示例 2** – 从例外日志中删除 TOKYO_DS 数据服务器的 `east_db` 数据库接收的事务。

  ```
  rs_delexception_range "destination", "TOKYO_DS.east_db"
  ```

- **示例 3** – 从例外日志中删除源自 rsuser1 用户的事务。

  ```
  rs_delexception_range "user", "rsuser1"
  ```

用法

- 您每次只能输入一个参数和相应值。例如，您不能在输入 "**org**", "*origin_dataserver.origin_database*" 后接着输入 "**user**", "*origin_user*"。
- 您必须输入参数并指定值。如果您没有指定任何参数，**rs_delexception_range** 会显示一条错误消息。在执行不带参数的 **rs_helpexception** 或 **rs_delexception** 以获取例外日志中有效事务的各个列的值的最新列表时，请查看 `OriginSite`、`Dest. Site` 和 `Dest. User` 列。
- 如果您仅为 **rs_delexception_range** 输入了 **"origin"**、**"destination**" 或 **"user"**，而没有指定相应值，则 **rs_delexception_range** 会显示错误消息。
- 如果您输入的命令没有导致删除任何事务，**rs_delexception_range** 将显示错误消息。

另请参见

- rs_delexception （第 500 页）

rs_dump_stats

提取在 RSSD 中通过 **admin stats** 收集的 Replication Server 统计信息（采用逗号分隔格式）。

语法

```
rs_dump_stats [ ‘comment’ ]
```

参数

- **comment** – 是所显示的统计信息的可选说明。它显示在输出文件的第一行中。

示例

- **示例 1** – 提取带有注释“Stats from 01/31/2006”的 Replication Server 统计信息。

  ```
  rs_dump_stats ‘Stats from 01/31/2006’
  ```

  计数器数据列依次为:

  - 观测周期的时间戳
  - 计数器在观测周期内进行观测的次数
  - 观测值总和
  - 最后观测值
  - 最大观测值

  根据计数器类别（有关计数器类别的说明，请参见《Replication Server 管理指南第二卷》的“性能调优”中的“使用计数器监控性能”）的不同，观测次数与观测值总和以及最后观测值与最大观测值之间可能具有很高的相关性。例如，观测器计数器只对事件的观测次数（如从队列中读取一条消息的次数）进行计数。对于观测器计数器，观测次数和观测值总和是相同的。同样，最后观测值和最大观测值都是 1（除非在观测周期内没有读取任何消息，此时两个值都为 0）。

  注意: 输出右侧包含一些注释，用于对此示例进行解释。这些注释不是 **rs_dump_stats** 输出的一部分。

  ```
  Comment: Stats from 01/31/2006    == Provided label
  Oct 17 2005  3:13:47:716PM    == End of the first observation
  period
  Oct 17 2005  3:14:24:730PM    == End of the last observation period
  2    == Number of observation periods
  0    == Number of minutes in each obs period.
  0 if less than one.(Calculated as the number of minutes between
  the first
  and last obs period, divided by the number of observations.)
  16384                         == Number of bytes in an SQM Block to
                                   aid calculations
  ```

```
64                              == Number of blocks in an SQM Segment
                                   to aid calculations
CM                              == Module Name. See rs_help_counter
                                   for a complete list.
13                             == Instance ID. See admin stats for an
                                   explanation.
-1                             == Inst Val/Mod Type. Further instance
                                   qualification when needed.
dCM                             == Instance description.
CM: Outbound database connection
requests                        == Counter description.
CMOBDBReq                       == Counter display name.
13003          , , 13,  -1     == Counter ID and instance qualifying
                                   information.
Oct 17 2005  3:13:47:716PM,  52,
52,  1,  1                      == Counter data. One row output for
                               each observation period. See below for
                                   explanation.
Oct 17 2005  3:14:24:730PM,  42,
42,  1,  1
ENDOFDATA                       == End of output for the previous
                                   counter
CM: Outbound non-database
connection requests             == Start of output for the next
counter
CMOBNonDBReq
13004           , , 13,  -1
Oct 17 2005  3:13:47:716PM,  2,  2,  1,  1
Oct 17 2005  3:14:24:730PM,  2,  2,  1,  1
ENDOFDATA
.
.
.
CM: Time spent closing an ob fadeout conn
CMOBConnFadeOutClose
13019           , , 13,  -1
Oct 17 2005  3:13:47:716PM,  0,  0,  0,  0
Oct 17 2005  3:14:24:730PM,  2,  6,  2,  4
ENDOFDATA
DIST                            == Start of output for the next
                                   module/instance
102
-1
DIST, 102 pds03.tpcc
DIST: Commands read from inbound queue
CmdsRead
30000           , , 102,  -1
Oct 17 2005  3:13:47:716PM,  1,  1,  1,  1
Oct 17 2005  3:14:24:730PM,  1,  1,  1,  1
ENDOFDATA
.
.
.
DSIEXEC: Number of 'message' results
DSIEResMsg
57127           , , 103,  7
```

```
Oct 17 2005  3:13:47:716PM,  1,  1,  1,  1
Oct 17 2005  3:14:24:730PM,  1,  1,  1,  1
ENDOFDATA
(return status = 0)            == End of output
```

用法

- 您可以使用文本文件来捕获 **rs_dump_stats** 输出，并随后使用电子表格或其它分析工具对其进行分析。
- 如果包含 **rs_dump_stats** 输出的文本文件太大而无法装载到分析工具中，可以将该文件拆分为多个文件。
  - 每个新文件必须包含原始文件的前 7 行和最后一行。
  - 在每个新文件的前 7 行和最后一行之间，插入与给定模块实例关联的所有行。

  取决于分析工具，通常不必在相同文件中包含某个模块的所有实例。
- **rs_dump_stats** 不会删除或更改 RSSD 中保存的统计信息。
- **rs_dump_stats** 会列出没有观测值的计数器，但不会为这类计数器显示计数器数据行。**rs_dump_stats** 会为采样周期内至少有一个观测值的所有计数器显示计数器数据行。

另请参见

- rs_helpcounter （第 512 页）
- admin stats （第 73 页）

rs_fillcaptable

将现有复制定义的估计事务发生率记录在 *rs_captable* 表中。

语法

```
rs_fillcaptable RepDefName, InChRateI, InChRateD, InChRateU,
OutChRateI, OutChRateD, OutChRateU, InTranRate, OutTranRate, DelFlag
```

参数

- **RepDefName** – 复制定义的名称。
- **InChRateI** – 每秒插入数，包括未复制的插入。缺省值为每秒插入 15 次。
- **InChRateD** – 每秒删除数，包括未复制的删除。缺省值为每秒删除 15 次。
- **InChRateU** – 每秒更新数，包括未复制的更新。缺省值为每秒更新 15 次。
- **OutChRateI** – 每秒插入数，不包括未复制的插入。缺省值为每秒插入 15 次。
- **OutChRateD** – 每秒删除数，不包括未复制的删除。缺省值为每秒删除 15 次。
- **OutChRateU** – 每秒更新数，不包括未复制的更新。缺省值为每秒更新 15 次。
- **InTranRate** – 数据库中每秒发生的事务数。缺省值为每秒发生 5 个事务。

- **OutTranRate** – 数据库中每秒发生的复制事务数。缺省值为每秒发生 5 个事务。
- **DelFlag** – 设置为“n”或“N”可更新复制定义的行。设置为“y”或“Y”可从 *rs_captable* 中删除复制定义的行。将 *DelFlag* 设置为“Y”，并将 *RepDefName* 设置为“ALL”，可清除整个 *rs_captable* 表。

示例

- **示例 1** – 在本示例中，主数据库中的整体事务发生率为每秒 10 个事务。在这 10 个事务中有 8 个是复制的。因此，数据库的 *InTranRate* 是 10 而 *OutTranRate* 是 8。

  有两个复制事务：T1 和 T2。T1 每秒执行 5 次，对 *table1* 执行 2 次更新，对 *table2* 执行 1 次更新。T2 每秒执行 3 次，对 *table1* 执行 2 次插入，对 *table2* 执行 1 次插入。

  复制数据库中有两个预订，每个预订接收一半复制数据。事务在这两个预订中平均分配。因此，出站估计值是入站估计值的 50%。

  下表总结了上面示例情况中的信息：

		table1			table2		
		插入	更新	删除	插入	更新	删除
入站	T1（5 次/秒）		10		5		
	T2（3 次/秒）	6			3		
	总计	6	10		8		
出站	50% 已复制	3	5		4		

  要获取本示例的稳定队列大小要求的估计值，首先要清除 *rs_captable* 表。然后执行带有上述参数的 **rs_fillcaptable**。完成之后，请使用 *rs_captable* 表中的新内容执行 **rs_capacity** 存储过程。

- **示例 2** – 本示例清除 *rs_captable* 表。

```
rs_fillcaptable @RepDefName = ’ ALL’, @DelFlag = ’ Y’
```

- **示例 3** – 本示例用第一个复制定义的适当值填充 *rs_captable* 表。

```
rs_fillcaptable
 repdef1, /* replication definition for table1 */
 6,       /* InChRateI */
 0,       /* InChRateD */
 10,      /* InChRateU */
 3,       /* OutChRateI */
 0,       /* OutChRateD */
 5,       /* OutChRateU */
 10,      /* InTranRate */
 8,       /* OutTranRate */
 n        /* DelFlag */
```

- **示例 4** – 本示例用第二个复制定义的适当值填充 *rs_captable* 表。

```
rs_fillcaptable
 repdef2, /* replication definition for table2 */
 8,       /* InChRateI */
 0,       /* InChRateD */
 0,       /* InChRateU */
 4,       /* OutChRateI */
 0,       /* OutChRateD */
 0,       /* OutChRateU */
 10,      /* InTranRate */
 8,       /* OutTranRate */
 n        /* DelFlag */
```

有关使用由这些示例产生的输出结果来完成对稳定队列大小要求的估计的详细信息，请参见 **rs_capacity**。

用法

- 使用 **rs_fillcaptable** 来描述每个要包含在稳定队列估计值中的复制定义的事务。
- **rs_fillcaptable** 维护一个名为 *rs_captable* 的工作表，该表包含数据库中每个复制定义的更改率估计值。
- 使用 **rs_fillcaptable** 的输出作为 **rs_capacity** 存储过程的输入。

另请参见

- rs_capacity （第 499 页）

rs_helpcheckrepdef

显示仅用于定义主键列、带引号的表名或列名以及自定义函数字符串的复制定义。

语法

```
rs_helpcheckrepdef [replication_definition]
```

参数

- **replication_definition** – 指定复制定义，其名称以所输入文本开头。

示例

- **示例 1** – 假设为主 Replication Server 定义了两上复制定义：
  - **authors** – 仅指定主键信息：

```
create replication definition authors
        with primary at NY_DS.pdb1
        (au_id varchar(11),
        au_lname varchar(40) ,
        au_fname varchar(20) ,
```

```
        phone char(12),
        address varchar(40),
        city varchar(20),
        state char(2),
        zip char(5),
        contract bit)
        primary key (au_id)
```

- **titleauthor** – 除了主键外，还指定不同的目标列名：

```
create replication definition titleauthor
        with primary at NY_DS.pdb1
        (au_id varchar(11) as author,
        title_id varchar(6) as title,
        au_ord tinyint,
        royaltyper int)
        primary key (au_id, title_id)
```

如果在主 Replication Server 的 RSSD 或 ERSSD 中输入 **rs_helpcheckrepdef**，则会看到以下内容：

```
Replication Definition Name
---------------------------
authors


(1 row affected)
(return status = 0)
```

用法

- 在主 Replication Server 的 RSSD 或 ERSSD 中执行 **rs_helpcheckrepdef**。
- 如果不为 *replication_definition* 输入任何文本，**rs_helpcheckrepdef** 将会列出仅用于定义主键和带引号的表名或列名的所有复制定义。
- 如果为 *replication_definition* 输入任何文本，**rs_helpcheckrepdef** 将会列出名称以输入的 *replication_definition* 文本开头，并且仅用于定义主键和带引号的表名或列名的所有复制定义。
- 一旦 RepAgent 开始发送主键和带引号的标识符信息，即可以删除由 **rs_helpcheckrepdef** 标识的复制定义。

rs_helpclass

显示错误类和函数字符串类及其主 Replication Server；对于继承类，还会显示父类。

语法

```
rs_helpclass [class_name]
```

参数

- **class_name** – 与错误类或函数字符串类名称相对应的字符串。该字符串必须与整个名称或名称的第一部分相匹配。

示例

- **示例 1** – 显示 Replication Server 的所有错误类和函数字符串类的相关信息。

```
rs_helpclass
```

```
Function String Class(es)     PRS for CLASS      Parent Class
------------------------------ ------------------
------------------------
rs_default_function_class     Not Yet Defined.   Base class
rs_sqlserver_function_class   Not Yet Defined.   Base class
sqlserver2_function_class     TOKYO_RS           rs_default_funct
ion_class

Error Class(es)                PRS for CLASS
------------------------------
----------------------------------------
rs_db2_error_class             Not Yet Defined.
rs_msss_error_class            Not Yet Defined.
rs_oracle_error_class          Not Yet Defined.
rs_sqlserver_error_class       Not Yet Defined.
rs_udb_error_class             Not Yet Defined

RepServer Error Class(es)      PRS for CLASS
------------------------------
----------------------------------------
rs_repserver_error_class       Not Yet Defined.
```

- **示例 2** – 显示 *sqlserver2_function_class* 函数字符串类的相关信息。

```
rs_helpclass sqlserver2_function_class
```

用法

注意： 可以使用命令 **admin show_function_classes** 来获取有关错误类和函数字符串类的详细信息。

- 如果没有输入任何参数，**rs_helpclass** 将列出已定义的所有错误类和函数字符串类。
- 如果提供 *class_name* 字符串，**rs_helpclass** 将列出与 *class_name* 相匹配的错误类和函数字符串类。
- 如果未在 Replication Server 中定义某个类（Adaptive Server 的缺省类就属于这种情况），**rs_helpclass** 会将其作为未定义的类列出，并告诉您如何定义它。

rs_helpclassfstring

为具备函数字符串类范围的函数字符串显示函数字符串信息。

语法

```
rs_helpclassfstring class_name
[, function_name]
```

参数

- **class_name** – 要查看其函数字符串的函数字符串类。
- **function_name** – 与函数名对应的字符串。该字符串必须与整个函数名或函数名的第一部分相匹配。

示例

- **示例 1** – 显示函数字符串类 *rs_sqlserver_function_class* 的所有函数的参数和函数字符串文本。

  ```
  rs_helpclassfstring rs_sqlserver_function_class
  ```

- **示例 2** – 显示 *rs_sqlserver_function_class* 的 **rs_usedb** 函数的函数字符串文本。

  ```
  rs_helpclassfstring rs_sqlserver_function_class, rs_usedb
  ```

  ```
  Function Name  FString Name FSClass Name
  -------------- ------------ ---------------------------
  rs_usedb       rs_usedb     rs_sqlserver_function_class

  FString Text
  ------------------------------------------------------------
      use ?rs_destination_db!sys_raw?
  ```

用法

- 如果不提供 *function_name* 参数，**rs_helpclassfstring** 将显示为函数字符串类的所有函数定义的所有函数字符串。
- 如果提供 *function_name* 字符串，**rs_helpclassfstring** 将显示与 *function_name*（例如 **rs_insert**、**rs_delete**、**rs_update** 和 **rs_select** 或用户定义的函数）相匹配的函数字符串。
- 对于派生的函数字符串类，不显示非自定义的、继承的函数字符串。

rs_helpcounter

显示计数器的相关信息。

语法

```
rs_helpcounter [{sysmon | duration | observer | monitor
    | must_sample | no_reset | keep_old}
    | module_name [, {short | long}] | keyword [, {short | long}]]
```

参数

- **sysmon** – 指定对评估性能和收集复制系统配置文件信息最有用的那些计数器。
- **duration** – 指定所有测量持续时间的计数器（以百分之一秒为单位测量时间间隔）。
- **observer** – 指定记录事件发生次数的计数器。例如，从队列中读取某条消息的次数。
- **monitor** – 指定记录当前值的计数器。例如，最近从队列中读取的消息的大小（以字节为单位）。
- **must_sample** – 指定即使未打开采样也必须进行采样的计数器。
- **no_reset** – 指定在执行 **admin stats, reset** 时不重置值的计数器。
- **keep_old** – 指定同时保留当前值和上一值的计数器。
- **module_name** – 模块的名称：*dsi*、*dsiexec*、*sqt*、*cm*、*dist*、*rsi*、*sqm* 和 *repagent* 等。
- **short** – 指示 Replication Server 输出指定计数器的显示名称、模块名称和计数器说明。
- **long** – 指示 Replication Server 输出 *rs_statcounters* 表中每一列的值。
- **keyword** – 搜索关键字。在计数器长名称、计数器显示名称和计数器说明中进行搜索。

示例

- **示例 1** – 列出所有模块名称和 **rs_helpcounter** 使用语法。

```
1> rs_helpcounter
2> go

ModuleName
------------------------------
CM
DIST
DSI
DSIEXEC
REPAGENT
RSH
RSI
```

```
RSIUSER
SERV
SQM
SQMR
SQT
STS
SYNC
SYNCELE
(12 rows affected)
```

```
How to Use rs_helpcounter
------------------------------------------------------------------
rs_helpcounter  -> Shows module names and help.
rs_helpcounter [ sysmon | duration | observe | monitor
               | must_sample | no_reset | keep_old ]
rs_helpcounter ModuleName    [, {short | long }]
rs_helpcounter keyword       [, { short | long }]
   where “keyword” is part of the counter name, display name or
description
   (return status = 0)
```

- **示例 2** – 列出 SQM 读取器的显示名称、模块名称和计数器说明。

```
rs_helpcounter sqmr, short
```

```
Display Name       Module Name   Counter Description
-------------      ------------   --------------------------------
-----------
BlocksRead         SQMR          Number of 16K blocks read from a
stable
                                    queue by an SQM Reader thread.
ClocksReadCached   SQMR          Number of 16K blocks from cache
read by
                                    an SQM Reader thread.
CmdsRead          SQMR         Commands read from a stable queue by
an
                                    SQM Reader thread.
SQMRReadTime      SQMR         The amount of time taken for SQMR to
read
                                    a block.
SleepsStartQR     SQMR          srv_sleep() calls by an SQM Reader
                               client due to waiting for SQM thread
to
                                    start.
SleepsWriteQ      SQMR          srv_sleep() calls by an SQM read
client
                               due to waiting for the SQM thread to
                                    write.
XNLInterrupted    SQMR          Number of interruptions so far
when
                                reading large messages with partial
                              read. Such interruptions happen due to
                                    time out, unexpected wakeup, or
nonblock
                                    read request, which is marked as
                                    READ_POSTED.
XMLPartials       SQMR         Partial large messages read so far.
```

```
XNLReads          SQMR         Large messages read successfully so
                                  far. This does not count partial
                               messages, or timeout interruptions.
(return status = 0)
```

用法

- **rs_helpcounter** 可用于搜索 *rs_statcounters* 系统表。
- 当不带参数使用时，**rs_helpcounter** 将输出模块和语法列表。
- 有关 RSSD 中存储的计数器状态及其它计数器信息，请参见 rs_statcounters 系统表。

权限

任何用户都可以执行此命令。

rs_helpdb

提供有关 Replication Server 已知的数据库的信息。

语法

```
rs_helpdb [data_server, database]
```

参数

- **data_server** – 要显示其信息的数据库所在的数据服务器。
- **database** – 要显示其信息的数据库。

示例

- **示例 1** –

```
rs_helpdb
```

```
dsname                          dbname              conn_id   dbid
------------------------------    ----------------     -------
----
TOKYO_DS                         TOKYO_RSSD           101       101
SYDNEY_DS                       SYDNEY_RSSD          102       102
TOKYO_DS                         pubs2                105       105
TOKYO_DS                        pubs2_conn2          106       105

controlling_prs                   errorclass
--------------------------------  -------------------------
TOKYO_RS                          rs_sqlserver_error_class
SYDNEY_RS                         rs_sqlserver_error_class
TOKYO_RS                          rs_sqlserver_error_class
TOKYO_RS                          rs_sqlserver_error_class
```

```
repserver_errorclass              funcclass
------------------------------    --------------------------
rs_repserver_error_class          rs_sqlserver_function_class
rs_repserver_error_class          rs_sqlserver_function_class
rs_repserver_error_class          rs_sqlserver_function_class
rs_repserver_error_class          rs_sqlserver_function_class

status
------------------------------------------------------------------------
--
Log Transfer is ON, Distribution is ON
Log Transfer is ON, Distribution is ON
Log Transfer is ON, Distribution is ON
Log Transfer is ON, Distribution is ON
```

用法

- 如果没有提供 *data_server* 和 *database* 参数，**rs_helpdb** 将为 *rs_databases* 系统表中的所有数据库返回结果。
- **rs_helpdb** 在 Replication Server 的 RSSD 中执行。
- 对于每个数据库，**rs_helpdb** 将提供如下信息：

  dsname – 具有数据库的数据服务器的名称。

  dbname – 数据库的名称。

  connid – 分配的 ID 号，如果启用了多路径复制，则用于唯一地标识整个复制系统中到数据库的每个连接。

  dbid – 为在整个复制系统中唯一地标识该数据库而指派的 ID 号。

  controlling_prs – 管理数据库的 Replication Server。

  errorclass – Replication Server 处理从该数据库的数据服务器返回的错误时使用的错误类。

  repserver_errorclass – 用于处理从该数据库的 Replication Server 返回的错误的错误类。

  funcclass – 该数据库使用的函数字符串类。

  status – 判断数据库的日志传输和分发是开启的还是关闭的。

  ltype – 数据库连接的类型（逻辑或物理）。

  ptype – 数据库的类型（活动数据库、备用数据库或逻辑连接）。

rs_helpdbrep

显示与当前 Replication Server 关联的数据库复制定义的相关信息。

语法

```
rs_helpdbrep [db_repdef[, data_server[, database]]]
```

参数

- **db_repdef** – 指定数据库复制定义的名称。
- **data_server** – 指定要显示其数据库复制定义的数据服务器的名称。
- **database** – 指定要显示其数据库复制定义的数据库的名称。

示例

- **示例 1** – 在本示例中，Adaptive Server 显示在当前 Replication Server 中找到的所有数据库复制定义的相关信息：

```
rs_helpdbrep
```

```
DB Rep.Def.Name Primary DS.DB Primary RS Rep.DDL Rep.Sys. Rep.Tab
Rep.Func.
--------------- ------------- ---------- ------ -------- -------
---------
db_rep1         PDS.pdb1      PRS        Yes    Out-List  All     All
db_rep2         PDS.pdb2      PRS        Yes    Out-List  All     All

Rep.Tran. Rep.Upd. Rep.Del. Rep.Ins. Rep.Sel. Creation Date
--------- -------- -------- -------- -------- --------------------
All       All      All      All      All      Nov 26 2008 6:58AM
All       All      All       All      All       Dec 2 2008 6:12PM
```

- **示例 2** – 在本示例中，Adaptive Server 显示单个数据库复制定义 *db_rep1* 的相关信息：

```
rs_helpdbrep db_rep1
```

```
DB Rep.Def.Name Primary DS.DB Primary RS  Rep.DDL  Rep.Sys.
Rep.Tab Rep.Func.
--------------- ------------  ---------  ------- -------  -------
--------
db_rep1         PDS.pdb1      PRS        Yes     Out-List  All    All

Rep.Tran. Rep.Upd. Rep.Del. Rep.Ins. Rep.Sel. Creation Date
--------- -------- -------- -------- -------- --------------------
All       All      All      All      All      Nov 26 2008 6:58AM


Rep.Type      Owner     Name
------------  --------  -------------
Not Rep.Sys.  .         sp_setrepproc

DBRep.Def.Name DBSub.Name ReplicationDS.DB ReplicateRS Creation
Date
-------------- ---------- ---------------- -----------
------------------
db_rep1        db_sub1    RDS1.rdb1        RRS1        Nov 26 2008
6:58AM
db_rep1        db_sub2    RDS2.rdb2        RRS2        Nov 26 2008
6:59AM
```

用法

- Adaptive Server 只显示有关已命名的数据库复制定义的详细信息。
- 参数可以包含通配符“%”。此通配符表示任意字符串。例如，如果为 *db_repdef* 指派了字符串“abc%”，**rs_helpdbrep** 将列出数据库复制定义名称以“abc”为前缀的所有数据库复制定义。

另请参见

- rs_helpdbsub （第 517 页）

rs_helpdbsub

显示与复制数据服务器关联的数据库预订的相关信息。

语法

```
rs_helpdbsub [db_sub[, data_server[, database]]]
```

参数

- **db_sub** – 指定数据库预订。
- **data_server** – 指定要显示其数据库预订的数据服务器的名称。
- **数据库** – 指定要显示其数据库预订的数据库的名称。

示例

- **示例 1** – 在本示例中，Adaptive Server 显示单个数据库预订 *db_sub1* 的相关信息：

```
rs_helpdbsub db_sub1, RDS1, rdb1
```

```
DBSub.Name  ReplicateDS.DB  ReplicateRS  Status at
RRS  DBRep.Def.Name
----------  --------------  -----------  -------------  ---------
------
db_sub1     RDS1.rdb1       RRS1         Validate       db_rep

PrimaryDS.DB  PrimaryRS  Method      Trunc.Table    Creation Date
------------  ---------  ----------  -----------
    -----------------
PDS.pdb1       PRS        Bulk Create  Yes             May 2 2003
3:38PM
```

用法

- 如果不指定任何参数，**rs_helpdbsub** 将列出 Replication Server 中定义的数据库预订。

- 如果仅提供 *db_sub* 参数，**rs_helpdbsub** 将列出所有在 Replication Server 中定义且名称与 *db_sub* 匹配的数据库预订。
- 参数可以包含通配符“%”。此通配符表示任意字符串。例如，如果为 *db_sub* 指派了字符串“abc%”，**rs_helpdbsub** 将列出数据库预订名称以“abc”为前缀的所有数据库预订。

另请参见

- rs_helpdbrep （第 515 页）

rs_helperror

显示映射到给定数据服务器或 Replication Server 错误号的 Replication Server 错误操作。

语法

```
rs_helperror server_error_number [, v]
```

参数

- **server_error_number** – 数据服务器错误号。
- **v** – 显示 Adaptive Server 错误消息文本（如果有）。

示例

- **示例 1** –

```
rs_helperror 2601, v
```

```
DS Error Num      Error Action               Error Class
------------      ------------------------
-------------------------------
2601             Stop Replication         rs_sqlserver_error_class

Adaptive Server Error Message
------------------------------------------------------------------
-----
Attempt to insert duplicate key row in object '%.*s' with unique
index
 '%.*s'%S_EED

RS Error Num     Error Action            Replication Server Error
Class
------------      ------------               ---------------------
---------
```

用法

- 显示所有错误类的错误操作映射。
- 使用 **assign action** 命令将错误操作映射到数据服务器错误号。

另请参见

- assign action （第 172 页）

rs_helpexception

显示例外日志中的事务。

语法

```
rs_helpexception [transaction_id, [, v]]
```

参数

- **transaction_id** – 需要帮助的事务的号码。
- **v** – 在详细列表中包括事务的文本。

示例

- **示例 1** – 显示例外日志中所有事务的摘要信息。

  ```
  rs_helpexception
  ```

- **示例 2** – 显示事务号 1234 的详细信息，包括事务的文本。

  ```
  rs_helpexception 1234, v
  ```

用法

- 如果不输入任何参数，**rs_helpexception** 将显示例外日志中的事务摘要列表，包括所有事务号。
- 如果提供有效的 *transaction_id*，**rs_helpexception** 将显示某事务的详细说明。
- 使用 **rs_delexception** 删除例外日志中的事务。

另请参见

- rs_delexception （第 500 页）

rs_helpfstring

显示与复制定义相关联的函数的参数和函数字符串文本。

语法

```
rs_helpfstring replication_definition
[, function_name]
```

参数

- **replication_definition** – 要查看其函数的表复制定义或函数复制定义。
- **function_name** – 与函数名对应的字符串。该字符串必须与整个函数名或函数名的第一部分相匹配。

示例

- **示例 1** – 显示复制定义 *authors_rep* 的所有函数的参数和函数字符串文本。

  ```
  rs_helpfstring authors_rep
  ```

- **示例 2** – 显示复制定义 *authors_rep* 的 **rs_insert** 函数的参数和函数字符串文本。

  ```
  rs_helpfstring authors_rep, rs_insert
  ```

  ```
  Function String information for Replication Definition.
                         'authors_rep'
  Valid Parameters are:
   Parameter Name                   Datatype
   -------------------------------- -------------------------------
   @au_id                           varchar
   @au_lname                        varchar
   @au_fname                        varchar
   @phone                           char
   @address                         varchar
   @city                            varchar
   @state                           char
   @country                         varchar
   @postalcode                      char

  Rep.Def.Name  Function Name   FString Name  FSClass Name
  ------------- ------------- ------------
  ---------------------------
   authors_rep    rs_insert     rs_insert
  rs_sqlserver_function_class

      --- Begin FString Text ---
   ------------------------------------------------------------
       *** System-Supplied Transact-SQL Statement ***
         --- End FString Text ---
  ```

用法

- 如果不提供 *function_name* 参数，**rs_helpfstring** 将显示为复制定义的所有函数定义的所有函数字符串。
- 如果提供 *function_name* 字符串，则 **rs_helpfstring** 将显示与 *function_name*（例如 **rs_insert**、**rs_delete**、**rs_update** 和 **rs_select** 或用户定义的函数）相匹配的函数字符串。
- 系统生成的缺省函数字符串没有在 RSSD 中存储函数字符串文本。对于这些函数字符串，**rs_helpfstring** 将显示消息“System-Supplied Transact-SQL Statement”（系统提供的 Transact-SQL 语句）。

rs_helpfunc

显示可用于 Replication Server 或某个特定的复制定义的函数的相关信息。

语法

```
rs_helpfunc [replication_definition [, function_name]]
```

参数

- **replication_definition** – 需要其函数信息的复制定义。
- **function_name** – 与函数名对应的字符串。该字符串必须与整个函数名或函数名的第一部分相匹配。

示例

- **示例 1** – 显示所有可用的函数、复制定义和主 Replication Server。还显示每个函数的类范围。

  ```
  rs_helpfunc
  ```

- **示例 2** – 显示复制定义 *authors_rep* 的所有函数的函数信息，包括函数名、参数和数据类型。

  ```
  rs_helpfunc authors_rep
  ```

  ```
  Functions and Parameters for Replication Definition:
                       'authors_rep'
  System Function Names
   ---------------------
   rs_insert
   rs_delete
   rs_update
   rs_select
   rs_select_with_lock

  Parameter(s)     Datatype  Length
   ---------------- -------- ------
   @state           char          2
  ```

```
@postalcode     char         10
@au_id          varchar      11
@phone          char         12
@country        varchar      12
@city           varchar      20
@au_fname       varchar      20
@address        varchar      40
@au_lname       varchar      40
```

- **示例 3** – 显示复制定义 *authors_rep* 的 **rs_insert** 函数的参数和数据类型。

```
rs_helpfunc authors_rep, rs_insert
```

用法

- 如果不指定任何参数，**rs_helpfunc** 将列出 Replication Server 中定义的所有函数。
- 如果您提供 *replication_definition* 名称，则只列出为该复制定义所定义的函数。如果还提供 *function_name* 字符串，**rs_helpfunc** 将显示其名称与 *function_name* 相匹配的函数。
- 如果 **rs_helpfunc** 检测到可能妨碍异步事务的、重复的用户定义函数，它将通知您。

rs_helpobjfstring

显示目标作用域函数字符串的参数和函数字符串文本。

语法

```
rs_helpobjfstring data_server, database, [owner.]object_name[,
function_name]
```

参数

- ***data_server*** – 指定目标作用域函数字符串所用于的复制或备用数据服务器。
- ***数据库*** – 指定目标作用域函数字符串所用于的复制或备用数据库。
- **[owner.]object_name** – 用于查看自定义函数字符串的表或存储过程。如果表有所有者，则指定所有者。
- **function_name** – 与函数名相对应的字符串，必须完整输入。例如，如果函数名为 **rs_writetext**，则不要输入“rs_write”。

示例

- **示例 1** – 假设您要为 **upd_datetime** 存储过程创建目标作用域函数字符串：

```
create function string upd_datetime.upd_datetime
     for database NY_DS.rdb1
     with overwrite
          output language
          'update datetime set
```

```
          row_num = ?row_num!param?,
          datecol = ?datecol!param?,
          timecol = ?timecol!param?,
          ndatecol = ?ndatecol!param?,
          ntimecol = ?ntimecol!param?,
          comment = ?comment!param?
          where
          row_num = ?row_num!param?'
```

如果输入：

- `rs_helpobjfstring NY_DS,rdb1,upd_datetime`

  或

- `rs_helpobjfstring NY_DS,rdb1,upd_datetime,upd_datetime`

您会看到：

```
Function String information for Target Object: 'upd_datetime'.

Object Name     Object Type         Function Name
-----------     -----------         --------------
upd_datetime    stored procedure    upd_datetime

Function String Name    Output Type Option        System Generated
--------------------    -------------------       -----------------
upd_datetime            language not applicable   no


         --- Beginning of Function String Text ---

FString Text
-------------------------------------------------------------------
 update datetime set
         row_num = ?row_num!param?,
         datecol = ?datecol!param?,
         timecol = ?timecol!param?,
         ndatecol = ?ndatecol!param?,
         ntimecol = ?ntimecol!param?,
         comment = ?comment!param?
         where
         row_num = ?row_num!param?

         --- End of Function String Text ---


(return status = 0)
```

- **示例 2** – 为 dbo 表创建目标作用域函数字符串：

```
create function string dbo.datetime.rs_insert
     for database NY_DS.rdb1
     with overwrite
     output language
     'insert datetime values (
             ?row_num!new? ,
             ?datecol!new? ,
```

```
            ?timecol!new? ,
            ?ndatecol!new? ,
            ?ntimecol!new? ,
            ?comment!new?)
      update fn_monitor set insert_count = insert_count + 1'
```

如果输入：

```
rs_helpobjfstring NY_DS,rdb1,'dbo.datetime',rs_insert
```

您会看到：

```
Function String information for Target Object: 'dbo.datetime'.

Object Name       Object Type   Function Name
-------------     -----------   -------------
datetime          table         rs_insert


Function String Name   Output Type Option        System Generated
--------------------   -------------------       ----------------
rs_insert              language not applicable   no



         --- Beginning of Function String Text ---

FString Text
-------------------------------------------------------------------
     insert datetime values (
                          ?row_num!new? ,
                          ?datecol!new? ,
                          ?timecol!new? ,
                          ?ndatecol!new? ,
                          ?ntimecol!new? ,
                          ?comment!new?)
                          update fn_monitor
                          set insert_count =
                          insert_count + 1

        --- End of Function String Text ---
(return status = 0)
```

在此示例中，**create function string** 命令中的对象名包括表所有者 —dbo。

注意： dbo.datetime 必须带引号。

如果在创建函数字符串时忽略表所有者，则输入：

```
rs_helpobjfstring NY_DS,rdb1,datetime,rs_insert
```

您会看到：

```
Target Object 'datetime' does not have customized function string.
(return status = -1
```

- **示例 3** – 为 dbo.tbl1 表创建目标作用域函数字符串：

```
create function string dbo.tbl1.rs_writetext; unitext_fld1 for
NY_DS.rdb1
        output RPC
        'exec update_repl_unitext
                @p_key       = ?p_key!new?,
                @unitext_fld       = ?unitext_fld1!new?,
                @last_chunk         = ?rs_last_text_chunk!sys?'
```

如果输入：

```
rs_helpobjfstring NY_DS,rdb1,'dbo.tbl1',rs_writetext
```

您会看到：

```
Function String information for Target Object: 'dbo.tbl1'.

Object Name        Object Type    Function Name
-------------      -----------    -------------
tbl1               table          rs_writetext

Function String Name     Output Type Option    System Generated
----------------------   ------------------    ----------------
unitext_fld1             RPC  not applicable   no

         --- Beginning of Function String Text ---

 FString Text

-------------------------------------------------------------------
        exec update_repl_unitext
             @p_key = ?p_key!new?,
             @unitext_fld = ?unitext_fld1!new?,
             @last_chunk = ?rs_last_text_chunk!sys?

         --- End of Function String Text ---


(return status = 0)
```

- **示例 4** – 假设您为 dbo.tbl1 表创建目标作用域函数字符串：

```
create function string dbo.tbl1.rs_datarow_for_writetext
for NY_DS.rdb1
        output RPC
        'exec update_txtimg_stat
                @p_key       = ?p_key!new?,
                @txtfld_stat = ?unitext_fld1!text_status?'
```

如果输入：

```
rs_helpobjfstring NY_DS,rdb1,'dbo.tbl1',rs_datarow_for_writetext
```

您会看到：

```
Function String information for Target Object: 'dbo.tbl1'.

Object Name         Object Type    Function Name
-------------       -----------    -------------
tbl1                table          rs_datarow_for_writetext
```

```
Function String Name      Output Type Option     System Generated
--------------------      ------------------     ----------------
rs_datarow_for_writetext  RPC not applicable     no


         --- Beginning of Function String Text ---

 FString Text
 ----------------------------------------------------------------
 exec update_txtimg_stat
                @p_key = ?p_key!new?,
                @txtfld_stat = ?unitext_fld1!text_status?


         --- End of Function String Text ---


(return status = 0)
```

如果输入：

```
rs_helpobjfstring NY_DS,rdb1,'dbo.tbl1'
```

您可以从本示例中看到：

- `rs_helpobjfstring NY_DS,rdb1,'dbo.tbl1',rs_datarow_for_writetext`

  的函数字符串信息，也可以
- 从示例 3 中看到

  ```
  rs_helpobjfstring NY_DS,rdb1,'dbo.tbl1',rs_writetext
  ```

  的函数字符串信息

用法

- 如果不提供 *function_name*，**rs_helpobjfstring** 将显示该对象的所有函数字符串。
- 如果提供 *function_name*，则 **rs_helpobjfstring** 将显示与 *function_name*（例如 **rs_insert**、**rs_delete**、**rs_update** 和 **rs_select** 或用户定义的函数）相匹配的函数字符串。
- 系统生成的缺省函数字符串没有在 RSSD 中存储函数字符串文本。对于这些函数字符串，**rs_helpobjfstring** 将显示“System-Supplied Transact-SQL Statement”（系统提供的 Transact-SQL 语句）。

rs_helppartition

显示 Replication Server 分区的相关信息。

语法

```
rs_helppartition [partition_name]
```

参数

- **partition_name** – 与分区名对应的字符串。该字符串必须与整个分区名或分区名的第一部分相匹配。

示例

- **示例 1** – 显示 Replication Server 的所有可用数据库分区的摘要信息。

```
rs_helppartition
```

```
Displaying all partitions known to 'TOKYO_RS'.
Logical Name                    Size (MB)   Segments Allocated (MB)
------------------------------ -----------
------------------------
partition_1                              20
3
```

- **示例 2** – 显示有关名为 *partition_1* 的分区的详细信息。

```
rs_helppartition partition_1
```

```
Information for stable device: 'partition_1' on 'TOKYO_RS'.
This device is active.
Physical Name                                          Partition ID
------------------------------------------------- ------------
/remote/tyrell2/app/dev/tokyo_rs_p1.dat                       101
Partition Size (MB)  Segments Allocated (MB)
-------------------- -------------------------
20                   5
Inbound Database Queue(s) on this partition:
Connection Name                                  Number of Segments
-------------------------------------------------
------------------
LDS.pubs2                                                         1
TOKYO_DS.TOKYO_RSSD                                               1
Outbound Database Queue(s) on this partition:
Connection Name                                  Number of Segments
-------------------------------------------------
------------------
LDS.pubs2                                                         1
TOKYO_DS.TOKYO_RSSD                                               1
Outbound Replication Server Queue(s) on this partition:
Connection Name                                  Number of Segments
```

```
-------------------------------------------------
------------------
SYDNEY_RS                                                    1
```

用法

- 如果不指定任何参数，**rs_helppartition** 将列出所有 Replication Server 分区的摘要信息。
- 如果提供 *partition_name* 字符串，**rs_helppartition** 将显示其名称与 *partition_name* 相匹配的任何分区的信息。
- 如果 *partition_name* 字符串与某分区名完全匹配，则显示该分区的详细信息，包括逻辑名称和物理名称、总大小、每个分区中分配的 1MB 段的数量以及分区上的队列。
- 如果 *partition_name* 字符串与分区名不完全匹配，则显示其名称与 *partition_name* 相匹配的任何分区或所有已知分区的摘要信息。

rs_helppub

显示关于发布的信息。

语法

```
rs_helppub [publication_name, primary_dataserver, primary_db,
    article_name]
```

示例

- **示例 1** –

```
rs_helppub
```

```
 Publication Name             PRS              Primary DS.DB
 -----------------            -------          -------------
 funcpub                      prim_rs          P_DS.pdb1
 pub1                         prim_rs          P_DS.pdb1
 pub2                         prim_rs          P_DS.pdb1

 Num Articles      Status     Request Date
 ------------      -------    --------------------
 3                 Valid      Mar 23 1998 11:51AM
 7                 Valid      Mar 24 1998 10:41AM
 3                 Valid      Mar 24 1998 11:50AM

(return status = 0)
```

- **示例 2** –

```
rs_helppub funcpub:
```

```
Publication Name        PRS             Primary DS.DB
----------------        -------         --------------
```

```
funcpub                  prim_rs         P_DS.pdb1

Num Articles            Status           Request Date
------------            ------           --------------------
3                       Valid            Mar 23 1998 11:51AM

Article Name                    Replication Definition Type
------------                    ---------------------------
authors                         authors
authors                         authors
publishers                      publishers

Primary Object Name   Replicate Object Name   Request Date
-------------------   ---------------------   --------------------
many_rows_data        many_rows_data          Mar 23 1998 10:01AM
                                              Mar 23 1998 11:51AM

Sub Name     Replicate DS.DB  Owner      Req. Date
----------   ---------------  -----     -------------------------
funcsub1     R_DS.rdb1         sa        Mar 24 1998 11:12AM

(return status = 0)
```

- **示例 3** –

```
rs_helppub funcpub, P_DS, pdb1, publishers:
```

```
Article Name  Publication Name    Replication Definition
------------  ----------------    -------------------------
publishers    funcpub             publishers

Primary Object Name           Replicate Object Name
--------------------          ---------------------
publishers                     publishers

Type      Request Date          Status
----      --------------        ----------
Table     Mar 23 1998 11:51AM   Valid

Where clauses
-----------------------------------------------------------------

  where
  pub_id = "0736"

Sub. Name       Replicate DS.DB   Owner      Req Date
----------      ---------------   -----      --------------------
funcsub1        R_DS.rdb1         sa         Mar 24 1998 11:12AM

(return status = 0)
```

用法

- 如果在主节点执行 **rs_helppub**，则显示在该节点上创建的所有发布的信息。
- 如果在复制节点执行 **rs_helppub**，则只显示已在该节点创建其预订的发布的信息。

- 使用 **rs_helppubsub** 显示对发布或项目的预订的相关信息。
- 使用 **check_subscription** 可获取关于预订状态的最准确的报告。

另请参见

- rs_helppubsub （第 530 页）

rs_helppubsub

显示关于发布预订和项目预订的信息。

语法

```
rs_helppubsub subscription_name, publication_name,
primary_dataserver,
    primary_db, replicate_dataserver, replicate_db
```

示例

- **示例 1** – 列出此节点上所有已知的发布预订：

```
rs_helppubsub
```

```
Subscription Name         Publication Name
-------------------       ----------------
funcsub1                  funcpub

Primary DS.DB         Replicate DS.DB      PRS Status    RRS Status
--------------        ---------------      ----------    ----------
P_DS.pdb1              R_DS.rdb1             Unknown       Valid

Owner    Request Date
------   --------------------
sa       Mar 24 2007 11:12AM
(1 row affected)

Subscription Name               Article Name      Replication
Definition
------------------              -------------     ---------------
-------
funcsub1                        authors           authors

PRS Status     RRS Status     Request Date           Autocorrection
----------     -----------    --------------------
    --------------
Unknown         Valid          Mar 24 2007 11:11AM     off

Subscribe to Truncate Table    Dynamic SQL
---------------------------    -----------
Unknown                        On
(1 row affected, return status = 0)
```

- **示例 2** – 列出名为 *sub* 的所有发布预订。

```
rs_helppubsub sub
```

- **示例 3** – 为名为 *pub* 的发布列出所有名为 *sub* 的发布预订。

```
rs_helppubsub sub, pub
```

- **示例 4** – 列出指定发布的所有名为 sub 的预订。

```
rs_helppubsub sub, pub, primary_dataserver, primary_db
```

- **示例 5** – 列出该组中的发布预订和项目预订。

```
rs_helppubsub sub, pub, primary_dataserver, primary_db,
replicate_dataserver, replicate_db
```

```
Subscription Name     Publication Name     Primary DS.DB
 -----------------    ----------------     ----------------
 sub                  pub                  ost_cardhu_2.pdb1

 Replicate DS.DB      PRS Status     RRS Status     Owner
 -----------------    ----------     ----------     ---------
 ost_cardhu_2.rdb1    Unknown        Valid          rdb1_owner

 Request Date         Subscription Name     Article Name
 ----------------     -----------------     ------------
 February 25 1998     sub                   article1
                                            article2
                      sub                   article3
                      sub                   article4
                      sub                   article5

 PRS Status   RRS Status   Request Date   Replication Definition
 ----------   ----------   ------------   ----------------------
 Unknown      VALID        Feb 25, 1998   repdef1
                                          repdef2
 Unknown      VALID        Feb 25, 1998   repdef3
 Unknown      VALID        Feb 25, 1998   repdef4
 Unknown      VALID        Feb 25, 1998   repdef5

 Autocorrection   Subscribe to Truncate Table    Dynamic SQL
 --------------   ---------------------------    -----------
 on               off                            on
 off              on                             on
 off              off                            on
 off              off                            on
```

用法

- 使用 **rs_helppub** 确定某个项目或发布的所有预订。
- 使用 **check_subscription** 可获取关于预订状态的最准确的报告。

另请参见

- rs_helppub （第 528 页）

rs_helprep

显示关于复制定义的信息。

语法

```
rs_helprep [replication_definition]
```

参数

- **replication_definition** – 与复制定义名称对应的字符串。该字符串必须与整个复制定义名称或复制定义名称的第一部分相匹配。

示例

- 示例 **1** – `rs_helprep`

Rep Def	PRS	Primary DS.DB	Primary Table	Replicate Table	Type
authors	cardhu_11	cardhu_10.pdb1	authors	ling.authors_r1	Tbl
authors1	cardhu_11	cardhu_10.pdb1	authors	authors_r2	Tbl
discounts	cardhu_11	cardhu_10.pdb1	discounts	discounts	Tbl
publishers	cardhu_11	cardhu_10.pdb1	publishers	ling.publishers_r1	Tbl
publishers1	cardhu_11	cardhu_10.pdb1	publishers	publishers_r2	Tbl
roysched	cardhu_11	cardhu_10.pdb1	roysched	roysched	Tbl
rs_classes	cardhu_11	cardhu_10.emb	rs_classes	Tbl	
rs_columns	cardhu_11	cardhu_10.emb	rs_columns	Tbl	
rs_databases	cardhu_11	cardhu_10.emb	rs_databases	Tbl	
rs_erroractions	cardhu_11	cardhu_10.emb	rs_erroractions	Tbl	
rs_funcstrings	cardhu_11	cardhu_10.emb	rs_functstrings	Tbl	
rs_functions	cardhu_11	cardhu_10.emb	rs_functions	Tbl	
rs_objects	cardhu_11	cardhu_10.emb	rs_objects	Tbl	
rs_routes	cardhu_11	cardhu_10.emb	rs_routes	Tbl	
rs_systext	cardhu_11	cardhu_10.emb	rs_systext	Tbl	

- 示例 **2** – 显示有关使用 **create function replication definition** 创建的 authors 复制定义的信息：

```
rs_helprep authors
```

```
Replication Definition Name  PRS                 Type Creation Date
---------------------------- ------------------- ----
--------------------
authors                      primary_rs           Tbl  Nov 26, 2008
1:48PM

PDS.DB               Primary Owner        Primary Table
-------------------- -------------------- --------------------
pds.pdb                                   authors

Replicate Owner      Replicate Table
-------------------- --------------------
                     authors

Send Min Cols. Used by Standby Min Vers Dynamic SQL SQL Stmt.Rep.
-------------- --------------- -------- ----------- -------------
No             No                1000     On          UD

Col. Name   Rep. Col. Name   Datatype   Len.   Pri.
Col.   Searchable
---------   --------------   --------   ----   ---------   ------
----
au_id       au_id            varchar    11           1           1
au_lname    au_lname         varchar    40           0           1
au_fname    au_fname         varchar    20           0           1
```

- **示例 3** – 显示有关使用 **create applied function replication definition** 创建的 R1_app 复制定义的信息：

```
rs_helprep R1_app
```

```
Replication Definition Name  PRS              Type  Creation Date
---------------------------  --------------  ----  --------------
---------
R1_app                        ost_replnx4_12  Func  Feb 22 2008
12:15PM

PDS.DB    Primary Function  Replicate Function  Used by
Standby  Func_type
--------  ----------------  ------------------  ---------------
-------
PDS.pdb1  R1              R1_rep            No              Applied

Parameter  Datatype  Length  Searchable
---------  --------  ------  ----------
a          int       4        0

Function Name  FString Class              FString Source  FString
Name
-------------  --------------------------  --------------  -----
--------
R1             rs_sqlserver_function_class  Class Default    R1

Subscriptions known at this Site 'ost_replnx4_12'.
```

```
Subscription Name    Replicate DS.DB   Owner   Creation Date
------------------   ---------------   -----   --------------

(return status = 0)
```

- **示例 4** – 显示有关使用 **create request function replication definition** 创建的 R1_req 复制定义的信息：

```
rs_helprep R1_req
```

```
Replication Definition Name   PRS              Type   Creation Date
---------------------------   --------------   ----   -----------
--------
R1_req                        ost_replnx4_12   Func   Feb 22 2008
12:15PM

PDS.DB    Primary Function  Replicate Function  Used by
Standby  Func_type
--------  ----------------  ------------------  ---------------
---------
PDS.pdb1  R2              R2_rep            No              Request

Parameter      Datatype      Length      Searchable
---------      ---------     ------      ----------
a              int                4      0

Function Name  FString Class              FString Source  FString
Name
-------------  --------------------------  --------------  -----
--------
R2             rs_sqlserver_function_class  Class Default   R2

Subscriptions known at this Site 'ost_replnx4_12'.


Subscription Name       Replicate DS.DB     Owner    Creation Date
----------------------  ---------------     -----    ------------
--

(return status = 0)
```

- **示例 5** – 给定表和复制定义：

```
create table t1 (c1 int, c2 int)
```

```
create replication definition r1
   with primary at ost_wasatch_08.pdb1
   with all tables named t1
   (c1 int, "c2" int quoted)
    primary key (c1)
```

`rs_helprep r1` 将 *c2* 显示为带引号的标识符：

```
Replication Definition Name  PRS                 Type Creation Date
---------------------------- -------------------- ----
-------------------
r1                           ost_wasatch_09       Tbl  Nov 11, 2008
2:28PM
```

```
PDS.DB                Primary Owner         Primary Table
--------------------- --------------------- --------------------
ost_wasatch_08.pdb1                         t1

Replicate Owner       Replicate Table
--------------------- --------------------
                      t1

Send Min Cols. Used by Standby Min Vers Dynamic SQL SQL Stmt. Rep.
-------------- --------------- -------- -----------
--------------
No             No               1000     On           None

Col. Name   Rep. Col. Name   Datatype   Len.   Pri.
Col.   Searchable
---------   --------------   --------   ----   ---------   ------
----
c1          c1               int           4           1            0
"c2"        "c2"             int           4           0            0

Function Name   FString Class                  FString
Source   FString Name
-------------   ---------------------------   --------------   --
----------
rs_delete       rs_sqlserver_function_class   Class
Default    rs_delete
rs_insert       rs_sqlserver_function_class   Class
Default    rs_insert
rs_select       rs_sqlserver_function_class   Class
Default    rs_select
rs_select_      rs_sqlserver_function_class   Class
Default    rs_select_
with_lock                                                    with_lock
rs_truncate     rs_sqlserver_function_class   Class
Default    rs_truncate
rs_update       rs_sqlserver_function_class   Class
Default    rs_update

Subscriptions known at this Site 'ost_wasatch_09'.
Subscription Name   Replicate DS.DB   Owner   Creation Date
-----------------   ---------------   -----   -------------
(return status = 0)
```

- **示例 6** – 给定上例中定义的表和复制定义，在将 *t1* 定义为带引号的标识符时：

```
alter replication definition r1
alter replicate table name "t1" quoted
```

rs_helprep r1 将 *c2* 和 *t1* 显示为带引号的标识符：

```
Replication Definition Name  PRS                Type Creation Date
---------------------------- ------------------ ----
-------------------
r1                           ost_wasatch_09      Tbl  Nov 11, 2008
2:28PM

PDS.DB                Primary Owner         Primary Table
```

```
------------------- -------------------- --------------------
ost_wasatch_08.pdb1                      "t1"

Replicate Owner      Replicate Table
-------------------- --------------------
                     "t1"

Send Min Cols. Used by Standby Min Vers Dynamic SQL SQL Stmt. Rep.
-------------- --------------- -------- -----------
--------------
No             No              1000     On          None

Col. Name   Rep. Col. Name   Datatype   Len.   Pri.
Col.   Searchable
---------   --------------   --------   ----   ---------   ------
----
c1          c1               int          4          1            0
"c2"        "c2"             int          4          0            0

Function Name   FString Class                  FString
Source   FString Name
-------------   --------------------------   --------------   --
----------
rs_delete       rs_sqlserver_function_class   Class
Default    rs_delete
rs_insert       rs_sqlserver_function_class   Class
Default    rs_insert
rs_select       rs_sqlserver_function_class   Class
Default    rs_select
rs_select_      rs_sqlserver_function_class   Class
Default    rs_select_
with_lock                                                  with_lock
rs_truncate     rs_sqlserver_function_class   Class
Default    rs_truncate
rs_update       rs_sqlserver_function_class   Class
Default    rs_update

Subscriptions known at this Site 'ost_wasatch_09'.
Subscription Name   Replicate DS.DB   Owner   Creation Date
-----------------   ---------------   -----   -------------
(return status = 0)
```

- **示例 7** – 给定上例中定义的复制定义，在将 *c2* 定义为不带引号的标识符时：

```
alter replication definition r1
alter columns c2 not quoted
```

`rs_helprep r1` 将 *t1* 显示为唯一的带引号标识符：

```
Replication Definition Name  PRS                 Type Creation Date
---------------------------- ------------------- ----
-------------------
r1                           ost_wasatch_09      Tbl  Nov 11, 2008
2:28PM

PDS.DB               Primary Owner        Primary Table
-------------------- -------------------- --------------------
```

```
ost_wasatch_08.pdb1                         "t1"

Replicate Owner      Replicate Table
-------------------- --------------------
                     "t1"

Send Min Cols. Used by Standby Min Vers Dynamic SQL SQL Stmt. Rep.
-------------- --------------- -------- -----------
--------------
No             No              1000     On          None

Col. Name   Rep. Col. Name   Datatype   Len.   Pri.
Col.   Searchable
---------   --------------   --------   ----   ---------   ------
----
c1          c1               int          4          1            0
c2          c2               int          4          0            0

Function Name   FString Class                  FString
Source   FString Name
-------------   --------------------------   --------------   --
----------
rs_delete       rs_sqlserver_function_class   Class
Default    rs_delete
rs_insert       rs_sqlserver_function_class   Class
Default    rs_insert
rs_select       rs_sqlserver_function_class   Class
Default    rs_select
rs_select_      rs_sqlserver_function_class   Class
Default    rs_select_
with_lock                                                  with_lock
rs_truncate     rs_sqlserver_function_class   Class
Default    rs_truncate
rs_update       rs_sqlserver_function_class   Class
Default    rs_update

Subscriptions known at this Site 'ost_wasatch_09'.
Subscription Name   Replicate DS.DB   Owner   Creation Date
-----------------   ---------------   -----   -------------
(return status = 0)
```

- **示例 8** – 若要显示有关使用 **create function replication definition** 创建的“authors”复制定义的信息，请输入：

```
rs_helprep authors
```

请参见输出中的 Ref Objowner 和 Ref Objname 列：

```
Replication Definition Name  PRS                Type Creation Date
---------------------------- ------------------- ----
--------------------
authors                      primary_rs          Tbl  Nov 26, 2008
1:48PM

PDS.DB               Primary Owner        Primary Table
-------------------- -------------------- --------------------
pds.pdb                                   authors
```

```
Replicate Owner      Replicate Table
-------------------- --------------------
                     authors

Send Min Cols. Used by Standby Min Vers Dynamic SQL SQL Stmt.Rep.
-------------- --------------- -------- ----------- -------------
No             No                1000     On          UD

Col. Name   Rep. Col. Name   Datatype   Len.   Pri.
Col.   Searchable
---------   --------------   --------   ----   ---------   ------
----
au_id       au_id            varchar    11           1           1
au_lname    au_lname         varchar    40           0           1
au_fname    au_fname         varchar    20           0           1

Ref. Objowner   Ref. Objname
-------------   ------------


table2
```

用法

- 除非输入参数，否则 **rs_helprep** 将列出 Replication Server 中所有复制定义的摘要信息。
- 如果提供 *replication_definition* 字符串，**rs_helprep** 将显示其名称与 *replication_definition* 相匹配的所有复制定义的信息。
- 如果 *replication_definition* 字符串与某个复制定义名称完全匹配，则显示有关该复制定义的详细信息。这些信息包括：主 Replication Server、数据服务器和数据库、复制定义列、为该复制定义所定义函数，以及 Replication Server 已知的复制定义的预订。
- 此处显示的详细信息与表复制定义、函数复制定义和系统表复制定义略有不同。
- 如果 *replication_definition* 字符串与某个复制定义名称不完全匹配，则显示与 *replication_definition* 相匹配的所有复制定义的摘要信息。
- 在显示带引号的标识符时，将使用双引号字符将标识符引起来。
- **rs_helprep** 显示有关用于实时装载 (RTL) 和高容量自适应复制 (HVAR) 的表引用的信息。
- **rs_helprep** 不显示数据库复制定义。您可以使用 **rs_helpdbrep** 来显示数据库复制定义。

rs_helprepdb

显示有关当前 Replication Server 中带有对复制定义的预订的数据库的信息。

语法

```
rs_helprepdb [data_server, database]
```

参数

- **data_server** – 要显示其信息的数据库所在的数据服务器。
- **database** – 要显示其信息的数据库。

示例

- **示例 1** – 显示有关当前 Replication Server 中带有对复制定义的预订的所有数据库的信息。

```
rs_helprepdb
```

```
dsname             dbname          dbid          controlling_prs
-----------------  --------------  ------------  ----------------
SYDNEY_DS          SYDNEY_RSSD     102           SYNDEY_RS
```

- **示例 2** – 显示有关指定的数据服务器和数据库的信息。

```
rs_helprepdb SYDNEY_DS, pubs2
```

```
dsname             dbname          dbid          controlling_prs
-----------------  --------------  ------------  ----------------
SYDNEY_DS          pubs2           104           SYDNEY_RS
```

用法

- 在主 Replication Server 的 RSSD 中执行 **rs_helprepdb**。
- 除非指定 *data_server* 和 *database* 参数，否则 **rs_helprepdb** 将列出预订了任何 Replication Server 的复制定义的所有数据库。将显示每个数据服务器和数据库的数据库 ID 和管理 Replication Server。
- 如果提供 *data_server* 和 *database* 参数，则 **rs_helprepdb** 只显示有关指定数据库的信息。

rs_helprepversion

显示有关当前 Replication Server 中复制定义版本的信息。

语法

```
rs_helprepversion {repdef_name | repdef_version_id}
```

参数

- **repdef_name** – 复制定义名称。
- **repdef_version_id** – 复制定义版本 ID。

示例

- **示例 1** – 当提供复制定义名称 types11_pdb1 时，显示有关所有版本的复制定义的信息。

  ```
  rs_helprepversion types11_pdb1
  ```

  输出为:

  ```
  Repdef Version Name            Repdef Version ID    Active    Active
                                                     Inbound   Oubound
  ------------------------------------------------------------------
  -----
  types11_pdb1                     0x0107006500000067   Yes        No
  rs_drp0106006500000674a955c45  0x0106006500000067   No        Yes
  rs_drp0105006500000674a955c40  0x0105006500000067   No         No
  rs_drp0104006500000674a955c3f  0x0104006500000067   No         No
  rs_drp0103006500000674a955c3d  0x0103006500000067   No         No
  rs_drp0102006500000674a955c3c  0x0102006500000067   No         No
  rs_drp0101006500000674a955c3b  0x0101006500000067   No         No
  rs_drp0100006500000674a955c3a  0x0100006500000067   No         No
  (return status = 0)
  ```

- **示例 2** – 如果通过提供复制定义版本 ID（例如本例中的 0x0106006500000067）来指定复制定义版本，**rs_helprepversion** 将显示该复制定义版本的一般信息和列信息:

  ```
  rs_helprepversion 0x0106006500000067
  ```

  输出为:

  ```
  Repdef Version Name              Repdef Version
  ID    Active    Active
                                                     Inbound   Oubound
  ------------------------------------------------------------------
  ------
  rs_drp0106006500000674a955c45   0x0106006500000067   No       Yes
  ```

  ```
  Column      Replicate   Datatype  Len   Pri  Searchable  Ref
    Ref
  ```

```
Name         Col Name                      Col              Objowner
  Objname
------------ ------------ --------- ----- ---- ------------ ---------
-------
charcol      charcol      varchar   255     0         0
floatcol     floatcol     float       8     0         0
datecol      datecol      datetime    8     0         0
smdatecol    smdatecol    smalldatet  4     0         0
moneycol     moneycol     money       8     0         0
smmoneycol   smmoneycol   smallmoney  4     0         0
intcol       intcol       int         4     0         0
smintcol     smintcol     smallint    2     0         0
tinyintcol   tinyintcol   tinyint     1     0         0
row_num      row_num      int         4     1         0
(return status = 0)
```

用法

rs_helprepversion 显示有关复制定义版本的信息：

- 活动的入站复制定义版本 – 执行程序用它将数据打包到入站队列中。
- 活动的出站复制定义版本 – 分配器用它将数据打包到出站队列中。

另请参见

- alter replication definition （第 152 页）
- alter applied function replication definition （第 107 页）
- alter request function replication definition （第 159 页）
- rs_helprep （第 532 页）
- rs_send_repserver_cmd （第 546 页）

rs_helproute

提供路由的状态信息。

语法

```
rs_helproute [replication_server]
```

参数

- **replication_server** – 需要其路由状态信息的 Replication Server 的名称。

示例

- **示例 1** – 从 TOKYO_RS 到 SYDNEY_RS 的路由目前处于活动状态。

  ```
  rs_helproute
  ```

```
route                          route_status
-------------------------      ------------
TOKYO_RS -----> SYDNEY_RS       Active
```

用法

- 除非指定 *replication_server* 参数，否则 **rs_helproute** 将显示当前 Replication Server 已知的所有路由的信息。
- 如果提供 *replication_server*，则只显示起止于该 Replication Server 的路由的信息。
- Replication Server 使用定义的协议来创建和删除源 Replication Server 和目标 Replication Server 之间的路由。在使用该协议的过程中，路由将经历各种状态。在源 Replication Server 或目标 Replication Server 的 RSSD 上执行 **rs_helproute** 将显示协议的当前状态。
- 对于每个路由，**rs_helproute** 返回两种信息：
  - 路由状态

    该状态反映路由协议的状态。各路由的信息取决于执行 **rs_helproute** 的位置 — 路由的源位置还是目标位置。
  - 系统表预订的列表

    如果您正在创建路由，您会看到关于正在创建的系统表预订的信息。如果您正在删除路由，该列表会通知您正在删除哪个系统表预订。

    路由协议通常处理系统表预订。此信息帮助您确定是哪些预订妨碍您继续进行协议的下一步。如果没有列出系统表预订，则表明协议当前不存在与系统表预订有关的问题。

    系统表预订的实现或取消实现不完全是一个常见问题。如果在创建、删除或更改路由时发现任何问题，请检查 **rs_helproute** 输出以了解有关预订状态的信息。

rs_helpsub

显示关于预订的信息。

语法

```
rs_helpsub
 [subscription_name [, replication_definition
    [, data_server, database]]]
```

参数

- **subscription_name** – 与预订名称对应的字符串。该字符串必须与整个预订名称或预订名称的第一部分相匹配。
- **replication_definition** – 所预订的复制定义。
- **data_server** – 包含预订数据的数据库所在的数据服务器。

- **database** – 包含预订数据的数据库。

示例

- **示例 1** – 显示所有可用预订的摘要信息。RRS 列中的“Unknown”状态反映出当前 Replication Server 不知道列出的 Replication Server（主 Replication Server）上的预订状态：

```
rs_helpsub
```

```
** This Site is primary_rs **
                                                                Status at
 Subscription Name Rep. Def. Name  Replicate DS.DB  A/C
RRS      PRS
 ----------------- --------------- ---------------- --- --------
------
 authors_1         authors         RDS.rdb          0   Unknown  Valid
 many_rows_1       many_rows       RDS.rdb          0   Unknown  Valid
 publishers_1       publishers       RDS.rdb           0   Unknown
Valid
 titleauthor_1      titleauthor      RDS.rdb           0   Unknown
Valid
 titles_1          titles          RDS.rdb          0   Unknown  Valid

Dynamic SQL
-----------
On
On
On
On
On
(return status = 0)
```

- **示例 2** – 显示有关 *authors_sub* 预订的详细信息：

```
rs_helpsub authors_sub
```

```
Subscription Name Rep. Def. Name  Replicate DS.DB  A/C RRS      PRS
----------------- --------------- ---------------- --- --------
------
authors_sub         authors_rep       RDS.rdb         0
Defined  Unknown

Dynamic SQL    Owner                  Creation Date
-----------    --------------------   -------------
On             sa                     Oct 2 2007

Subscription  Text
------------------------------------------------------------------
----
 create subscription authors_sub
   for authors_rep
   with replicate at RDS.rdb
   where
```

```
   state = "CA"
(return status = 0)
```

用法

- 如果不指定任何参数，**rs_helpsub** 将列出 Replication Server 中定义的所有预订的摘要信息。这些信息包括：复制定义、复制数据服务器和数据库、自动更正状态以及复制 Replication Server 和主 Replication Server 上的预订实现状态。
- 如果提供 *subscription_name* 字符串，**rs_helpsub** 将显示其名称与 *subscription_name* 相匹配的所有预订的信息。
- 如果 *subscription_name* 字符串与某个预订名称完全匹配，还将显示所有者、创建日期和预订的文本。
- 如果 *subscription_name* 字符串与某个预订名称不完全匹配，则显示其名称与 *subscription_name* 相匹配的所有预订的摘要信息。
- 如果还提供了 *replication_definition*，**rs_helpsub** 将只显示该复制定义的预订的信息。
- **rs_helpsub** 不显示预订复制定义。您可以使用 **rs_helpdbsub** 来显示预订复制定义。

rs_helpuser

显示有关 Replication Server 已知的用户登录名的信息。

语法

```
rs_helpuser [user]
```

参数

- **user** – 需要了解其信息的用户登录名。

示例

- **示例 1** – 显示有关所有用户的信息。

```
rs_helpuser
```

```
                        Users and Privileges Known at Site repl_rs
Primary Users
User Name               Permission(s) Name
----------------------- ------------------------------
TOKYO_RS_id_user        no grants
sa                      sa
TOKYO_RS_ra             connect source
TOKYO_RS_rsi            connect source
repuser                 create object
TOKYO_RSSD_prim         connect source, primary subscr

Maintenance Users
```

```
User name                Destination DS.DB
-----------------------  ----------------------------------------
-----
TOKYO_RSSD_maint         TOKYO_DS.TOKYO_RSSD
pubs2_maint              TOKYO_DS.pubs2
pubs2_maint              SYDNEY_DS.pubs2sb
```

- **示例 2** – 显示有关 *pubs2_maint* 用户的信息。

```
rs_helpuser pubs2_maint
```

```
                        Users and Privileges Known at Site TOKYO_RS
Primary User(s)
User Name               Permission Name
----------------------- ------------------------------
pubs2_maint             TOKYO_DS.pubs2
pubs2_maint             SYDNEY_DS.pubs2sb
```

用法

- 除非输入参数，否则 **rs_helpuser** 将显示有关当前 Replication Server 已知的所有用户登录名的信息。
- 如果提供 *user* 登录名参数，**rs_helpuser** 将只显示有关该用户登录名的信息。

rs_helpreptable

显示有关针对主表创建的复制定义的信息。

语法

```
rs_helpreptable database, [owner,] table
```

参数

- **database** – 在其中创建表的数据库。
- **owner** – 表的所有者。
- **table** – 表的名称。

示例

- **示例 1** –

```
rs_helpreptable pdb1, authors
```

复制定义名称	Primary Owner	Primary Table	Primary Owner	Replicate Table	Used Standby	Min Vers
authors		authors	ling	authors_r1	Yes	1000
authors1		authors		authors_r2	No	1000

用法

- 只显示用户定义的表复制定义。

rs_init_erroractions

初始化新的错误类。

注意： 将不再支持 **rs_init_erroractions**。若要初始化新类，Sybase 建议您使用带有 **set template to** 选项的 **create error class**。

语法

```
rs_init_erroractions new_error_class, template_class
```

参数

- **new_error_class** – 已创建的新错误类的名称。
- **template_class** – 要作为新错误类模板的错误类的名称。

示例

- **示例 1** – 根据模板错误类 *rs_sqlserver_error_class* 创建错误类 *new_class*。

  ```
  rs_init_erroractions new_class, rs_sqlserver_error_class
  ```

用法

- 模板错误类可以是用户定义的错误类，也可以是系统提供的错误类，例如 *rs_sqlserver_error_class*。
- 使用 **create error class** 命令在主 Replication Server 中创建该错误类的新错误类。然后使用 **rs_init_erroractions** 初始化这个新类。

另请参见

- create error class （第 228 页）

rs_send_repserver_cmd

直接在主数据库中执行复制定义更改请求。

语法

```
rs_send_repserver_cmd 'rs_api'
```

参数

- **rs_api** – 包含您为 **rs_send_repserver_cmd** 指定的复制定义复制命令语言 (RCL) 命令和参数。*rs_api* 是一个 *varchar* 参数，对于 Adaptive Server，最大长度为 16370 字节，对于 Oracle 为 4000 字节，对于 Microsoft SQL Server 为 8000 字节。

  用单引号括起 *rs_api* 并将字符串中的每个单引号替换为两个单引号。

  如果 *rs_api* 的参数长度对于创建或更改复制定义请求太短，您可以将请求拆分为两个或多个请求。

示例

- **示例 1** – 在主数据库中执行“authors”**alter replication definition** 请求以删除 address、city、state 和 zip 列：

```
exec rs_send_repserver_cmd ‘alter replication
definition authors drop address, city, state, zip’
```

- **示例 2** – 如果复制定义 RCL 的长度大于 *rs_api* 的允许最大长度，您可以将请求拆分为两个或多个请求。

```
exec rs_send_repserver_cmd ‘alter replication
definition authors drop address, city’
go
exec rs_send_repserver_cmd ‘alter replication
definition authors drop state, zip’
```

- **示例 3** – 在此示例中，您需要用双引号括起“authors”，并用两个单引号括起“off”：

```
exec rs_send_repserver_cmd ‘alter replication definition
“authors” replicate sqldml ‘‘off’’’
```

用法

- 在主数据库中使用 **rs_send_repserver_cmd** 之前，请使用 **admin verify_repserver_cmd** 验证是否可以在 Replication Server 上成功执行复制定义请求。
- Replication Server 支持将 **rs_send_repserver_cmd** 用于以下复制定义命令：
  - **alter replication definition**
  - **create replication definition**
  - **drop replication definition**
  - **alter applied function replication definition**
  - **create applied function replication definition**
  - **alter request function replication definition**
  - **create request function replication definition**

注意： 除了 Adaptive Server 外，Replication Server 还将对 **rs_send_repserver_cmd** 的支持扩展到了受支持的以下非 ASE 数据库版本：Microsoft SQL Server 和 Oracle。有关支持的数据库版本，请参见《Replication Agent 发行公告》。

- 当您在主数据库中执行 **rs_send_repserver_cmd** 时，Replication Agent 会将存储在 *rs_api* 中的 RCL 命令发送到 Replication Server，Replication Server 随后将执行该 RCL 命令。这可确保 Replication Server 使用适当的复制定义版本复制主数据 — primary data before the **rs_send_repserver_cmd** 之前的主数据使用旧的复制定义版本进行复制，而 **rs_send_repserver_cmd** 之前的主数据使用新的复制定义版本进行复制。
- 您并不总是需要直接从主数据服务器发出复制定义更改请求。例如，在以下情况下，您可以直接从主 Replication Server 执行 **alter replication definition** 请求：
  - 没有对复制定义的预订
  - 有对复制定义的预订，但主数据库日志中没有表或存储过程的数据
  - 您要在表复制定义中添加或删除可搜索列
  - 您要在函数复制定义中添加或删除可搜索参数
  - 您要更改复制定义以打开或关闭动态 SQL

警告！ 由于 Replication Server 接受 Replication Agent 发送给 Replication Server 的所有命令，因此您必须在主数据库中控制对 **rs_send_repserver_cmd** 的访问。

另请参见

- admin verify_repserver_cmd （第 84 页）
- alter replication definition （第 152 页）
- create replication definition （第 258 页）
- drop replication definition （第 308 页）
- alter applied function replication definition （第 107 页）
- create applied function replication definition （第 206 页）
- alter request function replication definition （第 159 页）
- create request function replication definition （第 269 页）
- sysadmin skip_bad_repserver_cmd （第 374 页）

rs_ticket

主数据库中的一个存储过程，用于监控 Replication Server 性能、模块心跳、复制运行状况和表级停顿。

语法

```
rs_ticket h1 [, h2 [, h3 [, h4]]]
```

参数

- **h1 [, h2 [, h3 [, h4]]]** – short *varchar* 字符串中的标头信息。

示例

- **示例 1** – 定期执行 **rs_ticket**:

```
Exec rs_ticket 'heartbeat', 'beat-sequence-number'
```

- **示例 2** – 要测量性能，请从主数据库中执行以下命令:

```
Exec rs_ticket 'start'
Execute replication benchmarks
Exec rs_ticket 'stop'
```

用法

- **rs_ticket** 存储过程的票据版本号 V=2，票据大小为 1024 个字节。
- 如果应用程序只识别版本 1 票据，则调用 **rs_ticket_v1** 将以版本 1 格式生成票据。**rs_ticket_v1** 的语法如下:

```
rs_ticket_v1 h1 [, h2 [, h3 [, h4]]]
```

- **rs_ticket** 执行以下命令:

```
rs_marker 'rs_ticket rs_ticket_param'
```

  要避免发出格式错误的 **rs_marker** 并强制实施 *rs_ticket_param* 标准，您应该调用 **rs_ticket** 而不是 **rs_marker**。如果直接调用 **rs_marker** 并构成不正确的 **rs_marker** 子命令，Replication Server 将拒绝 **rs_marker** 并关闭 RepAgent 连接。在这种情况下，您必须从事务日志中跳过 **rs_marker**，这可能会导致数据丢失。
- Replication Server 的 EXEC、DIST、RSI 和 DSI 模块用于分析和处理 **rs_ticket** 子命令:
  - 当 EXEC 处理 **rs_ticket** 时，它将在 *rs_ticket_param* 后面附加一个时间戳，然后附加从 RepAgent 中收到的总字节数。EXEC 时间戳采用“EXEC(spid)=mm/dd/yy hh:mm:ss.ddd”格式。字节信息为“B(spid)=ddd”。EXEC 会将 **rs_ticket** 写回到入站队列中。
  - 当 DIST 处理 **rs_ticket** 时，它会在 *rs_ticket_param* 后面附加另一个时间戳。DIST 时间戳采用“DIST(spid)=mm/dd/yy hh:mm:ss.ddd”格式。
  - 当 RSI 处理 **rs_ticket** 时，它也会在 *rs_ticket_param* 后面附加另一个时间戳。RSI 时间戳采用“RSI(spid)=mm/dd/yy hh:mm:ss.ddd”格式。
  - 当 DSI 处理 **rs_ticket** 时，它也会在 *rs_ticket_param* 后面附加另一个时间戳。DSI 时间戳采用“DSI(spid)=mm/dd/yy hh:mm:ss.ddd”格式。
- 没有对 **rs_ticket** 的任何预订。DIST 不会将 **rs_ticket** 发送到 DSI，除非复制节点中至少有一个预订。
- **rs_ticket** 是轻量和非侵入性的，可以用于测试环境和生产环境。

- 从复制路径中完全刷新数据后，**rs_ticket** 将会通知您，而无需停顿 Replication Server。
- EXEC、DIST、RSI 和 DSI 线程通过 RSTicket 计数器跟踪 **rs_ticket** 的移动情况。每个线程具有一个 RSTicket 计数器，每当对应的线程收到 **rs_ticket** 时，计数器就会增加 1。从不会重置此计数器。

  可通过对 RSTicket 计数器进行采样来监控 **rs_ticket** 已到达的模块。RMS 或其它 Replication Server 监控工具使用这些计数器来产生 EXEC、DIST、RSI 和 DSI 心跳。

  还可以通过在主路径发送 **rs_ticket** 并检查 RSTicket 计数器来监控复制路径的运行状况。如果模块的 RSTicket 计数器没有增加，则说明这一段的复制路径中断。
- 您不得将 **rs_ticket** 标记为要进行复制。
- 只有在 Replication Server 为 15.0 或更高版本时，才能使用 **rs_ticket**。

另请参见

- rs_ticket_report （第 434 页）
- rs_ticket_history （第 625 页）

rs_zeroltm

将数据库的定位符值重置为零 (0)。在使用 Adaptive Server 命令 **dbcc settrunc** 禁用辅助截断点并截断日志之后，但在重新启动 Replication Server 之前使用该存储过程。

语法

```
rs_zeroltm data_server, database
```

参数

- **data_server** – 需要重置其定位符值的数据库所在的数据服务器。
- **database** – 需要重置其定位符值的数据库。

示例

- **示例 1** – 将 TOKYO_DS 数据服务器和 *pubs2* 数据库的定位符值重置为 0。

  ```
  rs_zeroltm TOKYO_DS, pubs2
  ```

用法

- 该命令用于启用了 RepAgent 的数据库。
- 在使用 **rs_zeroltm** 之前，使用 **dbcc settrunc** 禁用辅助截断点并截断日志。
- 复制数据库的定位符值由 Replication Server 维护，并存储在 *rs_locater* 表中。其值通常与存储在 Adaptive Server 中的辅助截断点的值相匹配。

  如果事务日志已满，可能需要使用 **dbcc settrunc** 命令来禁用辅助截断点并截断日志。**dbcc settrunc** 将重新设置辅助截断点，定位符值与辅助截断点随即不再匹配。

执行 **rs_zeroltm** 将值恢复到同步状态：使用 **rs_zeroltm** 将定位符值设置为零，即指示 Replication Server 从 Adaptive Server 获取新的辅助截断点，并将定位符设置为该值。

另请参见

- dbcc settrunc （第 447 页）

可执行程序

了解 Replication Server 可执行程序。这些可执行程序包括 Replication Server 和 **rs_subcmp** 过程。

repserver

Replication Server 可执行程序。

语法

```
{repserver | repsrvr} [-C config_file] [-i id_server]
[-S rs_name] [-I interfaces_file]
[-E errorlog_file] [-M] [-v] [-K keytab_file]
[-upgr] [-A erssd_release_dir] [-purgeq]
[-nodb {all|dbid_1[,dbid_2[,dbid_3[,…]]]}
[-e]
```

参数

- **-C *config_file*** – 指定 Replication Server 配置文件的名称和位置。**rs_init** 程序会创建一个配置文件，缺省情况下，该文件的名称是 *Rep_Server_name*`.cfg`，其中 *Rep_Server_name* 是 Replication Server 的名称。您可以使用 **-C** 标志指定该文件名。如果不使用 **-C** 标志，**repserver** 将在启动 Replication Server 的目录中查找名为 `config.rs` 的配置文件。
- **-i *id_server*** – 指定复制系统的 ID Server 的名称。ID Server 必须是第一个被启动的 Replication Server。只有当它处于运行和可访问状态时您才能启动新的 Replication Server。ID Server 的名称存储在配置文件中。可以使用 **-i** 选项指定一个不同的 ID Server。
- **-S *rs_name*** – 当前 Replication Server 使用的名称。如果启用了基于网络的安全性和统一登录，则指定主管用户的名称。
- **-I *interfaces_file*** – 指定定义 Replication Server 的 interfaces 文件的名称和位置。interfaces 文件中还必须包含与当前 Replication Server 通信的数据服务器和其它 Replication Server 的条目。位于复制节点的 interfaces 文件必须包含主 Replication Server 和主数据服务器的条目。如果不使用 **-I** 标志，Replication Server 将在 Sybase 版本目录中查找缺省的 interfaces 文件。

  有关您的平台上的该 interfaces 文件以及缺省 interfaces 文件名的详细信息，请参见适用于您的平台的 Replication Server 安装和配置指南。
- **-E *errorlog_file*** – 指定 Replication Server 错误日志文件的名称和位置，**repserver** 向该文件中写入错误消息。如果不使用 **-E** 标志，缺省的错误日志文件的名称是 `repserver.log`，位于启动了 Replication Server 的目录中。

- **-M** – 以独立模式启动 Replication Server，该模式用于启动恢复操作。有关在独立模式下运行 Replication Server 的详细信息，请参见《Replication Server 管理指南第二卷》。
- **-v** – 显示 Replication Server 的版本号。
- **-K *keytab_file*** – 指定包含登录到服务器中的用户的安全认证的 DCE keytab 文件的名称和位置。可以使用 DCE **dcecp** 实用程序来创建 keytab 文件。有关详细信息，请参见 DCE 文档。

  注意： **-K** *keytab_file* 选项仅用于 Windows 平台并且仅用于 DCE 网络安全性。
- **-upgr** – 指示 Replication Server 在升级下启动
- **-A *erssd_release_directory*** – 指定要升级的 ERSSD 的版本目录位置（如果 Replication Server 使用的是 ERSSD）。例如：
  - 在 UNIX 上 – `/sybase/REP-15_5/ASA11`
  - 在 Windows 上 – `c:\sybase\REP-15_5\ASA11`

  如果不包括 **-A** 选项，并且 Replication Server 配置文件包含相应的信息，则 Replication Server 从配置文件中获取版本目录位置。如果指定了 **-A** 选项，Replication Server 将忽略配置文件中的版本目录位置，因为您在 **repserver** 或 **repsrvr.exe** 命令中手动指定的内容将覆盖配置文件设置。
- **-purgeq** – 从入站队列中清除事务。如果要从 15.5 之前的 Replication Server 版本进行升级，则必须使用此选项。
- **-nodb all** – 从升级过程中排除所有用户数据库。
- **-nodb *dbid_1*[,*dbid_2*[,*dbid_3*[,...]]]** – 从升级过程中排除特定数据库。请使用逗号分隔多个数据库 ID，并且 ID 之间不包含空格。例如：

```
repserver -upgr . . .  -A . . .  -nodb 101,102,105
```

- **-e** – 在输入 **-upgr** 参数以进行升级时，记录 Replication Server 发送到数据服务器的 SQL 语句。如果未使用 **-e** 选项，则不会记录生成的 SQL 语句。不管有没有 **-e** 选项，在升级开始之前，升级进程都将使用 Replication Server 错误日志文件来记录 `rs_config` 表中存储的当前配置参数设置、升级进程中发生的任何错误以及为什么某些用户数据库未进行升级。如果需要降级，请参见错误日志文件以恢复先前的设置。

示例

- **示例 1** – 使用配置文件 *TOKYO_RS.cfg* 启动名为 TOKYO_RS 的 Replication Server。

```
repserver -STOKYO_RS -CTOKYO_RS.cfg
```

- **示例 2** – 使用配置文件 *SYDNEY_RS.cfg* 启动名为 SYDNEY_RS 的 Replication Server。*TOKYO_RS* 是用于复制系统的 ID Server。

```
repserver -SSYDNEY_RS -CSYDNEY_RS.cfg -iTOKYO_RS
```

- **示例 3** – 启动 Replication Server 并指定一个 interfaces 文件 `my_newinterfaces`，该文件覆盖缺省 interfaces 文件或 LDAP 目录服务。

```
repserver -STOKYO_RS _CTOKYO_RS.cfg
    -I$SYBASE/SYBASE_RS/my_newinterfaces
```

- **示例 4** – 使用 `/sybase/REP-15_5/ASA11` ERSSD 版本目录位置、RSSD *ny_rs.cfg* 配置文件、`my_newinterfaces` interfaces 文件和 `ny_rs_errorlog` 错误日志文件启动 NY_RS Replication Server 并进行升级：

```
repserver -upgr -SNY_RS -A/sybase/REP-15_5/ASA11 -Cny_rs.cfg -
Imy_newinterfaces -E ny_rs_errorlog
```

用法

- 可以使用 **repserver**（在 UNIX 中）或 **repsrvr.exe**（在 Windows 中）来启动 Replication Server 可执行程序。通常，可以通过执行由 **rs_init** 创建的运行文件来启动 Replication Server。为了方便起见，**repserver** 在两种平台上都引用该命令。
- **repserver** 可执行程序位于 Sybase 版本目录中的 `bin` 子目录中。有关详细信息，请参见适用于您的平台的 Replication Server 安装和配置指南。
- **repserver** 命令应当由“sybase”用户执行，以便 Replication Server 能够访问其磁盘分区。
- interfaces 文件中必须包含与当前 Replication Server 通信的其它 Replication Server 和数据服务器的定义。位于复制节点的 interfaces 文件必须包含主 Replication Server 和主数据服务器的条目。
- 如果口令是以加密形式存储的，则不能通过编辑 Replication Server 配置文件直接编辑它。要更改该文件中的加密口令，请使用 **rs_init** 程序。有关详细信息，请参见适用于您的平台的 Replication Server 安装和配置指南。
- 在配置 Replication Server 时，**rs_init** 自动将 *RSSD_primary_user* 和 *RSSD_maint_user* 指派给 *rs_systabgroup* 组。这使得这些用户能够修改系统表。您可以使用 Adaptive Server 系统过程 **sp_changegroup** 向该组添加其它用户登录名。有关详细信息，请参见《Adaptive Server Enterprise 系统管理指南》。
- 如果 RSSD 存在任何基于网络的安全性参数，则将 **use_security_services** 参数设置为“on”，并自动启动基于网络的安全性。
- 如果要升级 Replication Server，请使用 **-upgr** 参数。只有在使用 **-upgr** 时，您才能使用 **-A**、**-purgeq**、**-nodb** 和 **-e** 选项。请参见《Replication Server 配置指南》中的“使用 repserver 升级 RSSD 或 ERSSD 和用户数据库”。

表 48. Replication Server 配置文件参数

配置参数	说明
CONFIG_charset	用于编写 Replication Server 配置文件的字符集。仅当该字符集不同于 Replication Server 的字符集时，才需要使用此参数。它可以是任何与 Replication Server 的字符集兼容的字符集。
erssd_backup_dir	ERSSD 备份目录。

配置参数	说明
erssd_dbfile	ERSSD 数据库文件。
erssd_errorlog	ERSSD 错误日志。
erssd_logmirror	ERSSD 事务日志镜像文件。
erssd_ping_cmd	允许用户指定不同的命令来 ping ERSSD。仅用于调试。
erssd_port	网络监听器的 ERSSD 端口号。此端口号是从 interface 文件中获取的。
erssd_release_dir	允许用户指定不同的版本目录。仅用于调试。缺省值为 `$SYBASE/$SYBASE_REP/ASA11`。
erssd_ra_release_dir	允许用户为 ERSSD Replication Agent 指定不同的版本目录。仅用于调试。
erssd_ra_start_cmd	允许用户指定不同的命令来启动 ERSSD Replication Agent。仅用于调试。
erssd_start_cmd	允许用户指定不同的命令来启动 ERSSD。仅用于调试。
erssd_translog	ERSSD 事务日志文件。
ID_pw	ID Server 用户 (*ID_user*) 的口令。
ID_pw_enc	ID Server 用户 (*ID_user*) 的加密口令。
ID_server	作为复制系统的指定 ID Server 的 Replication Server 的名称。
ID_user	ID Server 上供其它 Replication Server 使用的登录名。
RS_charset	Replication Server 所使用的字符集。您可以指定 Sybase 支持的任何字符集。 在设置复制系统时，强烈建议给定 Replication Server 节点上的所有服务器都使用相同的字符集，虽然这不是必需的。我们还建议复制系统中的所有 Replication Server 都使用兼容的字符集。 有关详细信息，参见《Replication Server 设计指南》。
RS_language	Replication Server 将其消息显示到错误日志文件及其客户端中所使用的语言。您可以指定 Replication Server 已经进行本地化、与所选择的字符集兼容的任何语言。
RS_send_enc_pw	确保所有 Replication Server 客户端连接是使用加密口令建立的（与 RSSD 的第一次连接除外）。值为 on 和 off。 缺省值：off

配置参数	说明
RS_sortorder	Replication Server 使用的排序顺序。排序顺序控制表中的哪些行属于具有包含字符数据的 **where** 子句的预订。它还控制如何识别您所输入的标识符。 您可以指定 Sybase 支持的、与所选择的字符集兼容的任何排序顺序。复制系统中的所有排序顺序都应当是相同的。
RS_unicode_sort_order	Replication Server 使用的 Unicode 排序顺序。您可以指定 Sybase 支持的任何 Unicode 排序顺序。 缺省值：binary
RSSD_database	RSSD 的名称。
RSSD_embedded	指示是否嵌入 RSSD。
RSSD_ha_failover	指定是否允许 HA 故障切换。 缺省值：No。
RSSD_maint_pw	RSSD 维护用户的口令。
RSSD_maint_pw_enc	RSSD 维护用户的加密口令。
RSSD_maint_user	RSSD 维护用户的登录名。该登录名将被自动指派给 *rs_systabgroup* 组，该组的用户可以修改系统表。 您可以使用 Adaptive Server 系统过程 **sp_changegroup** 向该组添加其它用户登录名。有关详细信息，请参见《Adaptive Server Enterprise 系统管理指南》。
RSSD_msg_confidentiality	指定 Replication Server 是否发送和接收加密数据。如果设置为“required”，出站和入站的数据必须加密。如果设置为“not_required”，出站数据不加密，入站数据可以加密，也可以不加密。此选项尚未实现。 缺省值：not_required
RSSD_msg_integrity	指定是否检查数据有无被篡改。有效的条目为“required”和“not_required”。此选项尚未实现。 缺省值：not_required
RSSD_msg_origin_check	指定是否应检查数据的来源。有效的条目为“required”和“not_required”。此选项尚未实现。 缺省值：not_required
RSSD_msg_replay_detection	指定是否应当检查数据以确保它们尚未被读取或截取。有效的条目为“required”和“not_required”。此选项尚未实现。 缺省值：not_required

配置参数	说明
RSSD_msg_sequence_check	指定是否应当检查数据以确保顺序未被更改。有效的条目为“required”和“not_required”。此选项尚未实现。 缺省值：not_required
RSSD_mutual_auth	指定 RSSD 是否必须在 Replication Server 建立连接之前提供身份证明。有效的条目为“required”和“not_required”。此选项尚未实现。 缺省值：not_required
RSSD_primary_user	RSSD 主要用户的登录名。**rs_init** 在安装过程中自动将该用户指派给 *rs_systabgroup* 组。 可以使用 Adaptive Server 系统过程 **sp_changegroup** 向该组添加其它用户登录名。有关详细信息，请参见《Adaptive Server Enterprise 系统管理指南》。
RSSD_primary_pw	RSSD 主用户的口令。
RSSD_primary_pw_enc	RSSD 主要用户的加密口令。
RSSD_sec_mechanism	Replication Server 在启动时与 RSSD 进行初次联系时使用的安全性机制。此后，将从 `rs_config` 文件中读取与 RSSD 联系所用的网络安全性信息。此选项尚未实现。
RSSD_server	具有 RSSD 的 Adaptive Server 的名称。
RS_ssl_identity	SSL 标识文件。
RS_ssl_pw	SSL 私有密钥的口令
RS_ssl_pw_enc	SSL 私有密钥的加密口令。
RSSD_unified_login	指定 Replication Server 在启动时是否试图通过认证连接到 RSSD。此后，将从 `rs_config` 文件中读取与 RSSD 联系所用的网络安全性信息。有效的条目为“required”和“not_required”。此选项尚未实现。 缺省值：not_required

trace	打开 Replication Server 跟踪。可以使用此参数的多个实例来设置不同的可用跟踪。不允许包括空格。例如：
	`trace=DSI,DSI_BUF_DUMP`
	`trace=DIST,DIST_TRACE_COMMANDS`

trace_file	指示 Replication Server 日志文件的名称。

rs_subcmp

一个可执行程序，该程序会对复制表的数据和该表的主版本进行比较。**rs_subcmp** 还执行复制表和主表之间以及复制数据库和主数据库之间的模式比较。这些功能有助于查找（也可以进行调和）丢失的、孤立的和不一致的行和模式。在 UNIX 系统上，该程序称为 **rs_subcmp**。在 Windows 系统上，该程序称为 **subcmp**。

rs_subcmp 程序位于 Sybase 版本目录的 `bin` 子目录中。有关详细信息，请参见适用于您的平台的 Replication Server 安装和配置指南。

要使 **rs_subcmp** 正常工作，必须设置 *SYBASE* 环境变量和库路径环境变量。如果将 **rs_subcmp** 用于模式比较，请确保 **rs_subcmp** 可以找到 `ddlgen` 可执行文件并确保 `ddlgen` 可成功在您的 Replication Server 环境中运行。有关说明，请参见“用法”部分。

rs_subcmp 只用来调和 Sybase 数据库。

语法

```
rs_subcmp [-R | -r] [-v] [-V] [-z[1 | 2] [-g] [-h]]
 [-f config_file] [-F]
 -S primary_ds [-D primary_db]
 -s replicate_ds [-d replicate_db]
 -t table_name [-T primary_table_name]
 -c select_command [-C primary_select_command]
 -u user [-U primary_user]
 [-p passwd] [-P primary_passwd]
 [-B primary_init_batch]
 [-b replicate_init_batch]
 [-n num_iterations] [-w wait_interval]
 [-e float_precision] [-E real_precision]
 [-k primary_key_column [-k primary_key_column]...]
 [-i identity_column]
 [-l text_image_column_name
 [-l text_image_column_name]...]
 [-L text_image_length_in_kilobytes]
 [-N text_image_column_name
 [-N text_image_column_name]...]
 [-Z language]
 [-o sort_order]
 [-O sort_order]
 [-J rs_subcmp_charset]
 [-j rep_charset]
 [-a replicate_column_name primary_column_name
 [-a replicate_column_name primary_column_name]...]
 [-q unicode_sort_order]
 [-Q unicode_sort_order]
 [-x schema_flag]
 [-X filter_flag]
 [-I interface_file]
 [-H normalization_option]
```

参数

- **-R** – 调和复制数据与主数据，在主数据库中对数据的不一致性作最终检验。**rs_subcmp** 在复制数据库中插入、删除和更新行，以使复制数据与主数据相匹配。
- **-r** – 调和复制数据与主数据，但是不像 **-R** 那样在主数据库对数据的不一致性作最终校验。**rs_subcmp** 在复制数据库插入、删除和更新行，以使复制数据与主数据相匹配。
- **-v** – 显示版本信息。
- **-V** – 在显示器（标准输出）上（可见）显示比较结果。如果不使用 **-V** 标志，**rs_subcmp** 不会报告行之间的差异。不显示 *text*、*unitext* 或 *image* 数据的值。相反，**rs_subcmp** 报告不一致性问题是存在于 *text*、*unitext* 或 *image* 列中，还是存在于其它数据类型的列中。
- **-z** – 启用跟踪。缺省值 **-z1** 提供基本跟踪信息，例如列标题的比较。**-z1** 还输出有关数值精度差异的信息。**-z2** 提供有关所有行和命令比较的跟踪信息。
- **-f config_file** – 指定 **rs_subcmp** 的配置文件的名称。
- **-F** – 显示要用于 *config_file* 的格式（语法）。配置文件必须使用用 **-F** 选项显示的语法，并且必须包含所有必需的语法参数。
- **-S primary_ds** – 包含预订的主数据的数据服务器的名称。
- **-D primary_db** – 存储预订的主数据的数据库的名称。
- **-s replicate_ds** – 包含数据的复制副本的数据服务器的名称。
- **-d replicate_db** – 包含数据的复制副本的数据库的名称。
- **-t table_name** – 主数据库和复制数据库中包含要比较的数据的表的名称。如果在这两个数据库中该名称是不同的，请使用 **-T** 选项指定主数据库中该表的名称。您可以在此处包括表所有者的名称信息。
- **-T primary_table_name** – 主数据库中表的名称。当该表名在主数据库和复制数据库中不同时，应当使用此选项。您可以在此处包括表所有者的名称信息。
- **-c select_command** – 一个 **select** 命令，该命令从数据的主副本和复制副本中检索预订的数据。您可以使用 **-C** 为主数据指定一个不同的命令。**select** 命令必须根据主键对行进行排序。

  可以在 **select** 命令中包括数据类型为 *text*、*unitext* 或 *image* 的列，但应满足以下要求：

  - 数据类型为 *text*、*unitext* 或 *image* 的列不能作为主键列。
  - 必须将数据类型为 *text*、*unitext* 或 *image* 的列放在 **select** 列表的结尾。
  - 缺省情况下，复制表不允许 *text* 或 *image* 列有空值。必须在 **rs_subcmp** 可执行程序中包括 **-N** 标志，以指示复制表的 *text*、*unitext* 或 *image* 列中允许有 null 值。
- **-C primary_select_command** – 一个 **select** 命令，该命令从数据的主副本检索预订的数据。当主数据库和复制数据库需要使用不同的 **select** 命令时，请使用此选项和 **-c**。**select** 命令必须根据主键对行进行排序。
- **-u user** – 用于登录主数据服务器和复制数据服务器的登录名。如果需要不同的登录名，请使用 **-U** 选项指定一个不同的主数据服务器登录名。

- **-U primary_user** – 用于登录主数据服务器的登录名。当对主数据服务器和复制数据服务器要求不同的登录名时，请使用此选项和 **-u** 选项。
- **-p passwd** – 与 *user* 登录名和 *primary_user* 登录名（如果已提供）一起使用的口令。如果忽略此选项，**rs_subcmp** 将使用空口令。如果要为 *primary_user* 登录名指定不同的口令，请使用 **-P** 选项来指定。
- **-P primary_passwd** – 与 *primary_user* 登录名一起使用的口令。
- **-B primary_init_batch** – 最初连接到主数据库时要执行的一批命令。该批命令可用于任何目的，如设置隔离级别。该批命令在 **rs_subcmp** 登录主数据库后运行。
- **-b replicate_init_batch** – 最初连接到复制数据库时要执行的一批命令。该批命令可用于任何目的，如在热备份应用程序中运行 **rs_subcmp** 时关闭触发器，或者设置隔离级别。该批命令在 **rs_subcmp** 登录复制数据库后运行。
- **-n num_iterations** – **rs_subcmp** 检查它所找到的不一致行的次数。缺省值为 10 次迭代。第一次迭代可能找到由于复制过程中的正常时间滞后而导致的许多不一致。其它迭代允许 **rs_subcmp** 从通过正常复制活动纠正的不一致行中区分出真正的不一致。
- **-w wait_interval** – **rs_subcmp** 在另一次迭代开始之前等待的秒数。缺省值为 5 秒。
- **-e float_precision** – 设置指数记数法中小数的位数，该数目应当与浮点值一致。缺省情况下，该值设置为平台所支持的最大精度。
- **-E real_precision** – 设置指数记数法中小数的位数，该数目应当与实值一致。缺省情况下，该值设置为平台所支持的最大精度。
- **-k primary_key_column** – 一个列名，它是表的主键的一部分。主键必须是唯一的，并且不能是 *text*、*unitext* 或 *image* 列。主键中的每一列使用 **-k** 选项。如果主列和复制列的名称不同，此处指定的名称为复制列的名称。
- **-i identity_column** – 复制表中 *xidentity* 列的名称。
- **-l text_image_column_name** – 关闭对复制 *text*、*unitext* 或 *image* 列的更新的记录。缺省情况下，记录对 *text*、*unitext* 或 *image* 列的更新。
- **-L text_image_length** – 设置数据服务器为 *text*、*unitext* 或 *image* 列返回的最长值。缺省值为 2048K。
- **-N text_image_column_name** – 指示复制表的 *text*、*unitext* 或 *image* 列中允许有 null 值。缺省情况下，复制表不允许 *text*、*unitext* 或 *image* 列中有 null 值。
- **-Z language** – **rs_subcmp** 生成错误和信息性消息时使用的语言的名称。如果未指定，将使用您的平台的“缺省”区域设置条目中指定的语言。
- **-o sort_order** – 复制系统中使用的排序顺序的名称。**rs_subcmp** 使用此信息来比较主键列。
- **-O sort_order** – 复制系统中使用的排序顺序的名称。**rs_subcmp** 使用此信息来比较所有的列。
- **-J rs_subcmp_charset** – **rs_subcmp** 错误和信息性消息以及在所有配置参数和命令行选项中使用的字符集的名称。如果不指定 *rs_subcmp_charset*，它将被设置为您的平台的“缺省”区域设置条目中指定的字符集。

- **-j rep_charset** – 复制数据服务器所使用的字符集的名称。**rs_subcmp** 程序在比较和调和表的复制版本和主版本时使用此字符集。如果不指定 *rep_charset*，它将被设置为 *rs_subcmp_charset* 字符集。
- **-a replicate_column_name primary_column_name** – 指定与复制列相关联的主列名称。如果复制列的名称与主列的名称不同，请使用此选项。

  注意： 使用 **-a** 选项时，复制列名称必须出现在相关联的主列名称之前。
- **-q unicode_sort_order** – 指定 **rs_subcmp** 用于比较 Unicode 主键列的 Unicode 排序顺序。
- **-Q unicode_sort_order** – 指定 **rs_subcmp** 用于比较所有 Unicode 列的 Unicode 排序顺序。
- **-x schema_flag** – 指定 **rs_subcmp** 比较类型。可能的 *schema_flag* 值有：
  - 0 – 数据比较。该值为缺省值。
  - 1 – 两个数据库之间的数据库模式比较。
  - 2 – 两个表之间的表模式比较。
- **-X filter** – 指定在比较中包括或排除的模式类型和子类型。如果该值以“+”开头，则只会选择模式类型用于比较，而忽略子模式类型。否则，既不选择模式类型也不选择子模式类型，因而也无法将这两种类型用于比较。有关 rs_subcmp 支持的模式类型和模式子类型的列表，请参见“rs_subcmp 支持的模式类型”表和“rs_subcmp 支持的模式子类型”表。
- **-I interface_file** – 指定 interface 文件位置。有关 interface 文件的详细信息，请参见适用于您的平台的 Replication Server 配置指南。
- **-g** – 为不一致的数据创建调和文件。
- **-h** – 执行快速比较。
- **-H normalization_option** – 指示在执行快速比较时如何规范化数据。有关 **rs_subcmp** 支持的规范化选项的列表，请参见“rs_subcmp 支持的规范化选项”表。

示例

- **示例 1** – 使用名为 `titleauthor.cfg` 的配置文件启动 **rs_subcmp**。

```
rs_subcmp -ftitleauthor.cfg
```

配置文件包含下列内容：

```
# titleauthor.cfg - Reconcile
# SYDNEY_DS.pubs2.dbo.titleauthor with
# TOKYO_DS.pubs2.dbo.titleauthor.
#
PDS        = TOKYO_DS
RDS        = SYDNEY_DS
PDB        = pubs2
RDB        = pubs2
PTABLE     = titleauthor
RTABLE     = titleauthor
PSELECT    = select au_id, title_id, au_ord,\ royaltyper
```

```
    from titleauthor order by au_id,\ title_id
RSELECT   = select au_id, title_id, au_ord,\ royaltyper
    from titleauthor order by au_id,\ title_id
PUSER     = repuser
RUSER     = repuser
PPWD      = piglet
RPWD      = piglet
KEY       = au_id
KEY       = title_id
RECONCILE = Y
VISUAL    = Y
NUM_TRIES = 3
WAIT      = 10
```

rs_subcmp 比较名为 *titleauthor* 的主表和复制表，并输出以下内容；

```
$SYBASE/bin/rs_subcmp -f ttl_au.cmp
INCONSISTENT ROWS:

_________Replicate row________
au_id        title_id  au_ord  royaltyper
-----------------------------------------
672-71-3249  TC7777    1       40

__________Primary row_________
au_id        title_id  au_ord  royaltyper
-----------------------------------------
672-71-3249  TC7777    1       50
```

- **示例 2** – 使用名为 `subcmp.cfg` 的配置文件启动 **rs_subcmp**。命令行标志将覆盖配置文件设置，并执行最终检验来调和 *authors* 表的主版本与复制版本之间的差异。

  ```
  rs_subcmp -R -fsubcmp.cfg -STOKYO_DS -Dpubs2 \
   -sSYDNEY_DS -dpubs2 -tauthors
  ```

  主数据服务器和数据库分别为 TOKYO_DS 和 *pubs2*。复制数据服务器和数据库分别为 SYDNEY_DS 和 *pubs2*。

- **示例 3** – 使用名为 `config.cfg` 的配置文件在两个不同服务器 PASE 和 R2ASE (每个服务器上都有一个名为 `pubs2` 的数据库) 上比较 `authors` 表的模式：

  ```
  rs_subcmp -f config.cfg
  ```

  该配置文件包含以下内容：

  ```
  PDS = PASE
  RDS = R2ASE
  PDB = pubs2
  PTABLE = authors
  RTABLE = authors
  PUSER = sa
  RUSER = sa
  PPWD =
  RPWD =
  SCHEMAFLAG = 1
  ```

- **示例 4** – 对没有配置文件的两个数据库中的模式进行比较：

```
rs_subcmp -Spds -srds -Dpdb -drdb -Usa -Psa_pwd -usa
          -psa_pwd -x1
```

- **示例 5** – 比较两个数据库的模式，但不包括索引、触发器和数据类型：

```
rs_subcmp -Spds -srds -Dpdb -drdb -Usa -Psa_pwd -usa
          -psa_pwd -x1 -XitD
```

- **示例 6** – 比较所有表模式和用户模式：

```
rs_subcmp -Spds -srds -Dpdb -drdb -Usa -Psa_pwd -usa
          -psa_pwd -x1 -X+TU
```

用法

- 在未发生主数据库更改时运行 **rs_subcmp**。
- 要使 **rs_subcmp** 正常工作，必须设置 *SYBASE* 环境变量和库路径环境变量。
  将 *SYBASE* 环境变量设置为 Sybase 版本目录。
  将库路径变量设置为 `$SYBASE/$SYBASE_OCS/lib` (UNIX) 或 `%SYBASE%\%SYBASE_OCS%\lib` (Windows)：
  - 对于 Solaris 和 Linux，库路径变量为 LD_LIBRARY_PATH。
  - 对于 HP，库路径变量为 SHLIB_PATH。
  - 对于 RS6000，库路径变量为 LIBPATH。
  - 对于 Windows，库路径变量为 PATH。
- 要使 **rs_subcmp** 模式比较正常工作，请设置 DDLGENLOC 和 SYBROOT 环境变量。
  rs_subcmp 必须能够找到并成功运行 `ddlgen` 可执行文件，以使模式比较正常进行。如果未设置 *DDLGENLOC*，则 **rs_subcmp** 会在其缺省位置 (`%SYBASE%\ASEP\bin\ddlgen`) 查找 `ddlgen`。若要确保 `ddlgen` 成功运行，必须正确设置 `ddlgen` 使用的环境变量。

  注意： 如果 `ddlgen` 不在其缺省位置，则必须将 *DDLGENLOC* 设置成完整路径，例如 `%SYBASE%\ASEP\bin\ddlgen`。

  SYBROOT 环境变量也必须设置为 SYBASE 环境变量。
- 下列要求适用于 **rs_subcmp**：
  - 如果提供了配置文件，并且还使用命令行选项，命令行值将覆盖配置文件中的值。
  - 小写的选项 **-d**、**-c**、**-u**、**-p** 和 **-t** 为主数据和复制数据提供值。您可以使用大写选项覆盖主数据的值。
  - 唯一必需的大写选项为 **-S**。
  - 使用 **-k** 指定的主键必须是唯一的。如果没有使用 **-k** 选项指定任何主键列，则将所有列视为主键的一部分。
  - 在 **-L** 中使用正整数为 text 和 image 列的字节长度指定新值，以覆盖缺省值 26K：

```
-L = <new_value>
```

例如，如果希望 text 和 image 列为 65,536 个字节，请输入：

```
-L = <64>
```

- 这些选项可用于指定非缺省表所有者或另一个主复制表或列名：
  - 如果使用选项 **-t**、**-T**、**-c** 和 **-C**，可以包括表所有者信息（例如 **ling.authors**）。
  - 为 **-c** 选项指定的所有者、表和列名应属于复制表内容。
  - 为 **-C** 选项指定的所有者、表和列名应属于主表内容。
  - 为 **-k** 选项指定的列名是复制表的列名。
- 在执行每个模式比较后，**rs_subcmp** 将创建一个报告文件。此报告文件详细说明了两个表或两个数据库之间的比较结果。此报告文件的名称为 `reportPROCID.txt`。如果存在不一致性问题，**rs_subcmp** 将创建一个名为 `reconcilePROCID.sql` 的调和脚本。报告文件和调和脚本保存在从中执行 **rs_subcmp** 的相同目录中。
- 调和文件的 SQL 语句不能包含 *text*、*unitext* 或 *image*。
- 如果指定了 **-g** 选项，**rs_subcmp** 将创建一个调和文件。该文件的名称为 `reconcile_file_PROCID.sql`，它位于当前工作目录中。

返回代码

rs_subcmp 可以返回以下返回代码：

表 49. rs_subcmp 返回代码

返回代码	含义
0	复制表和主表相同。
1	执行 **rs_subcmp** 时发生错误。
2	复制表和主表不同。

配置文件

您可以创建一个包含 **rs_subcmp** 参数的文件，并使用 **-f** 标志在命令行上指定它。配置文件中的每一行都包含一个参数名称、一个等号 (=) 和一个值。

表 50. rs_subcmp 配置文件参数 （第 565 页）列出了 **rs_subcmp** 配置文件中可以使用的参数以及各个参数的相应命令行选项。

表 50. rs_subcmp 配置文件参数

配置参数	命令行选项	值
PDS	**-S**	主数据服务器名
RDS	**-s**	复制数据服务器名

配置参数	命令行选项	值
PDB	**-D**	主数据库名
RDB	**-d**	复制数据库名
PTABLE	**-T**	主表名
RTABLE	**-t**	复制表名
PUSER	**-U**	主要用户名
RUSER	**-u**	复制用户名
PPWD	**-P**	主口令
RPWD	**-p**	复制口令
KEY	**-k**	复制表中的主键元素
PINITBATCH	**-B**	主数据库连接初始化批处理。如果换行符前有一个“\”（反斜杠），则可以跨多个行。每行最多允许 1024 个字符，总共允许 64K 个字符。
RINITBATCH	**-b**	复制主数据库连接初始化批处理。如果换行符前有一个“\”（反斜杠），则可以跨多个行。每行最多允许 1024 个字符，总共允许 64K 个字符。
PSELECT	**-C**	主 **select** 命令。如果换行符前有一个“\”（反斜杠），则可以跨多个行。每行最多允许 1024 个字符，总共允许 64K 个字符。
RSELECT	**-c**	复制 **select** 命令。如果换行符前有一个“\”（反斜杠），则可以跨多个行。每行最多允许 1024 个字符，总共允许 64K 个字符。
RECONCILE	**-r**	调和差异（Y 或 N）
RECONCILE_CHECK	**-R**	调和与主检验的差异（Y 或 N）
TRACE	**-z**	启动具有可选级别（可选整数）的跟踪
FPRECISION	**-e**	期望的浮点精度（整数 — 缺省值与平台相关）
RPRECISION	**-E**	期望的实际精度（整数 — 缺省值与平台相关）
WAIT	**-w**	比较之间的秒数（整数 — 缺省值为 5 秒）
NUM_TRIES	**-n**	比较的次数（整数 — 缺省值为 10 次迭代）
VISUAL	**-V**	显示结果（Y 或 N）
IDENTITY	**-i**	复制表中的 *identity* 列名

配置参数	命令行选项	值
TXT_IMG_LEN	**-L**	数据服务器为 *text*、*unitext* 或 *image* 列返回的最长值（以千字节为单位）。
NO_LOG	**-l**	不会记录对此复制 *text*、*unitext* 或 *image* 列的更新
NULLABLE	**-N**	复制表中的 *text*、*unitext* 或 *image* 列接受 null 值。
LANGUAGE	**-Z**	**rs_subcmp** 错误和信息性消息的语言
SORT_ORDER	**-o**	使用指定的排序顺序比较主键列
SORT_ORDER_ALL_COLS	**-O**	使用指定的排序顺序比较所有的列
SCHARSET	**-j**	**rs_subcmp** 的字符集
RCHARSET	**-J**	复制数据服务器的字符集
REP_PRI_COLNAME	**-a**	复制列名主列名对
UNICODE_SORT_ORDER	**-q**	**rs_subcmp** 用于比较 Unicode 主键列的 Unicode 排序顺序
UNICODE_SORT_ORDER_ALL_COLS	**-Q**	**rs_subcmp** 用于比较所有 Unicode 列的 Unicode 排序顺序
SCHEMAFLAG	**-x**	**rs_subcmp** 比较类型。
FILTER	**-X**	用于指示模式比较中包括或排除的模式类型和模式子类型的过滤器。有关 **rs_subcmp** 支持的模式类型和模式子类型的列表，请参见表 51. rs_subcmp 支持的模式类型 （第 571 页） 和表 52. rs_subcmp 支持的模式子类型 （第 571 页）。
IFILE	**-I**	interface 文件位置。
RECONCILE_FILE	**-g**	指示是否创建调和文件。 值: • Y – 创建调和文件。 • N – 不创建调和文件。 缺省值：N
FASTCMP	-h	指示是否执行快速比较。 值: • Y – 使用压缩数据执行快速比较。 • N – 执行正常比较。 缺省值：N

配置参数	命令行选项	值
HASH_OPTION	**-H**	指示用于快速比较的规范化选项。如果配置文件中不包含此参数，则 **rs_subcmp** 使用本地字节顺序和字符集规范化数据。有关 **rs_subcmp** 支持的规范化选项的列表，请参见“rs_subcmp 支持的规范化选项”表。

select 命令的要求

- **-c** (RSELECT) 和 **-C** (PSELECT) 指定的 **select** 命令必须从主数据库和复制数据库返回具有相同名称和数据类型的列。
- 主键上必须有聚簇索引，或者，**select** 命令中必须有一个 **order by** 子句。**select** 命令必须根据主键对行进行排序。如果 **rs_subcmp** 不是以正确顺序接收行，它可能会删除复制表中的行。
- 请不要使用 **-c** 或 **-C** 选项选择 *rs_address* 数据类型。如果复制表包含使用 *rs_address* 数据类型的列，这些列的主版本和复制版本可能不完全相同。Replication Server 将过滤出对这些列的更新，以避免对它们进行不必要的复制。

rs_subcmp 的工作方式

- **rs_subcmp** 登录主数据库和复制数据库，并执行提供的 **select** 命令。它根据每列的名称和数据类型检验命令是否返回相同的列。如果返回的列匹配，**rs_subcmp** 将比较主行和复制行，并创建下面的列表：
  - 丢失行 – 主数据库中的行，而不是复制数据库中的行
  - 孤立行 – 复制数据库中的行，而不是主数据库中的行
  - 不一致的行 – 具有匹配主键的复制数据库和主数据库中的行，但其他列则不同
- 编译这三个列表后，**rs_subcmp** 按照指定的次数进行迭代，检查以下内容：
  - 丢失的行是否出现在复制数据库中
  - 孤立的行是否从复制数据库中消失
  - 不一致行是否匹配
  - 新复制行的值是否匹配前一次迭代中的主行的值
- 完成指定次数的迭代后，如果指定了 **-V** 选项，将把这三个列表的内容显示到标准输出中。

调和不一致

- 如果指定了 **-R** 或 **-r** 选项，**rs_subcmp** 将调和丢失的、孤立的和不一致的行。
- 如果指定了 **-r** 选项，**rs_subcmp** 将调和主副本和复制副本。它将传递最终的列表并修改复制表，如下所示：
  - 插入留在丢失行列表中的行
  - 删除留在孤立行列表中的行
  - 更新不一致行，以匹配主行

- 如果指定了 **-R** 选项，**rs_subcmp** 将使用与 **-r** 选项相同的方式将复制表调和为主版本。但是，在插入丢失的行或删除孤立的行之前，它将登录主数据库并对行执行 **select**，以检验以下内容：
  - 行是否仍然存在（在复制表中有丢失行的情况下），或者
  - 行是否不存在（在复制表中有孤立行的情况下）。

调和 IDENTITY 列

- 如果某一行的 *identity* 列中的值不一致，**rs_subcmp** 将通过在插入主数据库中的该行之前删除复制数据库中的该行来调和它们。

调和 text、unitext 或 image 数据类型

- 与其它数据类型不同，不一致的 *text*、*unitext* 或 *image* 值不会存储到列表中。要调和包含 *text* 或 *image* 值的丢失行或不一致行，**rs_subcmp** 将再次登录到主数据库并重新执行 **select** 语句。如果找到了不一致的或丢失的行，**rs_subcmp** 将通过更新或插入该行来修改复制表。但是，如果主表中未找到不一致或丢失的行，**rs_subcmp** 将执行下列操作：
  - 如果是不一致行，**rs_subcmp** 将从复制表中删除该行
  - 如果是丢失的行，**rs_subcmp** 将不执行任何操作
- 如果将 Adaptive Server 选项 **set textsize** 用作 **select** 语句的一部分，则可以限制比较的文本量。例如，以下示例显示将 textsize 设置为 10 的效果。第一个 **select** 语句返回 30 个 *text* 字符：

```
set textsize 30 select * from zetext

a           b            c
--------    ---------    -----------
abba        apples       odd one here
beta        banana       rotten
caro        celery       not carrots
```

第二个 select 语句将文本的 *size* 设置为 10：

```
1> set textsize 10 select * from zetext
2> go
a           b           c

--------    ---------   ------------
abba        apples      odd one
beta        banana      rotten
caro        celery      not carrots
-----------------------------------------------------

(3 rows affected)
```

在国际环境中使用 rs_subcmp

- **rs_subcmp** 通过 **-Z***language*、**-o** *sort_order*、**-O** *sort_order*、**-q** *unicode_sort_order*、**-Q** *unicode_sort_order*、**-J** *rs_subcmp_charset* 和 **-j** *rep_charset* 选项提供对国际环境的支持。

- **rs_subcmp** 在比较并调和表的复制版本和主版本时执行字符集转换。转换方法类似于 Replication Server 字符集的转换方法，因此会得到类似的结果。
  例如，如果主数据服务器和复制数据服务器的字符集不兼容，将不会进行任何转换。如果字符集不兼容，但是主数据服务器的字符集中的单个字符在复制服务器的字符集中没有相应的表示形式，就用“?”替换该字符，并继续处理。
- **rs_subcmp** 在所有涉及用户数据的操作中均使用复制数据服务器的字符集。要指定复制数据服务器的字符集，请使用 **-j** 命令行选项或 *RCHARSET* 配置文件参数。

  注意： **rs_subcmp** 没有用于主数据服务器的字符集的参数，原因是所有的数据操作都是使用复制数据服务器的字符集完成的。该程序依赖于主数据服务器将所有的字符数据转换为复制数据服务器的字符集。这与 Replication Server 在预订实现期间的工作方式等效。
- 如果 **rs_subcmp** 的字符集不同于复制数据服务器的字符集，您也可以为它指定一个字符集。为此，请使用 **-J** 命令行选项或 *SCHARSET* 配置文件参数。指定字符集时，**rs_subcmp** 将把它的字符串类型的配置参数从 **rs_subcmp** 字符集转换为复制数据服务器的字符集。

字符集和排序顺序要求

- 下面的要求适用于指定 **rs_subcmp** 中的字符集和排序顺序：
  - 对象名（包括服务器、数据库、表和列名）中的所有字符都必须与 *rs_subcmp_charset* 和 *rep_charset* 字符集兼容；否则，**rs_subcmp** 将无法执行。
  - 如果复制数据服务器和主数据服务器的字符集不同，必须在主数据服务器安装复制数据服务器的字符集。这样，主数据服务器才能进行字符集转换。
  - 如果复制数据服务器和主数据服务器使用不同的排序顺序，并且 **select** 语句的 **where** 子句中包括 *character* 或 *text* 数据类型，结果可能会出现混淆。要避免混淆，第一次运行 **rs_subcmp** 时不使用 **-r** 或 **-R**（调和）选项，使用 **-V**（可见）选项，您可以看到对数据的潜在影响。

使用排序顺序

- 可以用两种方式指定非 Unicode 排序顺序：使用 **-o** 选项或使用 **-O** 选项。
- 如果指定了 **-o** 选项，**rs_subcmp** 将：
  1. 对主键列执行简单的二进制比较。
  2. 如果主键相匹配，**rs_subcmp** 将对其余的列执行二进制比较。如果这些列不匹配，将报告一个不一致的行。
  3. 如果主键列不匹配，**rs_subcmp** 将使用指定的排序顺序比较它们。
     - 如果主键列不匹配，将把该行报告为丢失行或孤立行。
     - 如果使用排序顺序时主键列的测试结果相同，将把该行报告为不一致。
- 如果指定了 **-O** 选项，**rs_subcmp** 将：
  - 使用指定的排序顺序对所有 *char*、*varchar* 和 *text* 类型的列执行列比较。
  - 不执行二进制比较。

- 如果未指定排序顺序，**rs_subcmp** 将对主行和复制行的每一列执行简单的二进制比较。

使用 Unicode 排序顺序

- 可以用两种方式指定 Unicode 排序顺序：使用 **-q** 选项或使用 **-Q** 选项。
- 如果指定了 **-q** 选项，**rs_subcmp** 将：
  1. 对 Unicode 主键列执行简单的二进制比较。
  2. 如果主键相匹配，**rs_subcmp** 将对其余的列执行二进制比较。如果这些列不匹配，将报告一个不一致的行。
  3. 如果主键列不匹配，**rs_subcmp** 将使用指定的排序顺序比较它们。
     - 如果 Unicode 主键列不匹配，将把该行报告为丢失行或孤立行。
     - 如果使用排序顺序时主键列的测试结果相同，将把该行报告为不一致。
- 如果指定了 **-Q** 选项，**rs_subcmp** 将：
  - 使用指定的排序顺序对所有的 Unicode 列执行列比较。
  - 不执行二进制比较。
- 如果未指定排序顺序，**rs_subcmp** 将对主行和复制行的每个 Unicode 列执行简单的二进制比较。

表 51. rs_subcmp 支持的模式类型

类型	说明
A	数据库中的所有别名。
D	数据库中的所有缺省值。
E	数据库中所有用户定义的数据类型。
G	数据库中的所有组。
R	数据库中的所有规则。
T	数据库中的所有用户表。包括表元素，如索引、键、约束和触发器。
U	数据库中的所有用户。
V	数据库中的所有视图。
P	数据库中的所有过程。

表 52. rs_subcmp 支持的模式子类型

类型	说明
c	约束
d	绑定缺省值
f	外键

类型	说明
g	授予
i	Index
m	过程模式
p	主键
r	绑定规则
t	触发器

表 53. rs_subcmp 支持的规范化选项

规范化选项	说明
lsb	将所有与字节顺序有关的数据规范化为最低有效位在前 (little-endian) 个字节顺序。
msb	将所有与字节顺序有关的数据规范化为最高有效位在前 (big-endian) 个字节顺序。
unicode	将字符数据规范化为 Unicode (UTF-16)。
unicode_lsb	将 lsb 与 Unicode 一起规范化以保持平台独立性。
unicode_msb	将 msb 与 Unicode 一起规范化以保持平台独立性。

Replication Server 系统表

了解 Replication Server 系统数据库 (RSSD) 或嵌入式 RSSD (ERSSD) 中的系统表。系统表存储在专用数据库（Adaptive Server for RSSD 或 SQL Anywhere ERSSD）中。

只有具有 **sa** 权限的用户或者 *rs_systabgroup* 组的成员才能访问系统表。系统表是由 RCL 命令维护的，不能直接修改这些表。若要更改位于 *rs_config* 表中的服务器值，请使用 **configure replication server** 命令。

有关 *rs_systabgroup* 组的详细信息，请参见 repserver。有关 **configure replication server** 的信息，请参见 **configure replication server**。

系统表包括用户定义的数据类型 *rs_id*，该数据类型被定义为 *binary(8)*。该数据类型用于那些存放对象名的列。有关标识符的详细信息，请参见“标识符”。

本章介绍了 *rs_lastcommit* 和 *rs_threads* 系统表，但这两个表是在各个用户数据库中（而非在 RSSD 或 ERSSD 中）创建和存储的。

rs_articles

存储有关此 Replication Server 的已知项目的信息。

列	数据类型	说明
articlename	*varchar(255)*	项目名
articleid	*rs_id*	唯一项目 ID
type	*char(1)*	• T – 表 • P – 过程
primaryname	*varchar(255)*	主表或过程名
primaryowner	*varchar(30)*	主表所有者的名称
objid	*rs_id*	相应的复制定义的 ID
pubid	*rs_id*	该项目所属发布的 ID
requestdate	*datetime*	在发布中添加该项目的日期和时间
minvers	*int*	支持该项目所需的最低 Replication Server 版本

索引

- （*articlename*、*pubid*）的唯一聚簇索引

- (*articleid*) 的唯一索引

rs_asyncfuncs

存储有关用户定义的函数（比对 Replication Server 中的复制定义）的信息。同样的信息也存储在 `rs_objfunctions` 中。

列	数据类型	说明
prsid	`int`	以该函数为主函数的节点
funcname	`varchar(255)`	函数名
funcid	`rs_id`	函数 ID
objid	`rs_id`	应用函数的对象。
conflicting	`tinyint`	如果函数存在冲突则为 1，否则为 0
userdefined	`bit`	如果是用户定义的函数则为 1，否则为 0
rowtype	`tinyint`	如果该行是复制行，则为 1，否则为 0

索引

- (`funcname`) 的聚簇索引
- （`objid` 和 `funcname`）的唯一索引
- (`funcid`) 的唯一索引

rs_classes

存储函数字符串类和错误类的名称。

列	数据类型	说明
classname	*varchar(30)*	类名
classid	*rs_id*	该类的 ID
classtype	*char(1)*	值为以下一项： • R – Replication Server 错误类 • F – 函数字符串类 • E – 数据服务器错误类 • D – 数据类型类

列	数据类型	说明
prsid	*int*	将该类作为主类的节点的 ID
parent_classid	*rs_id*	如果该类是派生类，则为其父类的 ID 如果该类是基类，则为 0；缺省值为 0
attributes	*int*	0x01 – 缺省类 对于 *rs_default_function_class* 和 *rs_db2_function_class*，缺省值为 1。否则，缺省值为 0。

索引

- （*classname*、*classtype*）的唯一聚簇索引
- (*classid*) 的唯一索引

rs_clsfunctions

存储有关类范围函数的信息。

列	数据类型	说明
prsid	*int*	以该函数为主函数的节点
funcname	*varchar(255)*	函数名
funcid	*rs_id*	函数 ID
objid	*rs_id*	0x00000000
conflicting	*tinyint*	如果函数存在冲突则为 1，否则为 0
userdefined	*bit*	如果是用户定义的函数则为 1，否则为 0
rowtype	*tinyint*	如果该行是复制行，则为 1，否则为 0

索引

- (*funcname*) 的唯一索引
- (*funcid*) 的唯一索引

rs_columns

包含有关复制定义的各个列的信息。

列	数据类型	说明
prsid	*int*	该对象的主 Replication Server
objid	*rs_id*	此列所属的表/函数复制定义 ID 或函数 ID
colname	*varchar(255)*	列或参数名
colnum	*smallint*	列号

列	数据类型	说明
coltype	*tinyint*	列或参数的数据类型： • 0 – *char* • 1 – *binary* • 4 – *text* • 5 – *image* • 6 – *tinyint* • 7 – *smallint* • 8 – *int* • 9 – *real* • 10 – *float* • 11 – *bit* • 12 – *datetime* • 13 – *smalldatetime* • 14 – *money* • 15 – *smallmoney* • 16 – *numeric* • 17 – *decimal* • 18 – *varchar* • 19 – *varbinary* • 25 – *unichar* • 27 – *date* • 28 – *time* • 29 – *unitext* • 30 – *bigint* • 31 – *usmallint* • 32 – *uint* • 33 – *ubigint* • 35 – *bigdatetime* • 36 – *bigtime* • 110 – *univarchar*
length	*int*	已声明的数据的长度
searchable	*tinyint*	如果是可搜索键，则为 1；否则为 0
primary_col	*tinyint*	如果是主键，则为 1；否则为 0
fragmentation	*tinyint*	如果是分段键，则为 1；否则为 0

列	数据类型	说明
rowtype	*tinyint*	如果要复制行，则为 1；否则为 0
status	*int*	掩码，可以是下面的一项或多项： • 0x01 – 列被声明为 *identity* • 0x02 – 列被声明为 *timestamp* 列 • 0x04 – 列为 *rs_address* 数据类型 • 0x08 – 列具有 **replicate_if_changed** 状态 • 0x10 – 复制表中的列允许使用空值（仅限于 *text*、*unitext* 或 *image* 列） • 0x20 – 列被发送到备用连接（仅在内部复制定义中） • 0x40 – 列被标记为已从内部复制定义中删除（仅在内部复制定义中） • 0x200 – 作为 *identity* 发布 • 0x400 – 作为 *timestamp* 发布 • 0x1000 – 声明为 Java 列 • 0x2000 – 发布为 Java 列
basecolnum	*smallint*	基本复制定义中的列位置。缺省值为 *colnum* 值。
repl_colname	*char(255)*	复制表中的列名。缺省值为 *colname*。
declared_dtid	*rs_id*	数据类型 ID。对于用户定义的数据类型，为表的外键。
publ_dtid	*rs_id*	复制定义中指定的已发布的数据类型。如果未指定已发布的数据类型，*publ_dtid* 等于 *declared_dtid*。
publ_base_col-type	*tinyint*	已发布数据类型的基本数据类型。如果未指定已发布的数据类型，*publ_base_coltype* 等于 *coltype*。
publ_length	*int*	已发布数据类型的最大长度。
version	*rs_id*	标识复制定义版本。
ref_objowner	*varchar(30)*	由 references 子句标识并用作 RI 约束的复制对象所有者。该对象所有者是复制数据库中标识为 RI 约束的表所有者。
ref_objname	*varchar(255)*	由 references 子句标识并用作 RI 约束的复制对象名称。该对象名称是复制数据库中标识为 RI 约束的表名称。

索引

- （*version*、*colname*）的唯一聚簇索引
- （*objid* 和 *basecolnum*）的唯一索引

- （*objid*、*colname*）的唯一索引
- （*objid* 和 *colnum*）的唯一索引
- （*version*、*colnum*）的唯一索引

rs_config

保存一组缺省配置参数值，可以使用 **configure replication server** 命令修改这些参数值。您还可以使用 **alter connection**、**alter logical connection** 或 **alter route** 命令为特定目标设置某些参数。

有关 *rs_config* 表中的配置参数的详细信息，请参见《Replication Server 管理指南第一卷》。

列	数据类型	说明
optionname	*varchar(30)*	参数的名称，例如：**memory_max**、**cm_max_connections** 要查看这些参数及其说明的列表，请对 *rs_config* 表执行 **select *** 语句。
objid	*rs_id*	该选项引用的对象的 ID。如果设置为 0，则适用于整个系统。
charvalue	*varchar(255)*	参数的字符值。
status	*tinyint*	不使用此列。
comments	*varchar(255)*	有关参数的注释。

索引

（*objid* 和 *optionname*）的唯一聚簇索引

rs_databases

存储 Replication Server 节点已知的数据库的名称。

列	数据类型	说明
dsname	*varchar(30)*	数据服务器名
dbname	*varchar(30)*	数据库名称
dbid	*int*	数据库的唯一标识符

列	数据类型	说明
conn_id	int	数据库连接的唯一标识符。对于: • 缺省连接 – connid 等同于 dbid • 替代连接 – connid 不等同于 dbid
dist_status	*cs_int*	连接的状态。可以是: • 0x1 – 有效 • 0x2 – 挂起 • 0x4 – 由与备用相关的操作挂起 • 0x8 – 等待标记 • 0x10 – 将发布 **dbcc ('ltm', 'ignore')** • 0x20 – 等待转储标记以初始化备份数据库 • 0x40 – 在 *ltype* 等于“P”时切换相关的重复检测 • 0x40 – 在 *ltype* 等于“L”时允许进行切换 • 0x80 – 暂时不进行任何分组 • 0x100 – 等待 resync 标记
src_status	*cs_int*	源的状态: • 0x1 – 有效 • 0x2 – 挂起 • 0x4 – 由与备用相关的操作挂起 • 0x10 – DIST 线程已挂起
attributes	*tinyint*	值为以下一项: • 1 – 分发 • 2 – 源
errorclassid	*rs_id*	该数据库的错误类
funcclassid	*rs_id*	该数据库的函数字符串类
prsid	*int*	管理该数据库的 Replication Server 的 ID
rowtype	*tinyint*	指示行类型: • 1 – 行已复制 • 0 – 行未复制

列	数据类型	说明
sorto_status	*tinyint*	指示排序顺序检查是否已完成。以下值之一： • 0 – 未检查 • 1 – 已检查
ltype	*char(1)*	此行所代表的数据库的类型。值为以下一项： • P – 物理数据库 • L – 逻辑数据库连接
ptype	*char(1)*	热备份应用程序中的数据库的类型。值为以下一项： • A – 活动数据库 • S – 备份数据库 • L – 逻辑数据库连接
ldbid	*int*	与数据库相关联的逻辑连接的 *dbid*。如果无逻辑连接，*ldbid* 与 *dbid* 相同。
enable_seq	int	在活动数据库切换期间或创建备用数据库过程中使用的序列号。
rs_errorclassid	*rs_id*	该数据库的 Replication Server 错误类

索引

- （*dsname*、*dbname*、*ltype*）的唯一聚簇索引
- （*ptype* 和 *ldbid*）的唯一索引
- （*dbid* 和 *ltype*）的唯一索引
- （*dsname*、*dbname*、*ptype*）的唯一索引

rs_datatype

存储复制定义中的所有用户定义数据类型 (UDD) 的属性信息。

列	数据类型	说明
prsid	*int*	可以是： • 主 Replication Server 的 ID • 对于全局定义的 UDD，为 0
classid	*rs_id*	数据类型所属的数据类型类的 ID

列	数据类型	说明
dtname	*varchar(30)*	数据类型的唯一名称
dtid	*rs_id*	数据类型的唯一 ID
base_coltype	*tinyint*	数据类型的基本数据类型的 ID。可以是： • 0 – *char* • 1 – *binary* • 2 – *longchar*（不使用） • 3 – *longbinary*（不使用） • 4 – *text* • 5 – *image* • 6 – *tinyint* • 7 – *smallint* • 8 – *int* • 9 – *real* • 10 – *float* • 11 – *bit* • 12 – *datetime* • 13 – *smalldatetime* • 14 – *money* • 15 – *smallmoney* • 16 – *numeric* • 17 – *decimal* • 18 – *varchar* • 19 – *varbinary* • 21 – *sensitivity* • 25 – *unichar* • 27 – *date* • 28 – *time* • 29 – *unitext*

列	数据类型	说明
		• 30 – *bigint* • 31 – *usmallint* • 32 – *uint* • 33 – *ubigint* • 35 – *bigdatetime* • 36 – *bigtime* • 101 – *numeric*（实际值） • 102 – *money*（实际值） • 103 – *real*（实际值） • 104 – *float*（实际值） • 105 – *identity*（实际值） • 106 – *timestamp*（实际值） • 107 – *sensitivity*（实际值） • 110 – *univarchar*
length	*int*	该数据类型的值的最大长度。对于将屏蔽定义为 *decimal* 或 *money* 的 UDD，该值为最大精度加 4。
status	*int*	状态。（参见 *rs_columns* 表中的 *status* 列。）
length_err_act	*tinyint*	值超过 *length* 中指定的长度时要执行的操作。可以是： • 1 – 错误 • 2 – 继续 • 3 – 截断左侧 • 4 – 截断右侧 • 5 – 上舍入 • 6 – 出现错误后上舍入并继续 • 7 – 出现错误后上舍入并使用缺省值 • 8 – 出现错误后上舍入并使用最小值 • 9 – 出现错误后上舍入并使用最大值 • 10 – 下舍入 • 11 – 出现错误后下舍入并继续 • 12 – 出现错误后下舍入并使用缺省值 • 13 – 出现错误后下舍入并使用最小值 • 14 – 出现错误后下舍入并使用最大值
mask	*varchar(255)*	数据类型屏蔽。对于非 null 屏蔽，数据类型的基本类型必须为 *char*。

列	数据类型	说明
scale	*int*	小数点后的最大位数。仅对 *money* 或 *decimal* 的屏蔽有效。
default_len	*tinyint*	值的长度，该值在 *default_val* 列中。
default_val	*binary(255)*	缺省值。在转换到该数据类型的过程中为目标值提供缺少的成分。
delim_pre_len	*tinyint*	*delim_pre* 值的长度。
delim_pre	*binary(30)*	将非 Java 值映射到函数字符串中时使用的后缀字符或字符串。如果使用了基本数据类型的分隔符前缀，则为空字符串。
delim_post_len	*tinyint*	*delim_post* 的长度。
delim_post	*binary(30)*	将非 Java 值映射到函数字符串中时使用的后缀字符或字符串。如果使用了基本数据类型的分隔符前缀，则为空字符串。
min_boundary_len	*tinyint*	*min_boundary* 列中的值的长度。 • 1 – 错误 • 2 – 使用缺省值 • 3 – 使用最小值 • 4 – 使用最大值
min_boundary	*binary(255)*	数据类型的最小可接受值。
min_boundary_err_act	*tinyint*	值超过 min_boundary 设置的最小界限时要执行的操作。可以是： • 1 – 错误 • 2 – 使用缺省值 • 3 – 使用最小值 • 4 – 使用最大值
max_boundary_len	*tinyint*	*max_boundary* 中的值的长度。
max_boundary	*binary(255)*	数据类型的最大可接受值。

列	数据类型	说明
maximum_boundary_err_act	*tinyint*	值超过 max_boundary 设置的最大界限时要执行的操作。可以是： • 1 – 错误 • 2 – 使用缺省值 • 3 – 使用最小值 • 4 – 使用最大值
rowtype	*tinyint*	指示某行是在 Replication Server 本地还是分布到域中的所有 Replication Server。可以是： • 0 – 本地 • 1 – 全局
canonic_type	*tinyint*	在发送动态 SQL 执行命令时，DSI 使用 *canonic_type* 值将 UDD 转换为正确的数据类型。值 255 表示该数据类型与动态 SQL 不兼容。

索引

- (*dtid*) 的唯一索引
- (*name*) 的唯一索引
- (*classid*) 的非唯一索引
- (*prsid*) 的非唯一索引

rs_dbreps

存储除名称集以外的所有关于数据库复制定义的信息。它将被复制到所有版本号为 12.6 或更高的节点。

列	数据类型	说明
dbrepid	*rs_id*	数据库复制定义 ID
dbrepname	*varchar (255)*	数据库复制定义名称
prsid	*int*	主 Replication Server ID
dbid	*int*	主数据库 ID
ownerid	*rs_id*	创建了数据库复制定义的 Replication Server 用户
requestdata	*datetime*	创建数据库复制定义的时间

列	数据类型	说明
status	*int*	子集内容的位图： • 0x0001 – 显示表列表 • 0x0002 – 否定表 • 0x0004 – 显示函数列表 • 0x0008 – 否定函数 • 0x0010 – 显示列表事务 • 0x0020 – 否定事务 • 0x0040 – 显示系统过程列表 • 0x0080 – 否定系统过程 • 0x0100 – 不复制 DDL • 0x0200 – 显示 **update** 列表 • 0x0400 – 否定列表的 **update** 语句复制 • 0x0800 – 显示 **delete** 列表 • 0x1000 – 否定列表的 **delete** 语句复制 • 0x2000 – 显示 **select into** 列表 • 0x4000 – 否定列表的 **select into** 语句复制 • 0x8000 – 显示 **insert select** 列表 • 0x10000 – 否定列表的 **insert select** 语句复制
minvers	*int*	可以将此表复制到的 Replication Server 的最早版本。

索引

（*dbrepid*、*dbid* 和 *dbrepname*）的唯一索引。

rs_dbsubsets

存储数据库复制定义的名称集。它将被复制到所有版本号为 12.6 或更高的节点。

列	数据类型	说明
dbrepid	*rs_id*	数据库复制定义 ID
prsid	*int*	主 Replication Server ID

列	数据类型	说明
type	*char*	项目类型： • T – 表名。 • F – 函数名。 • X – 事务名。 • P – 系统过程名。 • U – **update** 命令 • L – **delete** 命令 • I – **insert select** 命令 • S – **select into** 命令
owner	*varchar* (30)	表或函数的所有者名称，或者执行了事务或系统过程的用户名。 * 指代全部所有者或用户。
name	*varchar* (255)	表、函数、事务或系统过程名称。 * 指代全部表、函数、事务和系统过程。

索引

（*dbrepid*、*subtype*、*owner* 和 *name*）的唯一索引。

rs_dictionary

存储不允许在口令中使用的字符组合。

管理员必须通过使用其自己的脚本输入字符和数字组合来填充字典表。

列	数据类型	说明
words	*varchar(30)*	不允许使用的字符组合

rs_diskaffinity

存储有关磁盘分区和数据库连接或路由之间密切连接的信息。

列	数据类型	说明
partition_id	*int*	Replication Server 指派的分区 ID
dbid_or_siteid	*int*	Replication Server 或数据库的 ID

列	数据类型	说明
status	*int*	密切连接的状态。有效值为: • 0x01 - 有效 • 0x02 – 已过时

索引

(*dbid_or_siteid*) 的唯一聚簇索引

rs_diskpartitions

存储有关 Replication Server 用于稳定消息队列的磁盘分区的信息。

列	数据类型	说明
name	*varchar(255)*	磁盘设备的操作系统名
logical_name	*varchar(30)*	用户指派给分区的名称
id	*int*	Replication Server 指派的分区 ID
num_segs	*int*	分区的总大小（以段为单位）
status	*int*	磁盘分区的状态。有效值为: • 1 – 联机 • 2 – 分区正被删除
vstart	*int*	Replication Server 开始写入分区时的偏移量（以 MB 为单位）

索引

- (*logical_name*) 的唯一聚簇索引
- (*name*) 的唯一索引

rs_encryptionkeys

存储 Replication Server 中使用的加密密钥。

列	数据类型	说明
name	*varchar(30)*	加密密钥的名称。

列	数据类型	说明
Value	*binary(128)*	加密密钥的值。
status	*int*	加密密钥的状态。
ctime	*datetime*	创建时间或上次修改时间。

rs_erroractions

将数据服务器错误号映射为 Replication Server 要执行的一项操作。

列	数据类型	说明
ds_errorid	*int*	数据服务器错误号
errorclassid	*rs_id*	错误类 ID（参见 *rs_classes*）
action	*tinyint*	发生错误时要执行的操作： • 1 – 忽略错误 • 2 – 停止复制 • 3 – 输出警告消息 • 4 – 将条目写入例外日志 • 5 – 重试该事务，如果仍然失败，则记录该事务 • 6 – 重试事务达到一定次数之后，如果仍然失败则停止复制
prsid	*int*	以该行为主行的节点

索引

- （*ds_errorid*、*errorclassid*）的唯一索引
- (*errorclassid*) 的聚簇索引

rs_exceptscmd

存储用于从例外日志检索事务文本的信息。

存储在 *rs_systext* 系统表中的文本包括：

- 源命令 – Replication Server 收到的用户事务的文本。
- 输出命令 – Replication Server 通过函数字符串为数据库准备的事务的文本。输出命令可以是语言命令也可以是 RPC。

每个源命令或输出命令在 *rs_exceptscmd* 中各占一行。

列	数据类型	说明
sys_trans_id	*rs_id*	由系统为事务指派的事务 ID
src_cmd_line	*int*	已记录事务内源命令行号
output_cmd_index	*int*	已记录事务内输出命令的行号
cmd_type	*char(1)*	命令类型： • S – 源命令 • L – 语言输出命令 • R – RPC 输出命令
cmd_id	*rs_id*	*rs_systext* 中的索引

索引

(*cmd_id*) 的唯一索引

rs_exceptshdr

存储有关失败的事务的信息。事务的源命令和输出命令存储在系统表 *rs_exceptscmd* 和 *rs_systext* 中。*rs_exceptscmd* 和 *rs_exceptshdr* 中事务的所有行都由列 *sys_trans_id* 标识。

列	数据类型	说明
sys_trans_id	*rs_id*	由系统为此事务指派的事务 ID
rs_trans_id	*binary(120)*	由 Replication Server 生成的唯一事务 ID
app_trans_name	*varchar(30)*	由用户指定的事务名称
orig_siteid	*int*	源数据库的 ID
orig_site	*varchar(30)*	源数据库的数据服务器名
orig_db	*varchar(30)*	源数据库的名称
orig_time	*datetime*	事务开始的时间
orig_user	*varchar(30)*	在源节点上提交事务的用户
error_siteid	*int*	发生错误的节点的 ID
error_site	*varchar(30)*	发生错误的数据服务器的名称
error_db	*varchar(30)*	发生错误的数据库的名称

列	数据类型	说明
log_time	*datetime*	发生错误的时间
ds_error	*int*	数据服务器错误号
ds_errmsg	*varchar(255)*	数据服务器错误消息
error_src_line	*int*	导致错误的命令的行号
error_proc	*varchar(255)*	发生错误时正在执行的过程
err_output_line	*int*	导致错误的输出命令的行号
log_reason	*char(1)*	记录事务的原因： • O – 指示 DSI 队列中存在孤立事务 • E – 数据服务器错误映射到了 LOG 或 RETRY_LOG • S – 指示由于在执行 **resume connection** 命令时使用 **skip transaction** 选项而跳过事务 • D – 事务已由 **sysadmin log_first_tran** 命令记录
trans_status	*smallint*	事务状态 – 以下一项或多项： • 0x0001 – 孤立事务 • 0x0002 – 已记录发往主节点的事务 • 0x0004 – 事务发生冲突
retry_status	*smallint*	事务的重试状态 – 以下值之一： • 1 – 重试成功 • 2 – 事务尚未提交
app_usr	*varchar(30)*	在错误节点应用事务的用户的名称
app_pwd	*varchar(30)*	在错误节点应用事务的用户的口令

索引

(*sys_trans_id*) 的唯一索引

rs_exceptslast

存储写入例外日志中的上一次记录的事务的源 ID、辅助队列 ID 以及相关信息。

列	数据类型	说明
error_db	*int*	发生错误的数据库
origin	*int*	事务的源数据库
origin_qid	*binary(36)*	此源数据库中上一个事务的 qid
secondary_qid	*binary(36)*	此源数据库中记录的上一个事务的辅助 qid
status	*tinyint*	事务的状态： • 0 – 有效：此源数据库没有丢失事务 • 1 – 正在检测丢失：您应当确定源数据库中是否曾丢失任何事务 • 2 – 检测完丢失之后正在拒绝消息：此源数据库的可能已丢失事务
origin_time	*datetime*	事务发生时源节点的时间
log_time	*datetime*	记录事务的时间
lorigin	*int*	作为消息来源的逻辑数据库

索引

- （*error_db*、*origin*）的唯一索引
- （*error_db*、*origin*、*status*）的唯一索引

rs_funcstrings

存储与各个函数相关联的函数字符串。

列	数据类型	说明
prsid	*int*	以该行为主行的节点
classid	*rs_id*	函数字符串所属的类
funcid	*rs_id*	该字符串所适用的函数
name	*varchar(255)*	函数字符串名

列	数据类型	说明
fstringid	*rs_id*	该函数字符串的 ID
attributes	*int*	函数字符串的属性： • 0x01 – 函数发生冲突 • 0x02 – RPC • 0x04 – 已更改 • 0x08 – 用于除 **rs_writetext** 以外的所有函数，指示函数没有输出命令并且未向复制数据服务器发送任何内容。 • 0x10 – 缺省输入 • 0x20 – 缺省输出 • 0x40 – **writetext** 输出用于 **rs_writetext** 函数字符串 • 0x80 – **writetext** 输出和 **rs_writetext** 函数字符串的 **with log** 选项一起使用 • 0x100 – **rs_writetext**、**rs_textptr_init** 或 **rs_get_textptr** 函数的函数字符串 • 0x200 – **writetext** 输出和 **rs_writetext** 函数字符串的 **no log** 选项一起使用 • 0x400 – 函数字符串包括将访问非键列的值的一个或多个变量 • 0x800 – *rs_default_fs* 系统变量已用于输出语言模板 • 0x1000 – **none** 输出用于 **rs_writetext** 函数字符串
parameters	*smallint*	此函数字符串中的参数个数
param_hash	*int*	输入模板的散列值
expiredate	*datetime*	函数字符串应该到期的日期。它用于动态函数字符串到期
rowtype	*tinyint*	如果该行是复制行，则为 1，否则为 0
minvers	*int*	支持函数字符串所需的最低版本。这意味着，如果函数字符串的 *minvers* 值为 15.0，它将不会复制到低于 15.0 的节点

索引

- （*classid*、*funcid*、*name*）的唯一聚簇索引
- (*fstringid*) 的唯一索引
- (*funcid*) 的非唯一索引

rs_functions

存储有关 Replication Server 函数的信息。

`rs_functions` Replication Server 15.7 以前版本中的系统表。对于 15.7 和更高版本，`rs_functions` 是 `rs_clsfunctions` 和 `rs_objfunctions` 系统表联合的视图。

列	数据类型	说明
prsid	*int*	以该函数为主函数的节点。
funcname	*varchar(255)*	函数名。
funcid	*rs_id*	函数 ID。
objid	*rs_id*	应用函数的对象。类范围的函数的 NULL_OBJECT_ID (0x00000000) 存储在该列中。
conflicting	*tinyint*	如果函数存在冲突则为 1，否则为 0。
userdefined	*bit*	如果是用户定义的函数则为 1，否则为 0。
rowtype	*tinyint*	如果该行是复制行，则为 1，否则为 0。

索引

注意： 仅当 `rs_functions` 为表时存在索引，该函数存在于 Replication Server 15.7 之前的版本中。

- (*objid*) 的聚簇索引
- （*objid* 和 *funcname*）的唯一索引
- (*funcid*) 的唯一索引

rs_idnames

存储 ID Server 已知的 Replication Server 和数据库名称。rs_idnames 表仅在 ID Server 节点相关。

列	数据类型	说明
name1	*varchar(30)*	Replication Server 或数据服务器名
name2	*varchar(30)*	数据库名；如果是 Replication Server，则为 ""

列	数据类型	说明
type	*int*	Replication Server 或数据库： • 8 – Replication Server • 9 – 数据库
id	*int*	指派给 Replication Server 或数据库的唯一 ID
ltype	*char(1)*	数据库的类型： • P – 物理数据库 • L – 逻辑数据库

索引

（*name1*、*name2* 和 *ltype*）的唯一聚簇索引

rs_ids

存储用于各种类型的对象的上一个 ID。

列	数据类型	说明
typename	*varchar(30)*	该对象类型的名称。例如，“subscriptions”、“objects”
objid	*int*	用于此对象类型的上一个 ID

列	数据类型	说明

objtype	*tinyint*	• 对象类型：
		  • 1 – 预订
		  • 2 – 对象
		  • 3 – 类
		  • 4 – 用户
		  • 5 – 函数
		  • 6 – 函数字符串
		  • 7 – 错误日志
		• 例外日志类型：
		  • 12 – 拒绝事务
		• 节点 ID 类型：
		  • 8 – Replication Server ID
		  • 9 – 数据库 ID
		• 稳定队列参数：
		  • 10 – 磁盘分区 ID
		• 预订模块使用的计数器：
		  • 13 – 用于预订模块的计数器
		• 恢复管理器 ID：
		  • 14 – 恢复 ID 类型
		  • 15 – 重新实现 ID
		  • 16 – 发布 ID
		  • 17 – 项目 ID
		  • 18 – **where** 子句 ID
		  • 19 – UDD ID

索引

(*objtype*) 的唯一聚簇索引

rs_lastcommit

Replication Server 使用 `rs_lastcommit` 表中的信息查找从各个数据源提交的上一个事务。

rs_lastcommit 表存储在每个用户数据库中，而不是在 RSSD 中。

列	数据类型	说明
origin	*int*	某行所代表的主数据库的 ID 号码。
origin_qid	*binary*	标识源数据库中的稳定队列中上一次提交的事务。
secondary_qid	*binary*	如果存在源数据库的预订实现队列，该列中包含该队列中已在复制数据库中提交的上一个事务。
origin_time	*datetime*	事务发生时源节点的时间。
dest_commit_time	*datetime*	事务被提交到目标的时间。
pad1	*binary(255)*	填充内容，用于填充行以使每个数据库页只容纳一行。
pad2	*binary(255)*	填充内容，用于填充行以使每个数据库页只容纳一行。
pad3	*binary(255)*	填充内容，用于填充行以使每个数据库页只容纳一行。
pad4	*binary(255)*	填充内容，用于填充行以使每个数据库页只容纳一行。
pad5	*binary(255)*	填充内容，用于填充行以使每个数据库页只容纳一行。
pad6	*binary(255)*	填充内容，用于填充行以使每个数据库页只容纳一行。
pad7	*binary(255)*	填充内容，用于填充行以使每个数据库页只容纳一行。
pad8	*binary(83)*	填充内容，用于填充行以使每个数据库页只容纳一行。

索引

(*origin*) 的唯一聚簇索引

rs_locater

存储稳定队列从其各个发送方收到的上一个定位符字段。

列	数据类型	说明
sender	*int*	发送方节点 ID

列	数据类型	说明
type	*char(1)*	正在使用该行的一方： • R – RSI (route) • D – 用于预订的分配器定位器 • E – Replication Agent 的执行程序 • U – 上次系统升级时的定位器 • W – 用于热备份应用的分配器定位器 • C – 上次成功执行由 Replication Agent 发送的复制定义请求的定位符。 • F – 由 Replication Agent 发送的失败命令的定位符。 • S – Replication Server 跳过的失败命令的定位符。
locater	*binary(36)*	此发送方接收到的上一个队列 ID

在主数据库上执行 **rs_send_repserver_cmd** 时，*rs_locater* 根据是否执行了 **'rs_api'** 内的 RCL 将 RCL 的 OQID 存储在 *type* 中：

- 成功 – *type C*
- 不成功 – *type F*

在为失败的 RCL 执行 **sysadmin skip_bad_repserver_cmd** 时，Replication Server 会将 *rs_locater* 条目更新为 *type* "*S*"。Replication Agent 下次发送该命令时，Replication Server 将跳过失败的命令。

索引

（*sender* 和 *type*）的唯一聚簇索引

rs_maintusers

存储 Replication Server 用于访问其它 Replication Server 和数据服务器的用户登录名和口令。

列	数据类型	说明
destid	*int*	将登录到的 Replication Server 或数据库的节点 ID
username	*varchar(30)*	Replication Server RSI 用户或数据库维护用户的用户名
password	*varchar(30)*	口令

列	数据类型	说明
use_enc_password	*int*	• 0 – 使用普通口令 • 1 – 使用加密口令
enc_password	*varchar(66)*	加密的用户口令

索引

(*destid*) 的唯一聚簇索引

rs_msgs

存储在安装过程中使用的以及某些 Replication Server 存储过程使用的已本地化错误消息。

列	数据类型	说明
msgnum	*int*	消息的唯一 ID 号
langname	*char(30)*	此版本的消息文本的本地语言名。对应于 RSSD Adaptive Server 中的 *@@language* 全局变量。
msgtxt	*varchar(255)*	使用已本地化的语言显示的消息文本。

索引

（*msgnum 和 langname*）的唯一聚簇索引

rs_objects

存储复制定义，一行一个。

列	数据类型	说明
prsid	*int*	创建该对象的主 Replication Server
objname	*varchar(255)*	对象名
objid	*rs_id*	对象 ID
dbid	*int*	数据服务器和数据库的唯一 ID

列	数据类型	说明
objtype	*char(1)*	以下对象类型之一： • R – 表复制定义 • F – 函数复制定义

列	数据类型	说明
attributes	*int*	掩码，可以是下面的一项或多项： • 0x01 – 生成动态函数字符串。 • 0x02 – 复制定义具有 *bigdatetime* 或 *bigtime* 列，并且只能传播到 Replication Server 15.5 或更高版本。 • 0x04 – 为复制定义启用了最少列数。 • 0x08 – 复制定义具有 *identity* 列。 • 0x10 – **replicate_if_changed** 状态。 • 0x20 – 复制定义具有待执行的删除操作。 • 0x40 – 复制定义具有 *text*、*unitext* 或 *image* 列。 • 0x80 – 复制定义由备用数据库使用。 • 0x0100 – 复制定义的列将发送到备用数据库。 • 0x0200 – 复制定义将传播到 Replication Server 11.0.x 或更低版本。 • 0x0400 – 复制定义已用作主表的基本复制定义。 • 0x0800 – 复制定义仅供内部使用。 • 0x1000 – 主表和复制表中的对象名或列名不同。 • 0x4000 – 复制定义具有列级转换。 • 0x8000 – 复制定义具有使用 UDD 声明的列。 • 0x10000 – 复制定义具有超过 255 个字节的 *char*、*varchar*、*binary* 或 *varbinary* 列，并且只能传播到 Replication Server 12.5 或更高版本。 • 0x20000 – 复制定义具有 *unichar* 或 *univarchar* 列，并且只能传播到 Replication Server 12.5 或更高版本。 • 0x40000 – 复制定义具有 *date* 或 *time* 列，并且只能传播到 Replication Server 12.6 或更高版本。 • 0x80000 – 复制定义具有 *timestamp* 列。作为 *timestamp* 可传播到 Replication Server 15.1；作为 *varbinary* 可传播到 Replication Server 15.0.1 或更低版本。 • 0x200000 – 应用函数复制定义，只能传播到 Replication Server 15.1。 • 0x400000 – 请求函数复制定义，只能传播到 Replication Server 15.1。 • 0x800000 – 在表中不使用动态 SQL。 • 0x2000000 – 为 SQL 复制启用 **update**。 • 0x4000000 – 为 SQL 复制启用 **delete**。 • 0x8000000 – 为 SQL 复制启用 **insert select** • 0x10000000 – **repserver** 在内部禁用 SQL 复制

列	数据类型	说明
ownertype	*char(1)*	此对象的所有者的类型： • U – 用户 • S – 系统
crdate	*datetime*	创建日期和时间
parentid	*rs_id*	留作将来使用。
ownerid	*rs_id*	创建此对象的用户的 ID
rowtype	*tinyint*	如果该行是复制行，则为 1，否则为 0
phys_tablename	*varchar(255)*	主表名 – 在与和该对象相关的数据服务器通信时使用
deliver_as_name	*varchar(255)*	复制表或存储过程的名称
phys_objowner	*char(30)*	复制定义中所指定的主表所有者的名称。 如果未指定表所有者，则为空。
repl_objowner	*char(30)*	复制定义中所指定的复制表所有者的名称。 如果未指定表所有者，则为空。
has_baserepdef	*rs_id*	如果这不是基本复制定义，*has_baserepdef* 的值将与基本复制定义的 *objid* 的值相匹配。或者，将具有下面的值： 0x00 – 基本复制定义
minvers	*int*	指定复制定义的最低版本，从而指定该复制定义可以传播到的 Replication Server。可以是： • 1200 – 传播到 Replication Server 版本 12 或更高版本 • 1150 – 传播到 Replication Server 版本 11.5 或更高版本 • 1000 或 0（零） – 传播到任何 Replication Server • 0（零）-- 用于函数和系统复制定义
version	*rs_id*	唯一地标识复制定义版本。
active_inbound	*int*	执行程序使用此列确定使用哪个复制定义版本。执行程序使用 *active_inbound* 列值为 0 的复制定义版本。
attributes2	*int*	• 0x01 – 执行程序使用此值标识未由分配器使用的复制定义版本。 • 0x02 – 备用 DSI 使用此值标识未由备用 DSI 使用的复制定义版本。

在更改复制定义后，Replication Server 可能会创建一个新的复制定义版本。在这种情况下，Replication Server 会将旧复制定义版本的名称更改为带有“rs_drp”前缀的唯一名称。Replication Server 将名称带有“rs_drp”或“rs_in”前缀的复制定义视为内部复制定义。

当与内部复制定义关联的数据不再存在于复制系统中时，Replication Server 将删除该复制定义。

要在对 *rs_objects* 表的查询中排除内部复制定义，请向查询的 **where** 子句中添加以下内容：

```
and objname not like ‘rs_drp%’ and objname not like
‘rs_in%’
```

例如，下面的查询返回根据 *ling.authors* 主表创建的所有复制定义的名称并排除内部复制定义：

```
select objname from rs_objects where prsid = 16777317
and dbid = 104 and phys_tablename = 'authors' and
phys_objowner = 'ling' and objname not like 'rs_drp%'
and objname not like 'rs_in%
```

索引

- (*objname*) 的唯一聚簇索引
- (*dbid*、*phys_tablename*、*phys_objowner*、*objtype*、*has_baserepdef*、*active_inbound*) 的唯一索引
- (*objid*) 的唯一索引
- (*version*) 的唯一索引

rs_objfunctions

存储有关复制定义的用户函数的信息。

列	数据类型	说明
prsid	*int*	以该函数为主函数的节点
funcname	*varchar(255)*	函数名
funcid	*rs_id*	函数 ID
objid	*rs_id*	复制定义或函数所适用的目标对象的对象 ID。
conflicting	*tinyint*	如果函数存在冲突则为 1，否则为 0
userdefined	*bit*	如果是用户定义的函数则为 1，否则为 0
rowtype	*tinyint*	如果该行是复制行，则为 1，否则为 0

索引

- (*objid*) 的聚簇索引
- （*objid* 和 *funcname*）的唯一索引
- (*funcid*) 的唯一索引

rs_oqid

存储来自源节点的最后一个队列 ID，也用于协调截断点的重置。

列	数据类型	说明
origin_site_id	*int*	源节点的节点 ID
q_number	*int*	队列号
q_type	*int*	队列类型
origin_q_id	*binary(36)*	源数据库上的命令 ID
local_q_id	*binary(36)*	队列的本地 ID
有效的	*int*	检验状态: • 0 – 有效 • 1 – 正在检测丢失 • 2 – 检测完丢失之后正在拒绝消息
origin_lsite_id	*int*	源节点的逻辑数据库的节点 ID

索引

- （*origin_site_id*、*q_number* 和 *q_type*）的唯一聚簇索引

rs_passwords

为有权访问 Replication Server 的每个用户存储口令历史记录。

列	数据类型	说明
uid	*rs_id*	用户 ID。
enc_password	*varchar(66)*	加密口令。
password_date	*datetime*	最初设置口令的日期。

rs_profdetail

记录与 Replication Server 配置文件关联的详细信息。

列	数据类型	说明
profid	*rs_id*	配置文件 ID，这是 *rs_profile* 表的外键
name	*varchar(255)*	配置文件名称。可以为空字符串
pdetailtype	*int*	指定必须为配置文件执行的操作： • 1 – 在 **create connection** 命令后面附加连接配置 • 2 – 在 RSSD 中执行类级别转换定义。 • 3 – 在复制数据库中执行复制数据库对象定义。
pdetailid	*rs_id*	配置文件明细 ID，这是 *rs_systext* 表的外键
sequence	*int*	指示配置文件中的配置文件明细序列。Replication Server 使用序列来确定执行配置文件明细操作的顺序。 **注意：** 始终先执行 Replication Server 的 **create connection** 选项，而无论配置文件明细中的序列如何指定这些选项。

索引

- （*profid*、*sequence*）的唯一索引
- (*id*) 的唯一索引
- (*profid*) 的非唯一索引

rs_profile

存储当前定义的 Replication Server 配置文件。

列	数据类型	说明
name	*varchar(255)*	配置文件名
vers	*varchar(255)*	配置文件版本。可以为空字符串
id	*rs_id*	配置文件 ID
type	*char(1)*	配置文件类型： • C – 连接配置文件

列	数据类型	说明
comments	*varchar(255)*	配置文件说明和信息

索引

- （*name*、*vers*、*type*）的唯一索引
- (*type*) 的非唯一索引

rs_publications

存储有关此 Replication Server 的已知发布的信息。

列	数据类型	说明
prsid	*int*	创建该发布的主 Replication Server
pubname	*varchar(255)*	发布名
pubid	*rs_id*	唯一发布 ID
pdbid	*int*	发布的主数据服务器和数据库的唯一 ID
requestdate	*datetime*	在发布中添加上一个项目的日期和时间
ownerid	*rs_id*	创建此发布的用户的 ID
status	*int*	发布状态： • 0x00 – 无效 • 0x01 – 有效
minvers	*int*	支持该发布所需的最低 Replication Server 版本

索引

- （*pubname*、*pdbid*）的唯一聚簇索引
- (*pubid*) 的唯一索引

rs_queuemsg

将 Replication Server 队列转储到 RSSD 中时，队列条目将被存储到 *rs_queuemsg* 中。如果 rs_queuemsg 表已有某段的行，则先从表中删除这些行，然后转储该段中的最新行。

列	数据类型	说明
q_number	*int*	队列号
q_type	*int*	队列类型
q_seg	*int*	队列段
q_blk	*int*	队列块
q_row	*int*	队列行
len	*int*	队列条目的长度
origin_site_id	*int*	源节点 ID
origin_q_id	*binary(36)*	源指派的队列 ID
origin_time	*datetime*	事务开始的时间
origin_user	*varchar(30)*	在源节点上提交事务的用户
tran_name	*varchar(30)*	事务名称
local_q_id	*binary(36)*	本地 Replication Server 指派的队列 ID
status	*int*	消息状态
reserved	*int*	留待将来使用
tran_len	*smallint*	*tran_id* 的长度
txt_len	*smallint*	命令的长度
tran_id	*binary(120)*	事务 ID
lorigin_site_id	*int*	作为队列条目的源的逻辑连接的节点 ID。
version	*int*	消息的发行版本

索引

（*q_number*、*q_type*、*q_seg*、*q_blk* 和 *q_row*）的唯一聚簇索引

rs_queuemsgtxt

存储命令或消息的文本部分于稳定队列中。每个稳定队列条目由该表中的一行或多行表示。当稳定队列条目中数据的长度超过了最大命令字段长度（255 个字节）时，将需要多个行。

列	数据类型	说明
q_number	*int*	队列号
q_type	*int*	队列类型
q_seg	*int*	包含该消息的段
q_blk	*int*	包含该消息的段内的块
q_row	*int*	包含该消息的块内的行
q_seq	*int*	该条目的行的序列号
txt	*varchar(255)*	条目的文本
txtbin	*binary(255)*	二进制文本

索引

（*q_number*、*q_type*、*q_seq*、*q_seg*、*q_blk* 和 *q_row*）的唯一缺省索引

rs_queues

存储信息以允许节点恢复。由 Replication Server 稳定队列管理器和有保证的交付系统使用。

列	数据类型	说明
number	*int*	队列 ID。该列显示代表下列某一项的编号： • 入站队列的源数据库，或 • 出站队列的目标数据库或 Replication Server 这两个值与 *rs_databases* 系统表的 *dbid* 列中的数据库条目以及 *rs_sites* 系统表的 *id* 列中的 Replication Server 条目相对应。

列	数据类型	说明
type	*int*	队列类型： • 0 – 出站队列 • 1 – 入站队列 • 大负数 -- 预订实现队列
state	*int*	该队列的当前状态： • 0 – 失败 • 1 – 活动 • 2 – 正在删除
twosave	*int*	指示 SQM 段中的所有消息都已得到目标的确认后，Replication Server 维护此段的秒数。如果设置为 **-1**，表示精确的设置。
truncs	*int*	截断点的数目

索引

（*number* 和 *type*）的唯一聚簇索引

rs_recovery

记录当发生故障而进行恢复时必须由 Replication Server 执行的操作。

列	数据类型	说明
action	*int*	代表可恢复的操作： • 1 – 创建路由 • 2 – 删除路由 • 3 – 独立模式 • 4 – 重建队列 • 5 – 日志恢复 • 6 – 重新启动 LTM，转到日志顶部 • 7 – 创建备用 • 8 – 切换活动 • 9 – 用于 DSI 或实现队列的精确的保存间隔 • 10 – 切换到活动状态之后，退出 DSI 辅助重复检测 • 11 – 删除备用 • 12 – 变更分配器定位器 • 13 – 删除带复制定义的段 • 14 – 删除未决的复制定义 • 15 – 删除未决表或带引用计数器的函数复制定义 • 16 – 创建架构复制定义（适用于自动生成架构复制定义）
id	*rs_id*	给每个行指派一个唯一 ID。
seqnum	*int*	对于影响多行的操作，该列存储各行的序列号。
state	*int*	包含历经一定数量状态的可恢复操作的当前状态。
text	*binary(255)*	完成操作所需的数据。
textlen	*int*	文本数据的长度。

索引

(*id*) 的唯一索引

rs_repdbs

包含主 Replication Server 所知的所有数据库的有关信息。在复制节点上为数据库输入预订时，就会存储该信息。

列	数据类型	说明
dbid	*int*	唯一的数据库 ID
dsname	*varchar(30)*	数据服务器名
dbname	*varchar(30)*	数据库名称
controllerid	*int*	管理此数据库的 Replication Server

索引

- (*controllerid*) 的聚簇索引
- (*dbid*) 的唯一索引
- （*dsname*、*dbname*）的唯一索引

rs_repobjs

在复制 Replication Server 上存储复制定义的自动更正标志。您可以使用 **set autocorrection** 命令将该标志设置为“开启”或“关闭”。

列	数据类型	说明
objid	*rs_id*	复制定义对象 ID
dbid	*int*	存储复制数据的数据库的 ID
attributes	*int*	有效值: • 0x01 – 自动更正标志处于开启状态 • 0x02 – 复制定义不使用动态 SQL。

索引

（*objid* 和 *dbid*）的唯一聚簇索引

rs_routes

存储有关网络通信量的路由信息。

列	数据类型	说明
dest_rsid	*int*	数据服务器或 Replication Server 的 ID
through_rsid	*int*	通过该 Replication Server 到达目标。对于直接路由，*through_rsid* 的值与 *dest_id* 的值相同。
source_rsid	*int*	定义此路由时所在的 Replication Server
status	*tinyint*	路由的状态： • 1 – 正被初始化 • 2 – 路由在此节点有效（当源 Replication Server 和目标 Replication Server 的状态均为 2 时，路由有效） • 3 – 正常删除此路由 • 4 – 立刻删除此路由
suspended	*tinyint*	值为以下一项： • 0 – 路由处于活动状态 • 1 – 路由被挂起 • 2 – 正在重建路由。正在设置截断点。 • 3 – 路由被挂起。正在设置截断点。 • 8（屏蔽） – 对于 RSI 出站队列，指示复制 Replication Server 为这一正在发送的 Replication Server 将 *rs_locater* 表中的 *locater* 字段设置为 0。
src_version	*int*	该路由的源 Replication Server 的版本。请注意，该版本为 RSI 版本（而不是出现在 *current_rssd_version* 下 **rs_config** 存储过程中的版本）。 • 1000 – 指派给任何 10.1 之前的 Replication Server 的版本 • 1010 – 版本 10.1 • 1100 – 版本 11.0 • 1150 – 版本 11.5 • 1200 – 版本 12.0 有关支持的任何其它版本号，请参见《Replication Server 发行公告》。

索引

（*dest_rsid* 和 *source_rsid*）的唯一聚簇索引

rs_routeversions

存储有关路由的每一端上的 Replication Server 的版本信息。

列	数据类型	说明
dest_rsid	*int*	目标 Replication Server 的 ID
source_rsid	*int*	定义此路由时所在的源 Replication Server 的 ID
dest_rssd_id	*int*	目标 Replication Server 的 RSSD 的 ID
route_version	*int*	目标 Replication Server 和源 Replication 的最低节点版本
min_path_version	*int*	留待将来使用
marker_serial_no	*int*	供内部使用
status	*int*	路由状态： • 0x00 – 有效 • 0x01 – 正在进行路由升级/恢复，或者需要进行路由升级/恢复。 • 0x02 – 路由升级/恢复已完成。这是 Replication Manager 使用的临时状态。
proposed_version	*int*	正在转换新的路由值

索引

（*dest_rsid* 和 *source_rsid*）的唯一聚簇索引

rs_rules

存储预订规则。预订子句中的每一项在 rs_rules 表中各占一行。

列	数据类型	说明
prsid	*int*	该对象的主 Replication Server
subid	*rs_id*	应用该规则的预订的 ID。或者，如果是预订项目，则为应用该规则的 **where** 子句的 ID。

列	数据类型	说明
objid	*rs_id*	该预订的表或函数复制定义的 ID
dbid	*int*	存储预订数据的数据库的 ID
subtype	*int*	预订类型： • 0x01 – 范围预订 • 0x02 – 等同性预订 • 0x80 – 项目预订
primary_sre	*int*	如果设置，则预订应包括在主 Replication Server 上的预订解析引擎中
replicate_sre	*int*	如果设置，则预订应包括在复制 Replication Server 上的预订解析引擎中
colnum	*smallint*	基本列号的值
valuetype	*tinyint*	操作数的数据类型（如 SYBCHAR）
low_flag	*tinyint*	低值类型的位图： • 0x01 – 开区间 • 0x02 – 闭区间 • 0x04 – 无限 • 0x08 – 等同 • 0x20 – rs_address
high_flag	*tinyint*	高值类型的位图： • 0x01 – 开区间 • 0x02 – 闭区间 • 0x04 – 无限 • 0x08 – 等同 • 0x20 – rs_address
low_len	*int*	低值的长度
high_len	*int*	高值的长度
low_value	*binary(255)*	低值的二进制表示
high_value	*binary(255)*	高值的二进制表示
dtid	*rs_id*	复制定义中定义的列的已声明数据类型的 ID。

索引

- （*subid*、*colnum*、*primary_sre*、*replicate_sre*、*subtype*）的唯一索引
- （*subid* 和 *colnum*）的唯一索引
- （*objid*、*subtype*、*dbid*）的聚簇索引

rs_schedule

存储有关在 Replication Server 创建的日程表的信息。

列	数据类型	说明
sched_name	*varchar(30)*	日程表的名称。
sched_time	*varchar(255)*	采用限制的 UNIX cron 格式的日期和时间字符串，指示 Replication Server 执行指定操作的时间。
status	*int*	打开或关闭日程表。有效值为： • 0 – off • 1 – on
type	*int*	要在日程表中运行的命令的类型。值为： • 0 – shell 命令
ownerid	*rs_id*	创建此日程表的用户的 ID。

索引

(*sched_name*) 的唯一聚簇索引

rs_scheduletxt

存储您在 Replication Server 创建的日程表的命令部分。每个日程表条目由 rs_scheduletxt 表中的一行或多行表示。当信念超过最大命令字段长度（255 个字节）时，将需要多个行。

列	数据类型	说明
sched_name	*varchar(30)*	日程表的名称。
sequence	*int*	该日程表的行的序列号。
textval	*varchar(255)*	shell 命令的完整路径。

索引

- （*sched_name*、*sequence*）的唯一聚簇索引
- (*sched_name*) 的部分索引

rs_segments

包含有关各个段的分配的信息。Replication Server 使用原始磁盘空间存储消息数据。

列	数据类型	说明
partition_id	*int*	分区的唯一 ID
q_number	*int*	该分区所属的队列
q_type	*int*	该队列的类型
partition_offset	*int*	分区内段的偏移量
logical_seg	*int*	队列内段的偏移量
used_flag	*int*	段的当前状态： • 0 – 不活动 • 1 – 活动 • *n* – 保存间隔：*n* 表示删除此段所需的实际时间（从某基本日期算起，以秒为单位计量）
version	*int*	段的当前版本。每次使用后，版本号都会增加。
flags	*int*	切换为活动状态之后，在 DSI 队列的上一个段设置为 1

索引

（*partition_id* 和 *partition_offset*）的唯一聚簇索引

rs_sites

存储某节点已知的 Replication Server 的名称。

列	数据类型	说明
name	*varchar(30)*	Replication Server 名称
id	*int*	指派给此 Replication Server 的节点 ID

列	数据类型	说明
status	*tinyint*	不使用

索引

- (*name*) 的唯一索引
- (*id*) 的唯一聚簇索引

rs_statcounters

存储每个计数器的描述性信息。这些值不会更改。

列	数据类型	说明
counter_id	*int*	唯一计数器标识号
counter_name	*varchar(60)*	描述性计数器名称
module_name	*varchar(30)*	计数器所属模块的名称
display_name	varchar(30)	用于 RCL 命令的计数器名称
counter_status	*int*	计数器状态。位屏蔽值为: • 0x001 – 内部使用，不显示 • 0x002 – 内部使用，不显示 • 0x004 – sysmon（刷新为 **admin statistics, sysmon** 的输出的计数器） • 0x008 – 必需示例（始终被取样的计数器） • 0x010 – 不重新设置（从不重新设置计数器） • 0x020 – 持续时间（计数器记录完成操作所需的时间，通常以 0.01 秒计） • 0x040 – 内部使用，不显示 • 0x080 – 保留旧值（保留计数器原来的值，通常用于帮助下一个观察周期内的计算） • 0x100 – 内部使用，不显示 • 0x200 – 观测器 • 0x400 – 监视器 • 0x800 – 内部使用，不显示
description	varchar(255)	计数器说明

索引

(*counter_id*) 上的唯一聚簇键 *rs_key_statcounters*

rs_statdetail

存储已经刷新到 RSSD 的计数器指标。

列	数据类型	说明
run_id	*rs_id*	分配给运行或观测周期的编号
instance_id	*int*	标识模块实例的 ID。 计数器按模块分组。一个模块可能有一个或多个实例。适当的时候使用定义的模块 ID。例如，某个 DSI 模块的 *instance_id* 就是与此 DSI 关联的数据库 ID。
instance_val	*int*	当 *instance_id* 不能唯一地标识模块实例时，可以使用此 ID 来标识。
counter_id	int	唯一计数器标识号
counter_obs	int	观测次数
counter_total	int	运行或观测周期内的观测值总和
counter_last	int	运行或观测周期内的最后观测值
counter_max	int	运行或观测周期内的最大观测值
label	*varchar(255)*	与计数器相关联的模块实例的描述性信息，例如，数据服务器名和数据库名称。

索引

（*run_id*、*instance_id*、*instance_val* 和 *counter_id*）上的唯一非聚簇键 *rs_key_statdetail*

rs_statrun

存储有关每个观测周期或运行的描述性信息。

列	数据类型	说明
run_id	*rs_id*	分配给观测周期或运行的编号
run_date	datetime	观测周期或运行的日期和时间

列	数据类型	说明
run_interval	int	观测周期或运行的持续时间，以秒为单位
run_user	varchar(30)	将计数器刷新到 RSSD 的用户的名称
run_status	*int*	运行的状态

索引

(*run_id*) 上的唯一非聚簇键 *rs_key_statdetail*

rs_status

存储有关 Replication Server 和 Sybase IQ InfoPrimer 集成期间实现进度的信息。

rs_status 表存储在每个 Sybase IQ 用户数据库中，而不是在 RSSD 中。

列	数据类型	说明
schema	varchar (255)	要实现的表的所有者
tablename	varchar (255)	要实现的表的名称
action	varchar (1)	• I – 初始装载 • A – 自动更正阶段 • R – 复制
starttime	timestamp	操作开始的时间
endtime	timestamp	操作结束的时间
status	varchar (1)	• P – 正在进行操作 • X – 执行完成 • E – 执行错误
pid	int	保留

rs_subscriptions

存储有关预订、触发器和段的信息。

列	数据类型	说明
subname	*varchar(255)*	预订、触发器或段的名称。
subid	*rs_id*	此预订或段的 ID。
type	*int*	对象类型： • 0x00 – 预订 • 0x01 – 范围预订 • 0x02 – 等同性预订 • 0x04 – 整个表 • 0x08 – 发布预订 • 0x40 – 数据库预订 • 0x80 – 项目预订
objid	*rs_id*	表复制定义 ID、函数复制定义 ID、项目 ID 或该预订的发布 ID。或者为段的 ID 或该触发器的事件的 ID。
dbid	*int*	此对象所属数据库的 ID。
pdbid	*int*	对于系统表复制和发布或项目预订，*pdbid* 值是复制定义的主数据库的 ID。否则，值为 0。
requestdate	*datetime*	输入上一个 DDL 请求（**create**、**drop**、**alter**）的日期和时间。
pownerid	*rs_id*	主 Replication Server 中的用户 ID。
rownerid	*rs_id*	复制 Replication Server 中的用户 ID。

列	数据类型	说明

status	int	• 第 1 个字节包含复制数据库的实现状态：
		  • 0x01 – 预订是新建的
		  • 0x02 – 批量预订正在激活或基本/非基本预订已经完成实现队列的生成
		  • 0x04 – 批量/非基本预订处于活动状态
		  • 0x08 – 批量预订正在验证或非基本预订已实现
		  • 0x10 – 预订有效
		  • 0x40 – 预订在备份数据库上有效
		  • 0x40 – 在备用数据库中已删除预订
		  • 0x80000000 -- 数据库预订正在使用 **dump marker** 协调实现
		• 第 2 个字节包含主数据库取消实现状态：
		  • 0x100 – 新的
		  • 0x0200 – 正在激活
		  • 0x0400 – 活动
		  • 0x0800 – 有效
		• 第 3 个字节包含复制数据库的取消实现状态：
		  • 0x00010000 – 正在复制数据库中取消实现
		  • 0x00020000 – 正在复制数据库上删除
		  • 0x00100000 – 正在主数据库上取消实现
		• 第 4 个字节包含发布预订的可疑状态或重新实现状态：
		  • 0x02000000 – 由于切换活动状态而可疑
		  • 0x04000000 – 在备份数据库上删除时可疑
		  • 0x10000000 – 此发布预订内的项目预订一次实现一个
		  • 0x20000000 – 正在创建新的项目预订
		  • 0x40000000 – 包括 **truncate table**

recovering	*int*	预订恢复状态： • 0x0 – 预订状态良好 • 0x1 – 正在恢复 • 0x2 – 未决
error_flag	*int*	如果设置，则预订不可恢复
materializing	*int*	如果设置，则预订正在实现
dematerializing	*int*	如果设置，则预订正在取消实现

列	数据类型	说明
primary_sre	*int*	如果设置，则预订应包括在主 Replication Server 上的预订解析引擎中
replicate_sre	*int*	如果设置，则预订应包括在复制 Replication Server 上的预订解析引擎中
materialization_try	*int*	此原子实现已经尝试的次数
method	*int*	实现预订的方法： • 0x00 – 缺省方法 • 0x01 – 基本 • 0x02 – 批量 • 0x04 – 挂起 • 0x08 – 增量 • 0x10 – 非基本 • 0x80 – 在挂起备用 DSI 的情况下批量实现 **注意：** 对于函数复制定义，此列始终设置为 0x02（批量）
generation	*binary*	实现队列的原始队列 ID 的生成号
parentid	*rs_id*	如果当前预订是针对项目的，则为发布的预订 ID。
安全性	*int*	安全性设置： • 0x001 – unified_login 设置为“required” • 0x002 – mutual_auth 设置为“required” • 0x004 – msg_confidentiality 设置为“required” • 0x08 – msg_integrity 设置为“required” • 0x10 – msg_origin_check 设置为“required” • 0x20 – msg_reply_detection 设置为“required” • 0x40 – msg_sequence_check 设置为“required” 缺省值：0
mechanism	*char(30)*	安全性机制的名称 缺省值：空值

索引

- (*subid*) 的唯一聚簇索引
- （*objid*、*dbid* 和 *subname*）的唯一索引

- （*subid*、*recovering*、*error_flag*、*materializing*、*dematerializing*、*primary_sre* 和 *replicate_sre*）的唯一索引
- （*subid*、*status*）的唯一索引
- (*objid*) 的唯一索引
- (*pdbid*) 的唯一索引

rs_systext

为各种其它表存储重复组的文本，如 *rs_funcstrings*。

列	数据类型	说明
prsid	*int*	定义此对象时所在的 Replication Server
parentid	*rs_id*	此文本对应对象的 ID
texttype	*char(1)*	此行对应对象的类型： • S – 函数字符串的输入模板 • O – 函数字符串的输出模板 • C – 例外日志中已记录事务的命令 • P – Replication Server 配置文件
sequence	*int*	文本的序列
textval	*varchar(255)*	文本

索引

（*parentid*、*texttype* 和 *sequence*）的唯一聚簇索引

rs_targetobjs

存储目标表或存储过程的信息。

列	数据类型	说明
dbid	*int*	数据库的唯一标识符。
objname	*varchar(255)*	表或存储过程的名称。
objowner	*varchar(30)*	复制对象名。
objid	*rs_id*	对象 ID。

列	数据类型	说明
objtype	*char(1)*	以下对象类型之一： • S – 存储过程。 • T – 表。
attributes	*int*	此列显示： • 0x01 - 具有用于 **rs_writetext** 的自定义函数字符串。 • 0x02 - 具有用于 **rs_textptr_init** 的自定义函数字符串。 • 0x04 - 具有用于 **rs_get_textptr** 的自定义函数字符串。

索引

- （dbid、objname、objowner、objtype）的唯一索引
- (objid) 的唯一索引

rs_tbconfig

Replication Server 使用 *rs_tbconfig* 表中的信息支持参照约束。

rs_tbconfig 不是复制的系统表。

列	数据类型	说明
optionname	*varchar(30)*	参数的名称，例如：**memory_max**、**cm_max_connections** 要查看这些参数及其说明的列表，请对 *rs_tbconfig* 表执行 **select *** 语句。
dbid	*int*	数据库的唯一标识符。
objname	*varchar(255)*	复制数据库中定义的对象名。
objowner	*varchar(30)*	复制定义中所指定的复制对象所有者的名称。 如果未指定所有者，则为空。
charvalue	*varchar(255)*	参数的字符值。
status	*tinyint*	不使用此列。
comments	*varchar(255)*	有关参数的注释。

索引

（*optionname*、*dbid*、*objname*、*objowner*）的唯一聚簇索引

rs_threads

Replication Server 使用 *rs_threads* 表中的信息来检测死锁并在并行 DSI 线程之间执行事务序列化。每次启动某个事务以及为一个连接定义了多个 DSI 线程时，都会更新此表中的一个条目。

rs_threads 表存储在各个用户数据库中，而不是在 RSSD 中。

列	数据类型	说明
id	*int*	条目 ID 号。每个并行 DSI 线程有两个条目。
seq	*int*	上一次对此条目进行更新的序列号。每次重新启动连接时，序列号都从 0 开始。
pad1	*char(255)*	填充内容，用于填充行以使每个数据库页只容纳一行。
pad2	*char(255)*	填充内容，用于填充行以使每个数据库页只容纳一行。
pad3	*char(255)*	填充内容，用于填充行以使每个数据库页只容纳一行。
pad4	*char(255)*	填充内容，用于填充行以使每个数据库页只容纳一行。

索引

(*id*) 的唯一聚簇索引

rs_ticket_history

存储 **rs_ticket** 信息。

列	数据类型	说明
cnt	*int identity*	票据的唯一序列。
h1	*varchar(10)*	票据头。如果票据头不存在，则设置为“-”。
h2	*varchar(10)*	票据头。如果票据头不存在，则设置为“-”。
h3	*varchar(10)*	票据头。如果票据头不存在，则设置为“-”。
h4	*varchar(50)*	票据头。如果票据头不存在，则设置为“-”。
pdb	*varchar(30)*	主数据库名称。
prs	*varchar(30)*	主 Replication Server 名。如果未指定主 Replication Server，则设置为“-”。

列	数据类型	说明
rrs	*varchar(30)*	复制 Replication Server 名。
rdb	*varchar(30)*	复制数据库名称。
pdb_t	*datetime*	在主数据库中执行 **rs_ticket** 存储过程的时间。
exec_t	*datetime*	通过 Replication Server 执行程序线程传递票据的时间。
dist_t	*datetime*	票据通过 DIST 线程传递的时间。
rsi_t	*datetime*	票据通过 RSI 线程传递的时间。
dsi_t	*datetime*	票据通过 DSI 线程传递的时间。
rdb_t	*datetime*	票据到达复制数据库的时间。
exec_b	*int*	EXEC 线程收到的总字节数。
rsi_b	*int*	RSI 线程收到的总字节数。
dsi_tnx	*int*	DSI 观察到的事务总数。
dsi_cmd	*int*	DSI 观察到的命令总数。
ticket	*varchar(1024)*	原始票据。

索引

rs_ticket_history(*cnt*) 的唯一聚簇索引 *rs_ticket_idx*

rs_translation

存储有关类级数据类型转换的信息。

列	数据类型	说明
prsid	*int*	主 Replication Server 的 ID
classid	*rs_id*	连接的函数字符串类 ID
type	*char(1)*	转换类型。可以是: • D – 类级
source_dtid	*rs_id*	源数据类型的 ID
target_dtid	*rs_id*	目标数据类型的 ID

列	数据类型	说明
target_length	*int*	目标数据类型的值的最大长度
target_status	*int*	请参见 *rs_columns* 表中的 *status* 列
rowtype	*tinyint*	指示某行是在 Replication Server 本地还是分布到域中的所有 Replication Server。可以是: • 0 – 本地 • 1 – 全局

索引

- (*classid*、*source_dtid*、*target_status*) 的唯一复合索引
- (*classid*、*prsid*) 的非唯一索引

rs_users

为有权访问 Replication Server 的每个用户存储一行信息。

列	数据类型	说明
username	*varchar(30)*	ユーザ名。
uid	*rs_id*	用户 ID。
attributes	*int*	帐户和口令的状态及设置: • 0x000 是缺省值。 • 0x0001 – 初始口令。 • 0x0002 – 帐户已锁定。 • 0x0004 – 最大失败尝试锁定次数。 • 0x0008 – 必须对此登录名重置口令。 • 0x0010 – 口令不过期。 • 0x0020 – 登录名未使用。 属性值取决于用户和授予的权限。0x0000 是有效值。
expiration_interval	*smallint*	用户的口令有效期，以天数为单位。
failed_attempts	*smallint*	失败的登录尝试次数。
lock_date	*datetime*	帐户被锁定的日期。
last_login	*datetime*	上次登录日期。

列	数据类型	说明
password	*varchar(30)*	口令。
password_date	*datetime*	用户的上次口令更改日期。
权限	*smallint*	屏蔽，指示用户可以充当的角色： • 0x0001 – **sa**。 • 0x0002 – **connect source**。 • 0x0004 – **create object**。 • 0x0008 – **primary subscribe**。
use_enc_password	*int*	• 0 – 使用普通口令。 • 1 – 使用加密口令。
enc_password	*varchar(66)*	加密口令。

索引

- (*username*) 的唯一索引
- (*uid*) 的唯一索引

rs_version

为复制系统存储版本号信息。在本地 Replication Server 上，仅存储本地版本号和系统范围的版本号。在 ID Server 上，存储复制系统中所有 Replication Server 的版本号。

列	数据类型	说明
siteid	*int*	Replication Server 的 ID 号： • 0 – 系统范围版本号的节点 ID • 1 – 节点版本号的节点 ID • *n* – 各个 Replication Server 的节点 ID

列	数据类型	说明
version	*int*	版本号 • 1000 – 版本 10.0（指派给版本不明的任何 Replication Server） • 1003 – 版本 10.0.3 • 1011 – 版本 10.1.1 • 1100 – 版本 11.0 • 1101 – 版本 11.0.1 • 1102 – 版本 11.0.2 • 1103 – 版本 11.0.3 • 1150 – 版本 11.5 • 1200 – 版本 12.0 • 1210 – 版本 12.1 • 1250 – 版本 12.5 • 1260 – 版本 12.6 • 1500 – 版本 15.0、15.0.1 • 1510 – 版本 15.1 • 1520 – 版本 15.2 • 1550 – 版本 15.5 有关支持的任何其它版本号，请参见《Replication Server 发行公告》。

有关系统范围的版本号的详细信息，参见 **admin security_property**。

索引

(*siteid*) 的唯一聚簇索引

rs_whereclauses

存储此 Replication Server 的已知项目中使用的 **where** 子句的相关信息。

列	数据类型	说明
articleid	*rs_id*	此 **where** 子句中包含的项目的 ID
wclauseid	*rs_id*	此 **where** 子句的 ID
type	*int*	• 0x01 – 范围 • 0x02 – 等同

索引

(*wclauseid*) 的唯一聚簇索引

Replication Monitoring Services API

列出 Replication Monitor Service (RMS) API 命令。

表 54. RMS API 命令

命令	说明
add event trigger （第 633 页）	设置在发生特定事件时 RMS 执行的触发器，如过程或脚本。
add server （第 636 页）	添加 RMS 所监控的服务器。
configure component （第 638 页）	返回组件的配置参数，或设置指定的配置参数的值。组件是服务器（包括 Replication Server 和 Adaptive Server Enterprise）中的监控对象。
configure RMS （第 640 页）	返回 RMS 的配置参数信息，或设置指定的 RMS 配置参数的值。
configure server （第 642 页）	返回 Replication Server 或 Replication Agent 的配置参数信息，或设置指定的配置参数的值。还会检索和设置特定于 RMS 的参数。
connect to server （第 644 页）	提供传递模式，以使您能够将命令发送到 RMS 所监控的服务器。命令生成的结果集将传回到客户端。
create group （第 645 页）	使您能够定义一组服务器，并向所有组成员发出命令。
delete group （第 646 页）	删除使用 **create group** 命令添加的逻辑组。
disconnect server （第 646 页）	与建立了传递连接的服务器断开连接。
drop event trigger （第 647 页）	删除 RMS 所监控的触发器（使用 **add event triggers** 命令添加的）。
drop server （第 649 页）	删除 RMS 所监控的服务器。
filter connection （第 650 页）	返回连接的当前过滤器设置，或者为其设置过滤器设置。此命令可以过滤 Replication Agent 线程或 DSI 线程状态。
get component （第 651 页）	返回 RMS 所监控的 Replication Server 或 Adaptive Server Enterprise 组件的列表。组件是服务器中的监控对象。
get group （第 653 页）	返回一个结果集，其中包含组列表和每个组的累积状态，或者包含每个服务器的状态和指定组的累积状态。累积状态显示为组中的组件报告的最低状态。

命令	说明
get heartbeat （第 653 页）	检索 RMS 中已定义的心跳进程的列表。
get heartbeat tickets （第 656 页）	从 *rms_ticket_history* 表中，为指定的心跳进程、日期和时间范围检索一组票据。
get network spec （第 658 页）	检索 RMS 已知的所有服务器的连接信息。此列表是从 RMS 的 `interfaces` 文件或 LDAP 服务器中检索的。该列表包含服务器名、主机名以及服务器使用的端口号。
get rmiaddress （第 659 页）	检索远程方法调用 (RMI) 服务的地址。
get servers （第 660 页）	返回 RMS 所监控的服务器的列表以及 RMS 环境状态。RMS 状态是所监控的服务器的累积状态。
get status descriptions （第 661 页）	检索服务器或组件的状态说明的列表。
get threads （第 663 页）	显示有关 Replication Server 中正在运行的线程的信息。
get triggers （第 663 页）	显示 RMS 所监控的触发器的相关信息。
get version （第 665 页）	检索 RMS 的版本号。
log level （第 666 页）	返回当前日志级别设置。**log level** 也可更改 RMS 的日志级别设置。
resume component （第 666 页）	恢复指定服务器中的组件。此命令恢复 Replication Server 中的 DSI 线程、Replication Agent 线程、RepAgent 线程、队列或路由。
resume Replication Agent （第 668 页）	在 Replication Agent 中恢复复制。
shutdown server （第 668 页）	向服务器或 RMS 发出 **shutdown** 命令。
suspend component （第 671 页）	挂起指定服务器中的组件。此命令挂起 Replication Server 中的 DSI 线程、Replication Agent 线程、RepAgent 线程或路由。
start heartbeat （第 669 页）	设置并启动从指定主连接到指定复制连接的心跳进程。
stop heartbeat （第 670 页）	停止主数据库和复制数据库之间的心跳进程。可以选择截断 *rms_ticket_history* 表。
suspend Replication Agent （第 671 页）	在 Replication Agent 中挂起复制。
trace （第 673 页）	在 RMS 日志文件中显示跟踪信息。

要使用 RMS API 命令，必须为 RMS 所监控的每个服务器设置以下权限：

服务器	权限
Adaptive Server	对于任何主数据库，用户必须具有“sa”或“dbo”权限或Replication角色。对于任何 RSSD 数据库，用户必须具有“sa”或“dbo”权限。
Replication Server	用户必须具有“sa”权限。
Replication Agent	此服务器没有其它的用户权限。
镜像 Replication Agent	此服务器没有其它的用户权限。
DirectConnect™	用户必须具有成功登录到后端服务器上的权限。RMS 并不尝试读取或写入后端数据库。
SA	用户必须具有登录到 SA 数据库的权限。RMS 并不尝试读取或写入该数据库。
IQ	用户必须具有登录到 IQ 服务器上的权限。RMS 并不尝试读取或写入该数据库。
远程 RMS	此服务器没有其它的用户权限。
Open Server	用户必须具有建立到 Open Server 的连接的权限。

add event trigger

添加在复制域中发生特定事件时 RMS 执行的触发器。触发器能够识别 RMS 执行的进程或脚本。

语法

```
add {status | latency | size} trigger
      [{connection | logical connection | route | queue | rep agent |
        partition} [component_name]]
       [with primary primary_connection]
       for server_name
       {status changes to state |
         size {exceeds | falls below} size_threshold |
         latency {exceeds | falls below} latency_threshold}
       [wait wait_interval]
       [continuous continuous_flag]
       execute command
```

参数

- **status、latency、size** – 触发器类型。
- **connection、logical connection、route、queue、rep agent、partition** – 指定要监控的组件的类型。组件是服务器中的监控对象。Replication Server 组件是指连接、

逻辑连接、路由、队列和分区；Adaptive Server Enterprise 组件是指 RepAgent 线程。

- **component_name** – 指定要监控的组件的名称。
- **with primary primary_connection** – 指定连接延迟触发器的主连接。如果不满足主连接和复制连接之间延迟阈值的要求，此触发器将执行脚本。
- **for server_name** – 指定要监控的服务器的名称。如果此命令用于为组件添加触发器，则服务器是组件所有者。
- **size exceeds/falls below size_threshold** – 指示应在大小超过阈值时执行触发器，还是在低于阈值时执行触发器。
- **latency exceeds/falls below latency_threshold** – 指示应在延迟超过阈值时执行触发器，还是在低于阈值时执行触发器。
- **status changes to state** – 指定要监控的服务器或组件的状态。如果 *state* 更改为指定的值，则会执行触发器。状态值取决于对象类型。有关状态代码的信息，请参见“RMS 服务器和组件状态”。
- **wait wait_interval** – 指定在触发事件之前等待的秒数。这为对象恢复留出了时间。如果没有包含 **wait** 选项，则会立即触发事件。
- **continuous continuous_flag** – 一个布尔标志，如果设置为 true，将使 RMS 在每个后续监控间隔执行触发器脚本，直到状态发生变化。如果没有设置此标志，RMS 仅执行一次触发器脚本。
- **execute command** – 指定在触发事件时执行的命令。此命令因操作系统而异。

示例

- **示例 1** – 添加一个触发器，当名为 INVENTORY_RS 的服务器的状态更改为“DOWN”时，它将执行脚本 `email.sh`：

```
add status trigger for INVENTORY_RS
status changes to DOWN
execute /sybase/RMS/scripts/email.sh
```

- **示例 2** – 添加一个触发器，它将在 120 秒后执行脚本 `email.sh`。由于 INVENTORY_RS 服务器的“inventory_pds.pdb1”连接的状态更改为“SUSPENDED”，因此该触发器将在每个后续监控间隔执行脚本，直到状态发生变化：

```
add status trigger connection inventory_pds.pdb1 for
    INVENTORY_RS
status changes to Suspended
wait 120
continuous true
execute /sybase/RMS/scripts/email.sh
```

- **示例 3** – 在 Replication Server INVENTORY_RS 分区“p1”中添加一个触发器，当分区使用率超过 80% 时，它将执行脚本 `email.sh`。只要分区使用率超过 80%，就会在每个后续监控间隔执行该脚本：

```
add size trigger partition p1 for INVENTORY_RS
    size exceeds 80
```

```
    continuous true
    execute /sybase/RMS/scripts/email.sh
```

- **示例 4** – 在 Replication Server INVENTORY_RS 中添加一个触发器，当分区总使用率超过 75% 时，它将执行脚本 `email.sh`：

```
add size trigger partition for INVENTORY_RS
    size exceeds 75
    execute /sybase/RMS/scripts/email.sh
```

- **示例 5** – 在 Replication Server INVENTORY_RS 的队列“inventory_pds.vendor(Inbound)”中添加一个触发器，以在队列大小低于 100 MB 时执行脚本 `email.sh`。只要队列大小低于 100 MB，就会在每个后续监控间隔执行该脚本：

```
add size trigger queue inventory_pds.vendor(Inbound)
    for INVENTORY_RS
    size falls below 100
    continuous true
    execute /sybase/RMS/scripts/email.sh
```

- **示例 6** – 在复制 Replication Server INVENTORY_RS 的复制连接“inventory_rds.vendor”中添加一个触发器，以在主连接“inventory_pds.vendor”的延迟时间超过 5 分钟（300 秒）时执行脚本 `email.sh`：

```
add latency trigger connection inventory_rds.vendor
    with primary inventory_pds.vendor
    for INVENTORY_RS
    latency exceeds 300
    execute /sybase/RMS/scripts/email.sh
```

用法

- 您可以为每个服务器或组件状态添加一个状态触发器。例如，当状态更改为“DOWN”或“SUSPECT”时，可以在 Replication Server 中添加一个触发器，但不能为“DOWN”状态添加两个触发器。
- 添加延迟连接触发器时，必须将 *server_name* 设置为复制 Replication Server 的名称。在此示例中，INVENTORY_RS 是复制 Replication Server：

```
add latency trigger connection inventory_rds.vendor
    with primary invetory_pds.vendor
    for INVENTORY_RS
    latency exceeds 300
     execute /sybase/RMS/scripts/email.sh
```

- 在设置主连接来自 Replication Agent 或 MRA 的延迟连接触发器时，必须将配置参数 **ltl_origin_time_require** 设置为“true”。要设置此参数，请连接到 Replication Agent 或 MRA 并执行以下命令：

```
ra_config ltl_origin_time_required, true
```

- **add** *event* **trigger** 返回以下结果集：

表 55. `add event trigger` 的列说明

列	说明
Action	操作名称
Result	执行结果

另请参见

- drop event trigger （第 647 页）
- get triggers （第 663 页）

add server

添加 RMS 所监控的服务器。

语法

```
add {ASA | ASE | DirectConnect | IQ | Replication Agent | MRA |
Replication Server | RMS | Open Server | dbltm} server_name
              set username [to] user
              [set password [to] passwd]
              [set charset [to] charset]
              [set language [to] lang]
              [set rssd_username [to] rssd_user]
              [set rssd_password [to] rssd_passwd]
              [set rssd_charset [to] rssd_charset]
              [set rssd_language [to] rssd_lang]
              [set monitoring [to] {'true' | 'false'}]
              [set interval [to] interval]
              [set connection_ds [to] ds]
              [set connection_db [to] db]
```

参数

- **ASA、ASE、DirectConnect、IQ、Replication Agent、MRA、Replication Server、RMS、Open Server、dbltm** – 指定要添加到 RMS 中的服务器的类型。您可以将远程 RMS 添加到进行控制的 RMS 中。
- **server_name** – 指定 RMS `interfaces` 文件或 LDAP 服务器中列出的服务器的名称。
- **user** – 指定在建立到服务器的连接时 RMS 使用的用户名。用户名必须具有使 RMS 能够监控服务器所需的权限。
- **passwd** – 指定在建立连接时 RMS 使用的相应口令。

  注意： 如果口令为空，则不要包括 **set password** 子句。
- **charset** – 指定在建立到服务器的连接时 RMS 使用的字符集。如果未指定 *charset*，jConnect 将使用服务器的缺省字符集。

- **lang** – 指定在建立到服务器的连接时 RMS 使用的语言。如果未指定语言，jConnect 将使用服务器的缺省语言。
- **rssd_user** – 指定在建立到包含 RSSD 的服务器的连接时 RMS 使用的用户名。用户名必须具有使 RMS 能够监控服务器所需的权限。对于 Replication Server，此参数是必需的。
- **rssd_passwd** – 指定在建立到包含 RSSD 的服务器的连接时 RMS 使用的相应口令。
- **rssd_charset** – 指定在建立到包含 RSSD 的服务器的连接时 RMS 使用的字符集。如果未提供 *charset*，jConnect 将使用服务器的缺省字符集。
- **rssd_lang** – 指定在建立到包含 RSSD 的服务器的连接时 RMS 使用的语言。如果未提供语言，jConnect 将使用服务器的缺省语言。
- **监控** – 指定 RMS 是否监控服务器及其组件的状态。如果此值为 false，则禁用对此服务器的监控。如果此值为 true（缺省值），RMS 将自动监控此服务器。
- **interval** – 指定监控周期之间间隔的秒数。如果将 monitoring 属性设置为 true，RMS 将根据 *interval* 的值定期执行监控。例如，如果将该值设置为 120，RMS 每隔 120 秒检查一次服务器的运行状况。值范围是 30 秒到 1 小时，interval 的缺省值是 RMS 配置中 *ping_interval* 的值。
- **ds** – 指定主数据服务器的名称。在复制事务时，dbltm 会将 *ds.db* 发送到 Replication Server。*ds* 必须与 Replication Server 连接中使用的服务器名称相匹配。此参数是可选的，并且仅适用于 dbltm 服务器。
- **db** – 指定主数据库的名称。在复制事务时，dbltm 服务器会将 *ds.db* 发送到 Replication Server。*db* 必须与 Replication Server 连接中使用的数据库名称相匹配。此参数是可选的，并且仅适用于 dbltm 服务器。

示例

- **示例 1** – 将名为 INVENTORY_RS 的 Replication Server 添加到 RMS 中。在建立连接时，将使用用户名“sa”，而不设置口令、字符集或语言。在建立到 RSSD 的连接时，将使用用户名“sa”和口令“sa_pwd”：

```
add replication server INVENTORY_RS
    set username to sa
    set rssd_username to sa
    set rssd_password to sa_pwd
```

- **示例 2** – 将名为 INVENTORY_PDS 的服务器添加到 RMS 中，并设置用户名、口令、语言、监控和间隔：

```
add ASE INVENTORY_PDS
    set username to sa
    set password to sa_ps
    set language to Japanese
    set monitoring to true
    set interval to 120
```

用法

- 将 Replication Server 添加到 RMS 中时使用 RSSD 选项。不需要将包含 RSSD 的服务器添加到 RMS 中。
- 服务器名必须位于 RMS 使用的 `interfaces` 文件或 LDAP 服务器中。
- 发出 **add** *server* 时，RMS 将尝试连接到指定的服务器，并自动确定其类型和版本。如果类型或版本无效、无法确定类型或版本或者已经在监控该服务器，RMS 将返回一条错误消息。
- 如果新服务器为 Replication Server，应提供 RSSD 用户名。
- **add** *server* 命令返回以下结果集：

表 56. `add server` 的列说明

列	说明
Action	操作名称
Result	执行结果

另请参见

- configure server （第 642 页）
- connect to server （第 644 页）
- disconnect server （第 646 页）
- drop server （第 649 页）
- get servers （第 660 页）
- shutdown server （第 668 页）

configure component

返回 Replication Server 或 Adaptive Server 中的组件的配置参数信息，或设置指定的配置参数的值。组件是服务器中的监控对象。Replication Server 组件是指连接、逻辑连接和路由；Adaptive Server Enterprise 组件是指 RepAgent 线程。

语法

```
configure {connections | logical connections | routes | repagents}
component_name
    [for] {server_name | group_name} [param[= value]]
```

参数

- **connections、logical connections、routes、repagents** – 指定要配置的组件的类型。Replication Server 组件是指连接、逻辑连接和路由；Adaptive Server Enterprise 组件是指 RepAgent 线程。

- **component_name** – 指定要配置的组件的名称。
- **server_name** – 指定包含所请求的组件的服务器。
- **group_name** – 指定组名。可以针对组中的每个不同的组件修改 *group_name* 参数。
- **param** – 指定组件的配置参数名称。
- **value** – 要为 *param* 选项中指定的配置参数指派的值。

示例

- **示例 1** – 返回 INVENTORY_RS 服务器中的“inventory_pds.vendor”连接的所有配置参数列表：

```
configure connection inventory_pds.vendor
    for INVENTORY_RS
```

- **示例 2** – 返回服务器 INVENTORY_RS 中连接“inventory_pds.vendor”的 **dsi_cmd_batch_size** 配置参数信息：

```
configure connection inventory_pds.vendor
    for INVENTORY_RS dsi_cmd_batch_size
```

- **示例 3** – 将服务器 INVENTORY_RS 中连接“inventory_pds.vendor”的 **dsi_cmd_batch_size** 配置参数设置为 15000：

```
configure connection inventory_pds.vendor
    for inventory_rs dsi_cmd_batch_size = 15000
```

用法

如果不包含 *value* 参数，**configure** *component* 将返回以下结果集：

表 57. `configure component` 的列说明

列	说明
服务器	包含参数的服务器的名称。
Component Name	包含参数的组件的名称。
Component Type	组件类型（连接、路由或 RepAgent）。
Category	参数类别的名称。类别用于将相关参数组合在一起。
Parameter Name	参数名。
Current Value	参数的当前值。
Pending Value	在重新启动组件后，待定值将变为参数值。
Default Value	参数的缺省值。
Legal Values	定义参数合法值的字符串。它可以是列表，也可以是数值范围。

列	说明
Restart Required	指示是否必须重新启动服务器才能使参数生效的标志。

另请参见

- get component （第 651 页）
- resume component （第 666 页）
- suspend component （第 671 页）

configure RMS

返回 Replication Monitoring Services 的配置参数信息，或设置指定的 RMS 配置参数的值。

语法

```
configure [param [= value]]
```

参数

- **param** – 指定 RMS 配置参数的名称。
- **value** – 要为 *param* 选项中指定的配置参数指派的值。

表 58. RMS 参数

参数	值
Logconfig	RMS 日志配置文件的路径。
Name	RMS 服务器的名称。此名称必须出现在 Sybase interfaces 文件中。
Password	用于连接到 RMS 的口令。**configure** 命令不显示此参数的值。
ping_interval	一个监控周期结束到下一个监控周期开始之间间隔的秒数。其范围从 30 秒到 3600 秒。
Port	RMS 使用的 IP 端口。其范围从 1024 到 65,535。
SybaseHome	Sybase 主目录。此目录包含 interfaces 文件。
Username	用于连接到 RMS 的用户名。
Version	RMS 的版本字符串。这是只读参数。
includeLDAP	打开或关闭 LDAP 支持的标志。
ldapTimeout	用户可以配置的超时值。

示例

- **示例 1** – 返回 RMS 配置参数及其当前值的列表，格式如下：

```
configure
```

```
Parameter Name Parameter Type Current Value
-------------- -------------- ------------------
includeldap    boolean        false
ldaptimeout    integer        35
logconfig      string         ../plugins/
                              com.sybase.rms/
                              log4j.properties
name           string         RedtailRMS

Pending Value Default Value          Legal Values
------------- ---------------------- ----------------
NULL          false                  List: true,false
NULL          180                    N/A
N/A           ../log4j.properties    N/A
NULL          Rms                    N/A

Category  Restart Required
--------  -----------------
Rms       false
Rms       false
Rms       N/A
Rms       true

Description
-----------------------------------------
A flag that turns LDAP support on or off.
A user configurable timeout value.
The path to the RMS log config file.
The name of the RMS server.

...
```

- **示例 2** – 为 RMS 配置用户名“sa”：

```
configure username=sa
```

用法

如果没有包含值参数，**configure** *RMS* 命令将返回以下结果集：

表 59. 缺省 RMS 结果集

列	说明
Parameter Name	参数名称，如 `logconfig`、`name`、`port` 和 `password`。
Parameter Type	参数类型，如 `boolean`、`integer`、`string` 和 `password`。
Current Value	参数的当前值。

列	说明
Pending Value	重新启动服务器后的参数值。
Default Value	参数的缺省值。
Legal Values	定义参数合法值的字符串。它可以是列表，也可以是数值范围。
Category	参数类别的名称。可以使用类别将相关参数组合在一起。
Restart Required	指示是否必须重新启动服务器才能使参数生效的标志。
Description	参数说明。

另请参见

- get version （第 665 页）
- resume Replication Agent （第 668 页）
- suspend Replication Agent （第 673 页）
- trace （第 673 页）

configure server

返回 Replication Server 或 Replication Agent 和 Mirror Replication Agent (MRA) 的配置参数信息，或设置指定的配置参数的值。还会检索和设置特定于 RMS 的参数。

语法

```
configure server {server_name | group_name} [RMS] [param [= value]]
```

参数

- **server_name** – 指定要配置的服务器。
- **group_name** – 指定组名。可以针对组中的每个服务器修改 *group_name*。
- **RMS** – 指定 RMS 参数。
- **param** – 指定服务器的配置参数名称。
- **value** – 为 *param* 选项中指定的配置参数指派的值。

示例

- **示例 1** – 返回服务器 INVENTORY_RS 的所有配置参数的列表：

  ```
  configure server INVENTORY_RS
  ```

- **示例 2** – 返回服务器 INVENTORY_RS 的 *memory_limit* 配置参数信息：

  ```
  configure server INVENTORY_RS memory_limit
  ```

  ```
  Parameter     Parameter  Current  Pending  Default
  Name          Type       Value    Value    Value
  ```

```
------------  ---------  -------  -------  -------
memory_limit  NULL       20       55       NULL

Legal   Category  Restart    Description
Values            Required
------  --------  --------   -------------------------
NULL    NULL       NULL        NULL
```

- **示例 3** – 将服务器 INVENTORY_RS 的 *memory_limit* 配置参数设置为 50:

```
configure server inventory_rs memory_limit = 50
```

- **示例 4** – 检索所有特定于 RMS 的参数:

```
configure server INVENTORY_RS RMS
```

- **示例 5** – 更改 RMS 用于连接到服务器的用户名:

```
configure server INVENTORY_RS RMS username =  'rsa'
```

用法

- **configure server** 支持 Replication Server、Replication Agent 以及远程监控的 RMS 配置。
- **configure server** 可以为所有类型的服务器检索和设置特定于 RMS 的参数。服务器和 RMS 使用这些参数进行通信。
- 如果没有包含值参数，**configure server** 将返回以下结果集:

表 60. 缺省 `configure server` 结果集

列	说明
Parameter Name	参数名。
Parameter Type	参数类型。
Current Value	参数的当前值。
Pending Value	在重新启动服务器后，待定值将变为参数值。
Default Value	参数的缺省值。
Legal Values	定义参数合法值的字符串。它可以是列表，也可以是数值范围。
Category	参数类别的名称。类别用于将相关参数组合在一起。
Restart Required	指示是否必须重新启动服务器才能使参数生效的标志。
Description	参数说明。

另请参见

- add server （第 636 页）
- connect to server （第 644 页）
- disconnect server （第 646 页）

- drop server （第 649 页）
- get servers （第 660 页）
- shutdown server （第 668 页）

connect to server

提供传递模式，以使您能够将命令发送到 RMS 所监控的服务器。命令生成的结果集将传回到客户端。一次只能连接到一个服务器以发送命令。

语法

```
connect [to] server_name [username=username [,password = pwd]]
```

参数

- **server_name** – 指定要连接到的服务器的名称。
- **username** – 可选参数，指定在连接到服务器时使用的用户名。如果省略此参数，RMS 将使用添加服务器时使用的名称。
- **pwd** – 与用户名关联的口令。

示例

- **示例 1** – 建立到服务器 INVENTORY_RS 的连接：

  ```
  connect to INVENTORY_RS
  ```

用法

- 发出 **connect** 命令可建立到服务器的连接。消息 `Established a connection to the server server_name`（建立了到服务器 server_name 的连接）表示建立了连接。
- 后续命令将直接传递到服务器，直至客户端发出 **disconnect** 命令。请使用适用于服务器的 ISQL 命令；例如，适用于 Adaptive Server Enterprise 的 Transact-SQL 或适用于 Replication Server 的 RCL。

另请参见

- add server （第 636 页）
- configure server （第 642 页）
- disconnect server （第 646 页）
- drop server （第 649 页）
- get servers （第 660 页）
- shutdown server （第 668 页）

create group

定义服务器的逻辑组，以使您能够向组发出命令。

语法

```
create group group_name
    [add] server_name [, server_name]
```

参数

- **group_name** – 指定新组的名称。
- **server_name** – 指定要添加到组中的服务器。

示例

- **示例 1** – 添加一个名为“inventory_mra”的组，其中包含三个镜像 Replication Agent (MRA) 服务器：

```
create group inventory_mra
    add ny_mra, chi_mra, la_mra
```

用法

- 组名必须是唯一的。
- 组中的所有服务器必须为相同类型（即，所有服务器必须为 MRA、Replication Server 等）。
- 服务器可以属于多个组。
- **create group** 返回以下结果集：

表 61. `create group` 的列说明

列	说明
Action	操作名称
Result	执行结果，如 `Successfully created the group group_name`

另请参见

- delete group （第 646 页）
- get group （第 653 页）

delete group

删除使用 **create group** 命令添加的逻辑组。

语法

```
delete group group_name
```

参数

- **group_name** – 指定要删除的组的名称。

示例

- **示例 1** – 删除名为“inventory_mra”的组：

```
delete group inventory_mra
```

用法

- 删除组并不会从 RMS 中删除服务器。
- **delete group** 返回以下结果集：

表 62. `delete group` 的列说明

列	说明
Action	操作名称
Result	执行结果，如 `Successfully dropped the group group_name`

另请参见

- create group （第 645 页）
- get group （第 653 页）

disconnect server

与建立了传递连接的服务器断开连接。客户端可以使用 **connect** 命令通过 RMS 连接到管理的服务器。后续命令将转发到服务器，直至客户端发出 **disconnect** 命令。

语法

```
disconnect
```

示例

- **示例 1** – 从客户端中断开与服务器的连接：

```
disconnect
```

用法

发出 **disconnect** 命令可断开到服务器的连接。消息 `Disconnected from the server servername` 表示连接不再存在。

另请参见

- add server （第 636 页）
- configure server （第 642 页）
- connect to server （第 644 页）
- drop server （第 649 页）
- get servers （第 660 页）
- shutdown server （第 668 页）

drop event trigger

删除 RMS 所监控的触发器。触发器能够识别 RMS 执行的进程或脚本。可以使用 **add trigger** 命令来设置触发器。

语法

```
drop {status | latency | size} trigger
        [{connection | logical connection | route | queue | rep agent
|
            partition} [component_name]]
          [with primary primary_connection]
          for server_name
          {status changes to state |
           size {exceeds | falls below} size_threshold |
           latency {exceeds | falls below} latency_threshold}
```

参数

- **status、latency、size** – 指定触发器的类型。
- **connection、logical connection、route、queue、rep agent、partition** – 指定组件的类型。
- **component_name** – 指定组件的名称。组件是服务器中的监控对象。Replication Server 组件是指连接、逻辑连接、路由、队列和分区；Adaptive Server Enterprise 组件是指 RepAgent 线程。

- **with primary primary_connection** – 指定要删除的延迟连接触发器的主连接。在删除延迟连接时需要使用此参数。
- **server_name** – 指定一个服务器的名称，将要删除为其定义的触发器。
- **state** – 指定要删除的事件触发器的状态。有关状态信息，请参见“RMS 服务器和组件状态”。
- **size exceeds/falls below size_threshold** – 指示要删除的大小触发器。
- **latency exceeds/falls below latency_threshold** – 指示要删除的延迟触发器。

示例

- **示例 1** – 删除 INVENTORY_RS 服务器的“DOWN”状态触发器：

```
drop status trigger for INVENTORY_RS
    status changes to DOWN
```

- **示例 2** – 删除 INVENTORY_RS 服务器的“inventory_pds.pdb1”连接的“SUSPENDED”状态触发器：

```
drop status trigger connection inventory_pds.pdb1
    for inventory_rs
    status changes to SUSPENDED
```

- **示例 3** – 删除分区大小触发器：

```
drop size trigger partition p1
   for INVENTORY_RS
   size exceeds 80
```

- **示例 4** – 删除延迟时间连接触发器：

```
drop latency trigger
   connection inventory_rds.vendor
   with primary inventory_pds.ventory
   for INVENTORY_RS
   latency exceeds 300
```

用法

drop trigger 返回以下结果集：

表 63. `drop event trigger` 的列说明

列	说明
Action	操作名称
Result	执行结果

另请参见

- add event trigger （第 633 页）
- get triggers （第 663 页）

drop server

删除 RMS 所监控的服务器。

语法

```
drop server server_name
```

参数

- **server_name** – 指定要从 RMS 中删除的服务器的名称。

示例

- **示例 1** – 从 RMS 中删除名为 INVENTORY_RS 的服务器。此代理将不再监控该服务器：

  ```
  drop server inventory_rs
  ```

用法

drop server 返回以下结果集：

表 64. `drop server` 的列说明

列	说明
Action	操作名称
Result	执行结果

另请参见

- add server （第 636 页）
- configure server （第 642 页）
- connect to server （第 644 页）
- disconnect server （第 646 页）
- get servers （第 660 页）
- shutdown server （第 668 页）

filter connection

返回连接的当前过滤器设置，或者为其设置过滤器设置。此命令可以过滤 Replication Agent 线程或 DSI 线程状态。

语法

```
filter connection for replication_server_name [{rep agent | dsi}
[={on | off}]]
```

参数

- **connection** – 指定要过滤的连接的名称。
- **replication_server_name** – 要过滤的 Replication Server 的名称。
- **rep agent**、**dsi** – 指定要过滤的连接部分。
- **on**、**off** – 将连接过滤设置为 on 或 off。

示例

- **示例 1** – 返回 prs1 中的“inventory_pds.vendor”连接的过滤器设置列表：

  ```
  filter inventory_pds.vendor for prs1
  ```

- **示例 2** – 隐藏 prs1 中的“inventory_pds.vendor”连接的 DSI 线程状态：

  ```
  filter inventory_pds.vendor dsi for prs1 dsi = on
  ```

- **示例 3** – 关闭 prs1 中“inventory_pds.item”连接的 **rep agent** 过滤：

  ```
  filter inventory_pds.item for prs1 rep agent = off
  ```

用法

- 当打开过滤器时，连接状态将显示为“隐藏”。连接状态不会累积到 Replication Server 状态中。
- 如果打开 **rep agent** 过滤器，RMS 不会报告 Adaptive Server Enterprise、Replication Agent 或 Replication Server 中的 Replication Agent 线程或 RepAgent 线程的状态。
- 如果调用 **filter** 命令时没有指定任何选项，它将返回指定连接的列表。
- **filter** 返回以下结果集：

表 65. `filter connection` 结果集（所过滤的连接的列表）

列	说明
RepServer	Replication Server 名称
Connection	连接名

列	说明
DSI	DSI 的过滤值
rep agent	**rep agent** 的过滤值

- 如果为连接打开或关闭了过滤，**filter** 命令将返回以下结果集：

表 66. `filter connection` 结果集（打开/关闭了过滤）

列	说明
Action	操作名称
Result	执行结果

另请参见

- get network spec （第 658 页）
- get threads （第 663 页）

get component

返回 RMS 所监控的组件的列表。组件是服务器中的监控对象。Replication Server 组件是指连接、逻辑连接、路由、队列和分区；Adaptive Server Enterprise 组件是指 RepAgent 线程。

语法

```
get {connections | logical connections | routes | queues | partitions
|
        repagents}
    for server_name [, component_name]...
```

参数

- **connections、logical connections、routes、queues、partitions、repagents** – 返回 RMS 所监控的指定类型的组件。例如，返回 RMS 所监控的指定 Replication Server 中的所有连接。
- **server_name** – 指定包含所请求的组件的服务器。如果服务器不包含请求的任何组件，**get** *component* 将返回空结果集。
- **component_name** – 指定要返回的特定组件或组件列表。组件是服务器中的监控对象。Replication Server 组件包括连接、逻辑连接、路由、队列和分区。Adaptive Server Enterprise 组件是指 RepAgent 线程。

示例

- **示例 1** – 返回 RMS 所监控的 Replication Server INVENTORY_RS 中的所有连接的列表：

```
get connections for INVENTORY_RS
```

- **示例 2** – 返回 RMS 所监控的名为 INVENTORY_PDS 的 Adaptive Server Enterprise 服务器中的所有 RepAgent 线程的列表：

```
get repagents for INVENTORY_PDS
```

- **示例 3** – 返回 Replication Server INVENTORY_RS 的名为“inventory_rs.euro_sales”的路由的信息：

```
get routes for INVENTORY_RS, inventory_rs.euro_sales
```

用法

- 此命令还会返回远程 RMS 所监控的组件。
- **get connections** 支持检索与数据服务器或 Replication Agent 进程关联的连接。它支持除 Replication Server 以外的服务器：
  - ASE – **get connections** 返回 ASE 中的每个数据库的连接信息。RMS 将搜索 RMS 中的所有 Replication Server 以查找名为 *ASE_name.database* 的连接。
  - Replication Agent/MRA – **get connections** 返回与 Replication Agent 关联的主连接的信息。与 Replication Agent 或 MRA 关联的连接的名称存储在配置参数 *rs_source_ds* 和 *rs_source_db* 中。**get connections** 搜索 RMS 中的所有 Replication Server 以查找该连接。
  - dbltm – **get connections** 返回与 dbltm 关联的主连接的信息。在将服务器添加到环境中时，可以选择提供 dbltm 的连接信息。如果没有提供此信息，**get connections** 将返回空结果集，并在 RMS 日志中写入一条警告消息以指明缺少此信息。
  - DirectConnect – **get connections** 返回数据服务器与 DirectConnect 服务器名称匹配的所有连接的信息。
  - SA/IQ – **get connections** 返回数据服务器与 SA 或 IQ 服务器名称匹配的信息。SA 或 IQ 服务器不使用数据库名称。
- 如果 RMS 不监控指定的服务器，**get** *component* 命令将返回一条错误消息。
- **get** *component* 返回以下结果集（某些结果随组件类型而变化）：

表 67. get component 结果集的列说明

列	说明
Server	包含组件的服务器的名称。
Name	组件的名称。
Type	组件类型（连接、路由、队列、RepAgent）。

列	说明
Last Monitored	一个时间戳，它指示 RMS 上次监控组件的时间。时间戳采用格式 MM/DD/YYYY HH:MM:SS。
State	定义组件状态的说明。
State Constant	定义组件状态的整数常量。有关状态信息，请参见“RMS 服务器和组件状态”。
Description	描述组件状态的原因字符串。
More Descriptions	指示是否提供了附加信息。如果为 true，则组件状态包含多个说明。可以使用 **get status descriptions** 命令来检索组件的所有说明的列表。
Intermediate Rep-Server	指定路由的中间节点。如果路由为直接路由，*Intermediate RepServer* 应该为空。
Queue Number	队列号。
Queue Type	队列类型。
Size column	队列大小。

另请参见

- configure component （第 638 页）
- get status descriptions （第 661 页）
- get servers （第 660 页）
- resume component （第 666 页）
- suspend component （第 671 页）

get group

返回一个结果集，其中包含组列表和每个组的累积状态，或者包含某个组中每个服务器的状态和指定组的累积状态。累积状态显示为所报告的最低状态；例如，如果未“运行”组内的任何服务器，则该组的状态将报告为“可疑”。

语法

```
get group [group_name]
```

参数

- **group_name** – 指定要为其检索服务器列表的组的名称。

示例

- **示例 1** – 返回组名称列表以及每个组的累积状态：

```
get group
```

```
Group
Name  State       State     Description              More
          Constant                                   Descriptions
----------  --------    -------   ------------------------  -----
-------
group1      4           Suspect   inventory_rs1 is Suspect  False
```

- **示例 2** – 返回组“inventory_mra”所包含的每个服务器名称列表的状态以及该组的累积状态：

```
get group inventory_mra
```

```
Group Name     Server Name  Server Type         Last
Monitored
-------------  -----------  ------------------  ----------------
---
inventory_mra  RAObeta      Replication Agent    12/16/2005
13:38:30

Version String
-------------------------------------------------------------------
---------
Sybase Replication Agent for Unix & Windows/12.6.0.5001/B/generic/
JDK 1.4.2/main/5001/VM: Sun Microsystems Inc. 1.4.2_05/OPT/Wed
May  4
02:42:07 MDT 2005

State Constant  State   Description                   More
Descriptions
--------------  ------  ----------------------------  ----------
---------
6                Admin    Waiting for operator command.    false
```

用法

- 如果没有提供 *group_name* 参数，**get group** 将返回一个结果集，其中包含每个组的累积状态：

表 68. `get group` 的列说明（组列表和每个组的累积状态）

列	说明
Group Name	组名。
State Constant	定义组状态的整数常量。
State	定义组状态的说明。这是 State Constant 列的字符串表示形式。

列	说明
Description	描述组状态的原因字符串。如果包含多个说明，此字段应包含第一个说明。
More Descriptions	一个标志，它指示是否包含多个描述组状态的说明字符串。

- 如果提供 *group_name* 参数，**get group** 将返回一个结果集，其中包含每个服务器的状态：

表 69. `get group` 的列说明（单个服务器和指定组的累积状态）

列	说明
Group Name	组名。
Server Name	服务器的名称。
Server Type	服务器类型（Replication Server、Adaptive Server Enterprise、Replication Agent 等）。
Last Monitored	一个时间戳，它指示 RMS 上次监控服务器的时间。时间戳采用格式 *MM/DD/YYYY HH:MM:SS*。
Version String	返回服务器版本字符串。
State Constant	服务器的数值状态。
State	定义服务器状态的说明。这是 State Constant 的字符串表示形式。
Description	描述服务器状态的原因字符串。
More Descriptions	指示是否提供了附加信息。如果为 true，则服务器状态包含多个说明。使用 **get status** 说明可检索服务器的所有说明的列表。

另请参见

- create group （第 645 页）
- delete group （第 646 页）
- get status descriptions （第 661 页）

get heartbeat

检索 RMS 中已定义的心跳。心跳是一个进程，它在主数据库中按指定间隔运行 Replication Server **rs_ticket** 存储过程。输出或心跳票据存储在复制数据库的表中。

语法

```
get heartbeat [for ds.db]
```

参数

- **ds.db** – 参与心跳进程的连接的名称。此名称可以是主连接，也可以是复制连接。

示例

- **示例 1** – 检索 RMS 中定义的所有心跳：

```
get heartbeat
```

- **示例 2** – 检索为“inventory_pds.pdb1”连接所定义的心跳：

```
get heartbeat for inventory_pds.pdb1
```

用法

get heartbeat 返回以下结果集：

表 70. `get heartbeat` 的列说明

列	类型	说明
Primary	*varchar*	主数据服务器和数据库的名称。
Replicate	*varchar*	复制数据服务器和数据库的名称。
Interval	*int*	RMS 执行 **rs_ticket** 命令的间隔（以秒为单位）。
Max Rows	*int*	*rms_ticket_history* 表可以包含的最大行数。RMS 将在每个心跳间隔测试表的大小。如果表大于 *max_rows*，RMS 将删除最早的条目。

另请参见

- get heartbeat tickets （第 656 页）
- start heartbeat （第 669 页）
- stop heartbeat （第 670 页）

get heartbeat tickets

从 *rms_ticket_history* 表中，为指定的心跳进程、日期和时间范围检索一组票据。对于复制过程中的每一步，票据输出均包含一组日期和时间字段。该日期和时间将同步到复制数据服务器系统时间。

语法

```
get heartbeat tickets from pds.pdb to rds.rdb
    [start date time]
    [end date time]
    [last num_tickets]
```

参数

- **pds.pdb** – 主数据服务器和数据库的名称。
- **rds.rdb** – 复制数据服务器和数据库的名称。
- **start date time** – 票据范围的开始日期和时间。RMS 从此时间开始检索票据信息，一直检索到结束时间或表的末尾。如果没有提供此参数，RMS 将从表中最早的票据开始。
- **end date time** – 票据范围的结束日期和时间。RMS 从指定的时间开始检索票据信息，一直检索到此时间为止。如果没有提供此参数，RMS 将包括从开始时间开始的所有票据。
- **last num_tickets** – 从表中检索指定数量的票据。不能将此参数与 **start** 和 **end** 参数一起使用。

示例

- **示例 1** – 检索 *rms_ticket_history* 表中的所有行：

```
get heartbeat tickets
    from inventory_pds.vendor to inventory_dss.vendor
```

- **示例 2** – 检索 10 月 29 日到 11 月 3 日之间的所有行：

```
get heartbeat tickets
    from inventory_pds.vendor to inventory_dss.vendor
    start Oct 29, 2005 12:00am
    end Nov 3, 2005 12:00am
```

- **示例 3** – 检索表中从 10 月 29 日 1:30 时开始的所有行：

```
get heartbeat tickets
    from inventory_pds.vendor to inventory_dss.vendor
    start 10/29 1:30pm
```

- **示例 4** – 检索表中最新的 500 行：

```
get heartbeat tickets
    from inventory_pds.vendor to inventory_dss.vendor
    last 500
```

用法

- **start** 和 **end** 参数支持多种日期和时间格式；例如，可以输入 MM/DD/YYYY（如 10/29/2005）或 MMM DD, YYYY（如 Oct 29, 2005）格式的日期。时间字段支持没有秒或毫秒的条目以及已本地化的日期和时间格式。
- 结果集中的所有日期将同步到复制数据服务器系统时间。在生成结果集之前，RMS 将从数据服务器和 Replication Server 中检索日期和时间，并根据服务器时间与 RMS 系统时间之间的差值来调整时间。
- **get heartbeat tickets** 命令返回以下结果集：

表 71. `get heartbeat tickets` 的列说明

列	类型	说明
Primary	*varchar*	主数据服务器和数据库的名称。
Replicate	*varchar*	复制数据服务器和数据库的名称。
PDB	*datetime*	在主数据库中执行 **rs_ticket** 存储过程的时间。
EXEC	*datetime*	票据传递到主 Replication Server 执行程序线程的时间。
Bytes	*int*	执行程序线程从 RepAgent 或 Replication Agent 收到的总字节数。
DIST	*datetime*	票据传递到主 Replication Server 分配器线程的时间。
DSI	*datetime*	票据传递到复制 Replication Server DSI 线程的时间。
RDB	*datetime*	票据到达复制数据服务器的时间。结果集是按 RDB 字段进行排序的。

另请参见

- get heartbeat （第 655 页）
- start heartbeat （第 669 页）
- stop heartbeat （第 670 页）

get network spec

检索 RMS 已知的所有服务器的连接信息。此列表是从 RMS 的 `interfaces` 文件或 LDAP 服务器中检索的。该列表包含服务器名、主机名以及服务器使用的端口号。

语法

```
get network spec [[monitored] | [server_name [,server_name]]]
```

参数

- **monitored** – 返回 RMS 当前监控的服务器的列表。
- **server_name** – 指定要为其检索信息的一个或一组服务器的名称。

示例

- **示例 1** – 从 RMS 的 `interfaces` 文件或 LDAP 服务器中检索所有服务器的列表：

  ```
  get network spec
  ```

- **示例 2** – 检索 RMS 所管理的一组服务器的连接信息：

```
get network spec monitored
```

- **示例 3** – 检索服务器 INVENTORY_RS 和 INVENTORY_ASE 的连接信息：

```
get network spec INVENTORY_RS, INVENTORY_ASE
```

用法

- 如果请求的服务器不存在或者 `interfaces` 文件或 LDAP 服务器不可用，则返回空结果集。
- **get network spec** 返回以下结果集：

表 72. `get network spec` 的列说明

列	说明
Name	服务器的名称
Host	服务器所在的计算机的名称
Port	服务器监听的主机端口号

另请参见

- filter connection （第 650 页）

get rmiaddress

检索远程方法调用 (RMI) 服务的地址。通过使用 RMI，在一个 Java 虚拟机 (VM) 中运行的对象可以调用在另一个 Java VM 中运行的对象上的方法。RMI 可以在使用 Java 编写的程序之间提供远程通信。

RMS 为客户端应用程序提供了注册在发生特定事件时执行的回调例程的功能。RMS 使用远程 RMI 功能来提供异步回调。

语法

```
get rmiaddress
```

参数

- **rmiaddress** – 返回用于 RMI 服务的服务器和端口。

示例

- **示例 1** – 检索 RMI 服务的地址：

```
get rmiaddress
```

```
Rmi Address
--------------------
rmi://redtail:9999/
```

用法

get rmiaddress 返回 RMI 服务的地址。

get servers

返回 RMS 所监控的每个服务器的状态，后跟 RMS 环境状态。RMS 状态是所监控的服务器的累积状态，它显示报告的最低状态；例如，如果列表中的任何服务器的状态不是“UP”，则将 RMS 状态报告为“SUSPECT”。

语法

```
get servers [[for group group_name] | [{ASA | ASE | DirectConnect |
IQ |
Replication Agent | MRA | Replication Server | RMS | Open Server |
[server_name,]…}]]
```

参数

- **ASA、ASE、DirectConnect、IQ、Replication Agent、MRA、Replication Server、RMS、Open Server** – 仅返回 RMS 所监控的指定类型的服务器。例如，返回 RMS 所监控的所有 Replication Server。
- **group_name** – 指定要为其返回服务器的组。
- **server_name** – 指定要返回的特定服务器或服务器列表。如果 RMS 没有监控服务器，则返回空结果集。

示例

- **示例 1** – 返回 RMS 所监控的所有服务器的状态，后跟 RMS 环境状态：

  ```
  get servers
  ```

- **示例 2** – 返回 RMS 所监控的所有 Adaptive Server Enterprise 服务器的列表：

  ```
  get servers ASE
  ```

- **示例 3** – 返回一个列表，其中包含 INVENTORY_RS 和 INVENTORY_PDS 服务器的信息：

  ```
  get servers INVENTORY_RS, INVENTORY_PDS
  ```

用法

此命令还会返回远程 RMS 所监控的服务器。

表 73. `get servers` 的列说明

列	说明
Name	服务器名。
Type	服务器类型（Replication Server、Adaptive Server Enterprise、Replication Agent 等）。
Last Monitored	一个时间戳，它指示 RMS 上次监控服务器的时间。时间戳采用格式 *MM/DD/YYYY HH:MM:SS*。
Version String	服务器的完整版本字符串。
State Constant	定义服务器状态的整数常量。有关服务器状态信息，请参见“RMS 服务器和组件状态”。
State	定义服务器状态的说明。这是 State Constant 的字符串表示形式。
Description	描述服务器状态的字符串。
More Descriptions	指示是否提供了附加信息。如果为 true，则服务器状态包含多个说明。可以使用 **get status descriptions** 命令来检索服务器的所有说明的列表。

另请参见

- add server （第 636 页）
- configure server （第 642 页）
- connect to server （第 644 页）
- disconnect server （第 646 页）
- drop server （第 649 页）
- get component （第 651 页）
- get status descriptions （第 661 页）
- shutdown server （第 668 页）

get status descriptions

检索服务器或组件的状态说明的列表。组件是服务器中的监控对象。服务器或组件状态由状态整数常量和说明字符串列表组成。**get server** 和 **get component** 命令返回列表中的第一个说明，以及指示说明列表是否包含多个字符串的标志。

客户端应用程序可以使用 **get server** 或 **get component** 来显示 RMS 所监控的所有服务器的状态。如果需要详细信息，应用程序可以显示所有说明。

语法

```
get status descriptions {[for {connection | logical connection |
route | queue |
```

```
        rep agent | partition}
    component_name] for server_name | for group_name}
```

参数

- **connection、logical connection、route、queue、rep agent、partition** – 返回指定服务器或组件的状态说明。
- **component_name** – 指定要为其返回状态说明的组件的名称。组件是服务器中的监控对象。Replication Server 组件是指连接、逻辑连接、路由、队列和分区。Adaptive Server Enterprise 组件是指 RepAgent 线程。
- **server_name** – 指定要为其返回状态说明的服务器的名称。在返回组件的状态说明时，也会使用服务器名称。
- **group_name** – 指定要为其返回状态说明的组的名称。

示例

- **示例 1** – 检索服务器名称 INVENTORY_RS 的所有说明字符串：

  ```
  get status descriptions for INVENTORY_RS
  ```

- **示例 2** – 检索组名“group1”的所有说明字符串：

  ```
  get status descriptions for group1
  ```

- **示例 3** – 检索 INVENTORY_ASE 服务器中的“inventory_pds.pdb1”连接的所有说明字符串：

  ```
  get status descriptions
      for connection inventory_pds.pdb1 for INVENTORY_ASE
  ```

用法

- **get status descriptions** 返回说明列表中的所有字符串（包括第一个说明）。
- 可以使用 **get status descriptions** 来返回 RMS 的状态说明。
- **get status descriptions** 返回一个结果集，其中包括一个字符串列（包含一个状态说明）。结果集将返回多行，每个说明一行。

另请参见

- get component （第 651 页）
- get servers （第 660 页）

get threads

显示有关 Replication Server 中正在运行的线程的信息。

语法

```
get threads [for] server_name [{dist | dsi | rsi | sqm | sqt}]
```

参数

- **server_name** – 指定包含线程的 Replication Server。
- **dist | dsi | rsi | sqm | sqt** – 指定线程类型。如果未指定任何类型，将返回线程的摘要列表。

示例

- **示例 1** – 返回 Replication Server INVENTORY_RS 中的所有线程的摘要列表：

  ```
  get threads for INVENTORY_RS
  ```

- **示例 2** – 返回 Replication Server INVENTORY_RS 中的所有路由线程的线程信息：

  ```
  get threads for INVENTORY_RS rsi
  ```

用法

get threads 为指定的 Replication Server 执行 **admin who** 命令。此结果集与 **admin who** 结果集完全相同。

另请参见

- filter connection （第 650 页）
- resume component （第 666 页）
- suspend component （第 671 页）

get triggers

显示 RMS 所监控的触发器的相关信息。

语法

```
get status triggers
    [{connection | logical connection | route | queue | rep agent |
        partition}
    component_name for server_name]
```

参数

- **status** – 指定触发器的类型。
- **connection**、**logical connection**、**route**、**queue**、**rep agent**、**partition** – 指定要监控的组件的类型。组件是服务器中的监控对象。Replication Server 组件是指连接、逻辑连接、路由、队列和分区。Adaptive Server Enterprise 组件是指 RepAgent 线程。
- **component_name** – 指定要监控的组件的名称。
- **server_name** – 指定要监控的服务器的名称。

示例

- **示例 1** – 返回 RMS 中的所有触发器的列表：

```
get triggers
```

- **示例 2** – 返回为 Replication Server INVENTORY_RS 定义的所有触发器的列表：

```
get triggers for INVENTORY_RS
```

- **示例 3** – 返回为 Replication Server INVENTORY_RS 中的连接“inventory_pds.vendor”所定义的所有触发器的列表：

```
get triggers connection inventory_pds.vendor for
     INVENTORY_RS
```

用法

get triggers 返回以下结果集：

表 74. `get triggers` 的列说明

列	说明
Type	触发器类型。
Server Type	触发器的服务器类型。
Server Name	触发器的服务器名称。
Component Type	触发器的组件类型。
Component Name	触发器的组件名称。
Primary Connection	主连接的名称。
Change Value	使 RMS 执行触发器脚本的服务器或组件的值。
Change State	导致 RMS 执行触发器脚本的服务器或组件状态字符串。
Wait	在初始状态发生变化后，执行触发器脚本前等待的秒数。如果将 *waitInterval* 设置为零，则会立即执行脚本。

列	说明
Continuous	一个布尔标志，如果设置为 true，将使 RMS 在每个后续监控间隔执行触发器脚本，直到状态发生变化。如果没有设置此标志，RMS 仅执行一次触发器脚本。
Script	RMS 在发生事件时执行的操作系统脚本。

另请参见

- add event trigger （第 633 页）
- drop event trigger （第 647 页）

get version

检索 RMS 的版本字符串。

语法

```
get version
```

参数

- **version** – 返回一个字符串，其中包含几条用斜杠分隔的版本信息。

示例

- **示例 1** – 检索 RMS 的版本字符串：

```
version
```

```
------------------------------------------------------------------------
---------
Replication Monitoring Services/15.0/P/generic/JDK 1.4.2.03/main/
Build 102/VM:
Sun Microsystems Inc. 1.5.0_05/Opt/Wed Dec 7 15:26:13 CST 2005
```

用法

get version 返回 RMS 的版本字符串。

另请参见

- configure RMS （第 640 页）
- resume Replication Agent （第 668 页）
- suspend Replication Agent （第 673 页）
- trace （第 673 页）

log level

返回当前日志级别设置。**log level** 也可更改 RMS 的日志级别设置。

语法

```
log level [= {debug | info | warn | error | fatal}]
```

参数

- **debug、info、warn、error、fatal** – 日志级别值。

示例

- **示例 1** – 返回当前的日志级别设置：

  ```
  log level
  ```

- **示例 2** – 将日志级别设置为 error：

  ```
  log level = error
  ```

用法

日志级别依次为：**debug**、**info**、**warn**、**error**、**fatal**。必须将日志级别至少设置为 **info** 才能跟踪日志级别消息。

resume component

恢复指定服务器中的组件。此命令恢复 Replication Server 中的 DSI 线程、Replication Agent 线程、队列或路由，或者恢复 Adaptive Server Enterprise 中的 RepAgent 线程。

语法

```
resume {dsi | queue | rep agent | route} component_name
for {server_name | group_name} [skip transaction | execute
transaction]
```

参数

- **dsi、queue、rep agent、route** – 指定要恢复的组件类型。如果要恢复 Adaptive Server Enterprise 中的 RepAgent 线程，则组件为数据库名称。否则，组件为连接、队列或路由名称。
- **component_name** – 指定要恢复的组件的名称。
- **group_name** – 指定组名。将恢复组中的每个组件。

- **server_name** – 指定包含组件的 Replication Server 或 Adaptive Server Enterprise 的名称。
- **skip transaction** – 如果为 DSI 连接提供了该选项，将指示 Replication Server 重新开始执行连接的队列中的第二个事务。第一个事务将被写入数据库例外日志。

  如果为队列提供此选项，则指定 SQM 应跳过在重新启动后遇到的第一个大消息。

  如果为路由提供此选项，则忽略遇到的第一个宽消息大于 16K 个字节的事务。
- **execute transaction** – 如果系统事务是 DSI 队列中的第一个事务，在 DSI 启动后，将覆盖 Replication Server 针对此系统事务应用的限制。

示例

- **示例 1** – 恢复 Replication Server INVENTORY_RS 中连接“inventory_pds.vendor”的 DSI 线程。而无需等待当前操作完成：

```
resume dsi inventory_pds.vendor for INVENTORY_RS with
       nowait
```

- **示例 2** – 恢复 Replication Server INVENTORY_RS 中连接“inventory_pds.vendor”的 Replication Agent 线程。

```
resume rep agent inventory_pds.vendor for INVENTORY_RS
```

- **示例 3** – 启动 Adaptive Server Enterprise INVENTORY_PDS 中的数据库 *vendor* 的 RepAgent 线程：

```
resume rep agent vendor for INVENTORY_PDS
```

用法

- **rep agent** 组件类型用于恢复 Replication Server 中连接的 Replication Agent 线程或 Adaptive Server Enterprise 中的 RepAgent 线程。
- **skip transaction** 选项适用于 Replication Server DSI 连接、队列或路由。
- **execute transaction** 选项仅适用于 Replication Server DSI 连接。当恢复 Adaptive Server Enterprise 中的 RepAgent 线程时，**resume** 将发出 **sp_start_rep_agent**。
- **resume** 返回以下结果集。

表 75. resume component 的列说明

列	说明
Action	操作名称
Result	执行结果

另请参见

- configure component （第 638 页）
- get component （第 651 页）

- get threads （第 663 页）
- suspend component （第 671 页）

resume Replication Agent

在 Replication Agent 中恢复复制。

语法

```
resume {server_name | group_name}
```

参数

- **server_name** – 指定要恢复的 Replication Agent 的名称。
- **group_name** – 指定组名。将恢复组中的每个 Replication Agent。

示例

- **示例 1** – 恢复 Replication Agent“sales_ra”：

  ```
  resume sales_ra
  ```

用法

None

另请参见

- configure RMS （第 640 页）
- get version （第 665 页）
- suspend Replication Agent （第 673 页）
- trace （第 673 页）

shutdown server

向服务器发出 **shutdown** 命令。

语法

```
shutdown {server_name | group_name} [with nowait]
```

参数

- **server_name** – 指定要关闭的服务器。

- **group_name** – 指定组名。将关闭组中的每个服务器。
- **with nowait** – 立即关闭服务器，而无需等待正在执行的操作完成。

示例

- **示例 1** – 向名为 INVENTORY_RS 的服务器发出 **shutdown** 命令：

```
shutdown INVENTORY_RS
```

用法

用户只能 使用 RMS 关闭 Replication Server、Replication Agent 和 Mirror Replication Agent。

另请参见

- add server （第 636 页）
- configure server （第 642 页）
- connect to server （第 644 页）
- disconnect server （第 646 页）
- drop server （第 649 页）
- get servers （第 660 页）

start heartbeat

设置并启动从指定主连接到指定复制连接的心跳进程。

语法

```
start heartbeat from pds.pdb to rds.rdb
    [set interval [to] hb_interval]
    [set maximum rows [to] max_rows]
    [do not load rs_ticket_report]
```

参数

- **pds.pdb** – 主数据服务器和数据库的名称。此名称必须与现有主连接相关联。
- **rds.rdb** – 复制数据服务器和数据库的名称。此名称必须与现有主兼复制连接相关联，或者与现有仅复制连接相关联。
- **hb_interval** – RMS 执行 **rs_ticket** 命令的间隔（以秒为单位）。缺省值为 60 秒。
- **max_rows** – *rms_ticket_history* 表可以包含的最大行数。RMS 将在每个心跳间隔测试表的大小。如果表大于 *max_rows*，RMS 将删除最早的条目。RMS 从表中删除 10% *max_row* 大小的行。缺省值为 5000 行。

- **do not load rs_ticket_report** – 如果包含此标志，则 RMS 不装载 *rs_ticket_report*，您可以提供一个自定义的存储过程。提供的 **rs_ticket_report** 过程必须使用所需信息来装载 *rms_ticket_history* 表。

示例

- **示例 1** – 设置并启动心跳进程，然后每隔 60 秒执行一次 **rs_ticket** 过程；将 *rms_ticket_history* 表限制为 5000 行：

```
start heartbeat
      from inventory_pds.vendor to inventory_dss.vendor
```

用法

- 若要设置心跳，RMS 应使用向域中添加服务器时所提供的用户名。这些用户名必须具有执行以下操作的正确权限：在复制数据库中创建表和存储过程、在复制 Replication Server 中配置 DSI 以及在主数据库中执行 **rs_ticket** 存储过程。
- RMS 只能在主数据库和复制数据库之间创建一个心跳。如果心跳已存在，RMS 将会生成一个错误。
- 如果某个心跳已存在，RMS 并不会删除 *rms_ticket_history* 表，但假定另一个主数据库中的另一个心跳已在执行。
- RMS 假定将复制数据库设置为从 Replication Server 接收数据，并且它既不检查预订，也不生成新的预订。Replication Server 版本必须至少为 12.6。
- 在复制 Replication Server 发送 **rs_ticket** 信息之前，Replication Server 要求复制数据库必须至少有一个表、存储过程或数据库的预订。此预订不必针对任何特定的表或存储过程。如果没有任何预订，则 **rs_ticket** 在热备份环境中运行。

另请参见

- get heartbeat （第 655 页）
- get heartbeat tickets （第 656 页）
- stop heartbeat （第 670 页）

stop heartbeat

停止主数据库和复制数据库之间的心跳进程。可以选择截断 *rms_ticket_history* 表。

语法

```
stop heartbeat from pds.pdb to rds.rdb
    [delete history]
```

参数

- **pds.pdb** – 主数据服务器和数据库的名称。

- **rds.rdb** – 复制数据服务器和数据库的名称。
- **delete history** – 如果包含此选项，则在停止心跳时删除 *rms_ticket_history* 表。缺省情况下，不会删除该表。

示例

- **示例 1** – 停止心跳进程：

```
stop heartbeat
     from inventory_pds.vendor to inventory_dss.vendor
```

用法

可以选择在停止心跳时删除 *rms_ticket_history* 表。这意味着，您将无法再从表中检索票据。

另请参见

- get heartbeat （第 655 页）
- get heartbeat tickets （第 656 页）
- start heartbeat （第 669 页）

suspend component

挂起指定服务器中的组件。此命令挂起 Replication Server 中的 DSI 线程、路由，或者挂起 Adaptive Server Enterprise 中的 RepAgent 线程。

语法

```
suspend {dsi | rep agent | route} component_name
    for {server_name | group_name} [with nowait]
```

参数

- **dsi、rep agent、route** – 指定要挂起的组件类型。
- **component_name** – 指定要挂起的组件的名称。如果要挂起 Adaptive Server Enterprise 中的 RepAgent 线程，则组件为数据库名称。否则，组件为连接或路由名称。
- **server_name** – 指定包含组件的 Replication Server 或 Adaptive Server Enterprise 的名称。
- **group_name** – 指定组名。将挂起组中的每个组件。
- **with nowait** – 立即挂起组件，而无需等待正在执行的操作完成。

示例

- **示例 1** – 挂起 Replication Server INVENTORY_RS 中的“inventory_pds.vendor”连接的 DSI 线程，而无需等待当前操作完成：

```
suspend dsi inventory_pds.vendor
    for INVENTORY_RS with nowait
```

- **示例 2** – 挂起名为 INVENTORY_RS 的 Replication Server 中连接“inventory_pds.vendor”的 Replication Agent 线程：

```
suspend rep agent inventory_pds.vendor for INVENTORY_RS
```

- **示例 3** – 停止名为 INVENTORY_PDS 的 Adaptive Server Enterprise 中数据库 *vendor* 的 RepAgent 线程：

```
suspend rep agent vendor for INVENTORY_PDS
```

用法

- **rep agent** 组件类型用于挂起 Replication Server 中连接的 Replication Agent 线程或 Adaptive Server Enterprise 中的 RepAgent 线程。
- **with nowait** 选项适用于 Replication Server DSI 连接或 Adaptive Server Enterprise RepAgent 线程。
- 挂起 Adaptive Server Enterprise 中的 RepAgent 线程时，**suspend component** 将发出 **sp_stop_rep_agent** 存储过程。
- **suspend component** 返回以下结果集：

表 76. `suspend component` 的列说明

列	说明
Action	操作名称。
结果	执行结果。

另请参见

- configure component （第 638 页）
- get component （第 651 页）
- get threads （第 663 页）
- resume component （第 666 页）

suspend Replication Agent

在 Replication Agent 中挂起复制。

语法

```
suspend {server_name | group_name}
```

参数

- **server_name** – 指定要挂起的 Replication Agent 的名称。
- **group_name** – 指定组名。将挂起组中的每个 Replication Agent。

示例

- **示例 1** – 挂起 Replication Agent“sales_ra”：

  ```
  suspend sales_ra
  ```

用法

None

另请参见

- configure RMS （第 640 页）
- get version （第 665 页）
- resume Replication Agent （第 668 页）
- trace （第 673 页）

trace

在 RMS 日志文件中显示跟踪信息。

语法

```
trace [flag | all {on | off}]
```

参数

- **flag** – 指定要更改设置的跟踪标志名称。
- **all** – 用于将开关值应用于所有跟踪标志的关键字。
- **on、off** – 指示是为 flag 选项中指定的跟踪点启用还是禁用跟踪。

示例

- **示例 1** – 返回所有 RMS 跟踪标志的当前设置：

```
trace
```

- **示例 2** – 打开 *RMS_Command* 跟踪标志：

```
trace RMS_Command on
```

- **示例 3** – 关闭所有跟踪标志：

```
trace all off
```

用法

- 只应由知识渊博的用户使用 **trace** 命令来解决 RMS 问题。
- 如果调用 **trace** 时没有指定任何选项，它将返回所有 RMS 跟踪标志的当前设置。
- 如果调用 **trace** 时指定了 flag 和 **on/off** 选项，它将更改在 flag 选项中指定的跟踪点的设置。
- 使用 **trace** 命令进行的更改会立即生效。
- RMS 支持以下跟踪标志：

表 77. 跟踪标志

标志	说明
Add_Drop_Server	添加或删除服务器时，在日志中写入一条消息。
Add_Drop_Trigger	添加或删除触发器时，在日志中写入一条消息。
Client_Connection	当客户端最初连接到 RMS 时，显示连接的相关信息。
Configuration	每次更改 RMS 配置参数时，在日志中写入一条跟踪消息。
Filter_Conn	对连接进行过滤时，在日志中写入一条跟踪消息。
Monitoring	在监控周期的每一步，在 RMS 中添加跟踪消息；并在监控每个服务器之前写入一条消息。
Network_Connection	只要创建到服务器的连接，就会在 RMS 中添加跟踪消息。将在跟踪消息中包含所有连接信息（口令除外）。
RMS_Command	将 RMS 收到的每个命令写入到错误日志中。
Server_Command	将 RMS 发送到所监控的服务器的每个命令写入到错误日志中。
Shutdown_Server	关闭服务器时，在日志中写入一条消息。
Start_Stop_Heartbeat	启动或停止心跳时，在日志中写入一条消息。
Startup	在启动过程的每一步，在 RMS 中添加跟踪消息。

标志	说明
Status_Change	在状态发生更改时，显示服务器和组件的结果说明。
Suspend_Resume_Component	挂起或恢复组件时，在日志中写入一条消息。
Trigger_Execution	显示一条消息，说明已执行事件触发器。

另请参见

- configure RMS （第 640 页）
- get version （第 665 页）
- resume Replication Agent （第 668 页）
- suspend Replication Agent （第 673 页）

首字母缩略词和缩写

列出在 Replication Server 文档中使用的或可能在 Replication Server 消息中遇到的首字母缩略词和缩写。

在《Replication Server 管理指南第二卷》中的词汇表中，可以找到许多术语的定义。

表 78. 首字母缩略词列表

首字母缩略词	代表
APC	Asynchronous Procedure Call（异步过程调用）
API	Application Program Interface（应用程序编程接口）
BM	Bitmap（位图）
C/SI	Client/Server Interfaces（客户端/服务器接口）
CM	Connection Manager（连接管理器）
dAIO	Asynchronous I/O Daemon（异步 I/O 守护程序）
dALARM	Alarm Daemon（报警守护程序）
DBO	Database Owner（数据库所有者）
dCM	Connection Manager Daemon（连接管理器守护程序）
DDL	Data Definition Language（数据定义语言）
DIST	Distributor（分发器）
DML	Data Manipulation Language（数据操纵语言）
dREC	Recovery Daemon（恢复守护程序）
DSI	Data Server Interface（数据服务器接口）
dSUB	Subscription Retry Daemon（预订重试守护程序）
ELM	Exceptions Log Manager（例外日志管理器）
ERSSD	Embedded Replication Server System Database（嵌入式 Replication Server 系统数据库）
EXC	Exception（例外）
EXEC	Executor（执行程序）
FSTR	Function String（函数字符串）
HDS	Heterogeneous datatype support（异构数据类型支持）

首字母缩略词	代表
HTS	Hash Table（散列表）
LAN	Local Area Network（局域网）
LL	Linked List（链接列表）
LTI	Log Transfer Interface（日志传送接口）
LTL	Log Transfer Language（日志传送语言）
MD	Message Delivery（消息传送）
MEM	Memory Management（内存管理）
MP	Multiprocessor（多处理器）
MSA	Multi-site Availability（多节点可用性）
NRM	Normalization（规范化）
OQID	Origin Queue ID（原始队列 ID）
PDS	Primary Data Server（主数据服务器）
PRS	Primary Replication Server（主 Replication Server）
PRS	Parser（语法分析程序）
QID	Queue ID（队列 ID）
RA	Replication Agent
RCL	Replication Command Language（复制命令语言）
RDS	Replicate Data Server（复制数据服务器）
REP AGENT	RepAgent 线程，用于 Adaptive Server 的 Replication Agent
RM	Replication Manager
RMI	Remote Method Invocation（远程方法调用）
RMP	Replication Manager plug-in（Replication Manager 插件）
RMS	Replication Monitoring Services
RPC	Remote Procedure Call（远程过程调用）
RRS	Replicate Replication Server（复制 Replication Server）
RS	Replication Server
RSI	Replication Server Interface（Replication Server 接口）
RSP	Replicated Stored Procedure（复制存储过程）

首字母缩略词	代表
RSA	Replication System Administrator（复制系统管理员）
RSI	Replication Server Interface（Replication Server 接口）
RSSD	Replication Server System Database（Replication Server 系统数据库）
SA	System Administrator（系统管理员）
SP	Stored Procedure（存储过程）
SQM	Stable Queue Manager（稳定队列管理器）
SQT	Stable Queue Transaction Interface（稳定队列事务接口）
SRE	Subscription Resolution Engine（预订解析引擎）
STS	System Table Services（系统表服务）
SUB	Subscription（预订）
TD	Transaction Delivery（事务传送）
TDS	Tabular Data Stream™（表式数据流™）
WAN	Wide Area Network（广域网）

Replication Server 设计限制

列出各种复制系统对象的最大和最小参数和值。

Replication Server 限制

变量 *For_Life_Of* 是指可以为 Replication Server 创建的对象总数，无论对象是否被删除。

例如，如果限制为 100,000，在创建了 100,000 个对象后，就不能再创建更多对象，即使删除部分或全部对象。只要复制软件仍处于已安装状态，*For_Life_Of* 计数和限制就有效。如果从系统中删除整个服务器然后重新安装它，就可以重新开始 *For_Life_Of* 计数。

表 79. Replication Server 限制

对象类型	数字
Replication Server 复制定义的 *For_Life_Of*	2^{24} (16,777,216)
Replication Server 用户的 *For_Life_Of*	2^{24} (16,777,216)
Replication Server 拒绝日志命令的 *For_Life_Of*	2^{32} - 2^{29} (3,758,096,384)
Replication Server 拒绝日志事务的 *For_Life_Of*	2^{31} (2,147,483,648)
每个 ID Server 的 Replication Server	2^{24} (16,777,216)
每个 ID Server 的数据库	2^{24} (16,777,216)
每个 Replication Server 的数据库	2^{24} (16,777,216)
Replication Server 分区的 *For_Life_Of*	2^{16} (65,536)
初始分区（用于安装 RS）的最小大小	20MB
其它分区的最小大小	1MB
最大分区大小	1TB
每个 Replication Server 的稳定队列	2^{64}
Replication Server 预订的 *For_Life_Of*	2^{31}
每个 Replication Server 的连接： • 传入连接（Replication Agent、DIST、RS、用户） • 传出连接（DSI、路由）	2^{24} - 1 2^{32} - 1

对象类型	数字
Replication Server 函数字符串的 *For_Life_Of*	2^{24} (16,777,216)
Replication Server 错误类的 *For_Life_Of*	2^{24} (16,777,216)

平台特定的限制

了解某些特定于平台操作系统的限制（如每个进程的文件描述符数），这些限制可能会影响 Replication Server 操作。

要了解特定限制，请参见针对您的平台的发行公告。

复制定义和预订限制

了解复制定义和预订限制。

表 80. 复制定义限制

对象类型	数字
每个复制定义的列	限制为 1024
每个复制定义的主键列	限制为复制定义中指定的列数
每个复制定义的可搜索列	限制为复制定义中指定的列数
每个复制定义的预订	无限制
预订 **where** 子句的字符串宽度	限制为 255 个字节

函数字符串限制

了解 Replication Server 中的函数字符串限制。

表 81. 函数字符串限制

对象类型	数字
每个函数字符串类的函数字符串	无限制
每个语言类型函数字符串模板的字节数	64K
变量值替换之后每个语言类型函数字符串模板的字节数	64K - 1 个字节
每个函数字符串的嵌入变量	无限制
函数字符串输入模板中的用户变量	1024

编程限制和参数

了解编程限制和参数。

表 82. 编程限制和参数

对象类型	数字
预订 **where** 子句中的条件数	无限制
DSI 事务组中的事务	20
一批 DSI 命令中的源命令	50
Replication Server 所处理的每个命令的字节数	16K
为每个错误类指派的操作	2^{31} (2,1474,836,448)
写入稳定队列的消息的最大大小	无限制

RMS 服务器和组件状态

提供有关 Replication Monitoring Services (RMS) 服务器和组件状态的信息。

RMS 可以监控复制环境中的服务器和组件，并提供帮助您解决问题的信息。可以使用以下两种方法来监控复制环境：主动查看有关服务器和组件状态的信息，或者在事件发生时由系统向您发出通知。

任何服务器或组件对象的状态包括：

- 一个整数状态值
- 一个描述当前状态产生原因的字符串列表

例如，Replication Server 可能因两个不同连接处于“Suspended”状态而处于“Suspect”状态。

每个监控对象的整数状态值是不同的，并且说明可能已本地化。

服务器状态

提供服务器状态的摘要。

RMS 监控以下服务器：

- Replication Server
- Adaptive Server Enterprise
- IQ
- DirectConnect
- Open Server
- Replication Agent
- RMS

表 83. 服务器状态摘要

服务器类型	值	说明
Replication Server	5	ACTIVE
	3	UNKNOWN
	0	DOWN
	4	SUSPECT
	6	HIBERNATE

服务器类型	值	说明
	7	REBUILDING
	8	RECOVERY
	9	STANDALONE
	1	TIMEOUT
	10	QUIESCE
Adaptive Server Enterprise (ASE)	5	ACTIVE
	3	UNKNOWN
	0	DOWN
	4	SUSPECT
	1	TIMEOUT
IQ	5	ACTIVE
	3	UNKNOWN
	0	DOWN
	1	TIMEOUT
DirectConnect	5	ACTIVE
	3	UNKNOWN
	0	DOWN
	1	TIMEOUT
Replication Agent / 镜像 Replication Agent (MRA)	5	ACTIVE
	3	UNKNOWN
	0	DOWN
	1	TIMEOUT
	6	ADMIN
RMS	5	ACTIVE
	3	UNKNOWN
	4	SUSPECT
	0	DOWN
	1	TIMEOUT

服务器类型	值	说明
Open Server	5	ACTIVE
	3	UNKNOWN
	0	DOWN
	1	TIMEOUT

Replication Server

了解 RMS 确定 Replication Server 状态的方式。

RMS 通过以下操作确定 Replication Server 的状态:

1. 测试到 Replication Server 的连接
2. 测试到包含 RSSD 的服务器的连接
3. 确定 Replication Server 的运行状况
4. 确定服务器连接、路由和队列的状态

Replication Server 可以处于多种状态，但 RMS 仅返回一种状态。例如，服务器状态可以同时为 HIBERNATE 和 QUIESCE。

表 84. Replication Server 状态

状态类型	值	含义	说明
正常	5	ACTIVE	Replication Server 正在运行，并且当前正在复制数据。
	10	QUIESCE	Replication Server 正在运行，但当前没有复制数据。
警告	3	UNKNOWN	在已确定实际状态之前的初始值。UNKNOWN 还可能表示服务器不是复制环境的一部分。
	4	SUSPECT	至少有一个 Replication Server 连接、路由或队列已关闭。
	6	HIBERNATE	Replication Server 处于休眠模式。此状态是由 **admin health** 命令返回的。
	7	REBUILDING	Replication Server 正在重建队列。此状态是由 **admin health** 命令返回的。
	8	RECOVERY	Replication Server 处于独立模式，并且正在重建队列。此状态是由 **admin health** 命令返回的。
	9	STANDALONE	Replication Server 处于独立模式。此状态是由 **admin health** 命令返回的。

状态类型	值	含义	说明
错误	0	DOWN	RMS 无法连接到 Replication Server 或包含 RSSD 的服务器。如果用户或口令不正确，则也会将服务器状态设置为 DOWN。
	1	TIMED OUT	尝试连接到 Replication Server 或包含 RSSD 的服务器时超时。这表示服务器已停止响应。

Adaptive Server Enterprise

了解 RMS 确定 Adaptive Server Enterprise 状态的方式。

RMS 通过以下操作确定 Adaptive Server Enterprise 的状态：

1. 测试到 Adaptive Server 的连接
2. 确定 Adaptive Server 的 RepAgent 线程的状态

RMS 仅测试参与复制的数据库的 RepAgent 线程，而不是测试 Adaptive Server 中的所有数据库。不会对脱机数据库进行查询。

表 85. Adaptive Server 状态

状态类型	值	含义	说明
正常	5	ACTIVE	已成功连接到 Adaptive Server；启用并启动了此环境中连接的所有 RepAgent 线程。
警告	3	UNKNOWN	在已确定实际状态之前的初始值。这还表示服务器不是复制环境的一部分。
	4	SUSPECT	在检查 RepAgent 线程状态时设置。如果禁用或停止了此环境中的连接的任何线程，则会将服务器状态设置为 SUSPECT。
错误	0	DOWN	RMS 无法连接到 Adaptive Server。如果用户或口令不正确，则也会将服务器状态设置为 DOWN。
	1	TIMED OUT	尝试连接到 Adaptive Server 时超时。表示服务器已停止响应。

IQ

IQ 使用 TDS 来参与复制环境。RMS 使用 jConnect 连接到服务器。IQ 服务器包含内部 RepAgent 线程。

RMS 测试到 IQ 服务器的连接来确定其可用性。

表 86. IQ 服务器状态

状态类型	值	含义	说明
正常	5	ACTIVE	已成功连接到 IQ 服务器。
警告	3	UNKNOWN	在已确定实际状态之前的初始值。还表示服务器不是复制环境的一部分。
错误	0	DOWN	RMS 无法连接到 IQ 服务器。如果用户或口令不正确，则也会将服务器状态设置为 DOWN。
	1	TIMED OUT	尝试连接到 IQ 服务器时超时。表示服务器已停止响应。

DirectConnect

了解 RMS 确定 DirectConnect 状态的方式。

RMS 通过以下操作确定 DirectConnect 的状态：

1. 测试到 DirectConnect 的连接
2. 测试从 DirectConnect 到后端数据服务器的连接

表 87. DirectConnect 服务器状态

状态类型	值	含义	说明
正常	5	ACTIVE	RMS 已成功连接到 DirectConnect，并且 DirectConnect 可以连接到后端数据服务器。
警告	3	UNKNOWN	在已确定实际状态之前的初始值。还表示代理不是复制环境的一部分。
错误	0	DOWN	RMS 无法连接到 DirectConnect。如果用户或口令不正确，则也会将服务器状态设置为 DOWN。另外，如果 DirectConnect 无法连接到后端数据服务器，则会将该状态设置为 DOWN。
	1	TIMED OUT	尝试连接到 DirectConnect 时超时。表示服务器已停止响应。

Open Server

了解 RMS 确定 Open Server 状态的方式。

RMS 测试到 Open Server 的连接。

表 88. Open Server 状态

状态类型	值	含义	说明
正常	5	ACTIVE	已成功连接到 Open Server。

状态类型	值	含义	说明
警告	3	UNKNOWN	在已确定实际状态之前的初始值。还表示服务器不是复制环境的一部分。
错误	0	DOWN	RMS 无法连接到 Open Server。DOWN 还可能表示用户或口令不正确。
	1	TIMED OUT	尝试连接到 Open Server 时超时。这表示服务器已停止响应。

Replication Agent

了解 RMS 确定 Replication Agent 状态的方式。

RMS 通过以下操作确定 Replication Agent 的状态:

1. 测试到 Replication Agent 的连接
2. 确定该代理是处于“管理”模式还是“复制”模式

表 89. Replication Agent (MRA/MRO) 状态

状态类型	值	含义	说明
正常	5	ACTIVE	已成功连接到 Replication Agent。代理处于复制状态。此状态是由 **ra_status** 命令返回的。
警告	3	UNKNOWN	在已确定实际状态之前的初始值。还表示代理不是复制环境的一部分。
	6	ADMIN	已成功连接到 Replication Agent。代理处于管理状态，并且当前没有复制数据。此状态是由 **ra_status** 命令返回的。
错误	0	DOWN	RMS 无法连接到 Replication Agent。如果用户或口令不正确，则也会将代理状态设置为 DOWN。
	1	TIMED OUT	尝试连接到 Replication Agent 时超时。表示代理已停止响应。

RMS

中央 RMS 测试到远程 RMS 的连接。

表 90. RMS 状态

状态类型	值	含义	说明
正常	5	ACTIVE	中央 RMS 已成功连接到远程 RMS。

状态类型	值	含义	说明
警告	3	UNKNOWN	在已确定实际状态之前的初始值。还表示远程 RMS 不是复制环境的一部分。
	4	SUSPECT	表示远程 RMS 服务器或组件处于 DOWN 或 SUSPENDED 状态。
错误	0	DOWN	中央 RMS 无法连接到远程 RMS。DOWN 还可能表示用户或口令不正确。
	1	TIMED OUT	尝试连接到远程 RMS 时超时。这表示服务器已停止响应。

组件状态

了解 Replication Server 中由 RMS 监控的组件。

- 连接
- 逻辑连接
- 队列
- 路由
- 分区
- RepAgent 线程

表 91. 组件状态摘要

组件类型	值	说明
连接	5	ACTIVE
	2	SUSPENDED
	3	UNKNOWN
逻辑连接	5	ACTIVE
	2	SUSPENDED
	3	UNKNOWN
队列	5	ACTIVE
	2	SUSPENDED
	6	LOSS_DETECTED
路由	5	ACTIVE
	2	SUSPENDED

组件类型	值	说明
	3	UNKNOWN
分区	6	ONLINE
	7	OFFLINE
	8	DROPPED
RepAgent 线程 (ASE)	6	DISABLED
	7	SUSPENDED
	8	ACTIVE

连接

了解 RMS 监控 Replication Server 的数据库连接状态的方式。

数据库连接包括两个部分：RepAgent 和 DSI。Replication Server 线程的状态决定了连接状态。RMS 执行 **admin who** 命令来检索线程的状态。

RMS 分别返回 DSI 和 RepAgent 的状态。在显示连接状态时，客户端应用程序（如 Replication Manager Java 插件）可能会合并线程状态（和实际 RepAgent 的状态）。

表 92. 连接状态

状态类型	值	含义	说明
正常	5	ACTIVE	Replication Server DSI 或 RepAgent 线程未处于 DOWN 和 SUSPENDED 状态。
错误	2	SUSPENDED	Replication Server DSI 或 RepAgent 线程处于 DOWN 或 SUSPENDED 状态。
警告	3	UNKNOWN	主连接的 RepAgent 不是复制环境的一部分。

逻辑连接

了解 RMS 监控 Replication Server 的逻辑连接状态的方式。

逻辑连接包含一对在热备份环境中配置的物理连接。复制数据源是活动数据库；而复制目标是备用数据库。要对逻辑连接进行监控，RMS 需要确定活动连接的 Replication Agent 线程的状态以及备用连接的 DSI 的状态。

RMS 对活动连接的 Replication Agent 线程的状态和备用连接的 DSI 线程的状态分开进行报告。将在结果集的单独行中报告每个线程。

表 93. 逻辑连接状态

状态类型	值	含义	说明
正常	5	ACTIVE	活动物理连接的 Replication Agent 和备用物理连接的 DSI 线程均处于活动状态。
错误	2	SUSPENDED	逻辑连接可能会由于以下原因而挂起： • 没有为逻辑连接定义活动或备用物理连接。 • 活动连接的 Replication Agent 线程已挂起。 • 备用连接的 DSI 线程已挂起。 • 逻辑连接正在切换活动和备用数据库。
警告	3	UNKNOWN	活动连接的 Replication Agent 线程未知，或者备用连接的 DSI 线程未知。

队列

了解 RMS 监控 Replication Server 队列状态的方式。队列状态存储在 RSSD 中。

无论队列是启动还是关闭以及系统是否检测到任何数据丢失，存储过程 **rma_queue** 都将返回队列的名称。

表 94. 队列状态

状态类型	值	含义	说明
正常	5	ACTIVE (UP)	队列未挂起。
错误	2	SUSPENDED	队列已挂起。
警告	6	LOSS_DETECTED	在队列中已检测到数据丢失。只有在队列处于 UP 状态时，才能将状态设置为 LOSS DETECTED。

路由

了解 RMS 监控 Replication Server 路由状态的方式。

RMS 监控 Replication Server 路由的状态，并通过以下操作确定某个路由的状态：

1. 检查路由在其源和目标中的状态
2. 查询 RSSD

RMS 使用此信息来确定路由是处于 UP 还是 DOWN 状态以及查明原因。

表 95. 路由状态

状态类型	值	含义	说明
正常	5	ACTIVE	打开了路由，数据可以从源 Replication Server 传递到目标 Replication Server。
错误	2	SUSPENDED	路由不可用，无法在两个 Replication Server 之间传递数据。说明部分提供了路由挂起的原因；例如： • 路由遇到内部错误。 • 正在创建路由。 • 路由已挂起。 • 路由在目标服务器上遇到错误。 • 正在删除路由。 • 正在使用 NOWAIT 删除路由。 • 间接路由正更改为直接路由。
警告	3	UNKNOWN	目标 Replication Server 不是复制环境的一部分。

分区

了解 RMS 监控 Replication Server 分区的方式。

Replication Server 命令 **admin disk_space** 可返回分区的状态。

表 96. 分区状态

状态类型	值	含义	说明
正常	6	ONLINE	分区设备可用并且正常运行。
错误	7	OFFLINE	找不到此设备。
	8	DROPPED	已删除此设备，但有些队列仍在使用它。

RepAgent 线程

了解 RMS 监控 Adaptive Server Enterprise RepAgent 线程的方式。

sp_help_rep_agent 可确定参与复制的每个数据库的 RepAgent 线程的状态。

表 97. RepAgent 线程状态

状态类型	值	含义	说明
正常	8	ACTIVE	启用并启动了 RepAgent 线程。
错误	6	DISABLED	未启用 RepAgent 线程。

状态类型	值	含义	说明
	7	SUSPENDED	启用了 RepAgent 线程，但已将其停止。

事件触发器参数

提供有关 Replication Monitoring Services (RMS) 事件触发器参数的信息。事件触发器参数包含某个事件执行情况的相关信息，如事件名称、事件的发生日期和时间以及执行事件脚本的 RMS 的名称。只要执行了事件触发器，RMS 就会传递这些参数。

连接状态事件参数

介绍连接状态事件的参数。

连接有两种类型：入站连接和出站连接。入站连接是指从数据库经由 Replication Agent 到达 Replication Server 的连接。出站连接是指从 Replication Server 到数据库的连接。

表 98. 连接状态事件触发器参数

参数	说明
connection	将事件标识为连接状态事件的关键字。
date_time	事件发生的日期和时间。格式：Month Day Year HH:MM:SS:TTTMeridian
rms	执行事件脚本的 RMS 的名称。
object_id	发生事件的服务器。
source_type	引发事件的服务器的类型。这些值为： • repserver • 数据库
source_name	引发事件的 Replication Server 或数据服务器的名称。
ra_type	Replication Agent 的类型。这些值为： • rep agent • rep agent 线程 • dbltm • 如果是出站连接，则为空字符串 (")。
ra_name	Replication Agent 名称。如果是出站连接，则为空字符串 (")。
dest_type	目标服务器类型。这些值为： • repserver • 数据库
dest_name	目标服务器名称。

参数	说明
state	新的连接状态。

分区状态事件参数

了解分区状态事件的参数。

表 99. 分区状态事件触发器参数

参数	说明
partition	将事件标识为分区状态事件的关键字。
date_time	事件发生的日期和时间。格式：Month Day Year HH:MM:SS:TTTMeridian
rms	执行事件脚本的 RMS 的名称。
object_id	拥有分区的 Replication Server 的名称。
part_name	稳定设备的逻辑名称。
state	新的分区状态。

路由状态事件参数

了解路由状态事件的参数。

表 100. 路由状态事件触发器参数

参数	说明
route	将事件标识为路由状态事件的关键字。
date_time	事件发生的日期和时间。格式：Month Day Year HH:MM:SS:TTTMeridian
rms	执行事件脚本的 RMS 的名称。
object_id	发生事件的服务器。
repserver	将路由的源服务器标识为 Replication Server 的关键字。
server_name	源 Replication Server 的名称。
thru_type	中间服务器的类型。这些值为: • repserver • 如果路由没有中间服务器，则为空字符串 (")。
thru_name	中间 Replication Server 的名称。如果路由没有中间服务器，则为空字符串 (")。

参数	说明
repserver	将路由的目标服务器标识为 Replication Server 的关键字。
dest_name	目标 Replication Server 的名称。
state	新的路由状态。

服务器状态事件参数

了解服务器状态事件的参数。

表 101. 服务器状态事件触发器参数

参数	说明
server	用于将事件标识为服务器状态事件的关键字。
date_time	事件发生的日期和时间。格式：Month Day Year HH:MM:SS:TTTMeridian
rms	执行事件脚本的 RMS 的名称。
object_id	发生事件的服务器。
old_state	发生事件之前的服务器状态。
new_state	发生事件之后的服务器状态。
reason	发生事件的原因。

数据库连接延迟事件参数

了解数据库连接延迟事件的参数。

表 102. 数据库连接延迟事件参数

参数	说明
latency	将事件标识为延迟事件的关键字。
date_time	事件发生的日期和时间。格式：Month Day Year HH:MM:SS:TTTMeridian
rms	执行事件脚本的 RMS 的名称。
object_id	要监控其延迟的 Replication Server 的名称。
origin_dbname	从中发送事务的数据服务器和数据库的名称。
dest_dbname	将事务发送到的数据服务器和数据库的名称。

参数	说明
delta_diff	在主数据库中提交事务的时间与在复制数据库中提交事务的时间的差值（以秒为单位）。
last_commit_time	上次在目标数据库中提交事务的日期和时间。格式：Month Day Year HH:MM:SS:TTTMeridian
secs_since_last_commit	自上次提交后经历的时间（以秒为单位）。
dest_type	数据库连接的类型。这些值为： • 主兼复制 • 仅复制
reason	发生事件的原因。

队列延迟事件参数

了解队列延迟事件的参数。

表 103. 队列延迟事件参数

参数	说明
queue_latency	将事件标识为队列延迟状态事件的关键字。
date_time	事件发生的日期和时间。格式：Month Day Year HH:MM:SS:TTTMeridian
rms	执行事件脚本的 RMS 的名称。
object_id	拥有队列的 Replication Server 的名称。
log_name	队列的逻辑名称。
phys_name	队列的物理名称。
latency_in_secs	第一个块在队列中已停留的时间（以秒为单位）。

分区和队列大小阈值事件参数

了解分区和队列大小阈值事件的参数。

表 104. 分区和队列大小阈值事件参数

参数	说明
threshold	将事件标识为分区阈值或队列阈值事件的关键字。

参数	说明
date_time	事件发生的日期和时间。格式：Month Day Year HH:MM:SS:TTTMeridian
rms	执行事件脚本的 RMS 的名称。
object_id	拥有分区或队列的 Replication Server 的名称。
log_name	分区或队列的逻辑名称。
phys_name	分区或队列的物理名称。
size	指示分区使用的区域（以百分比为单位）或队列的大小（以 MB 为单位）。
object_type	标识阈值事件类型。这些值为： • 分区 • 队列

获取帮助及其它信息

使用 Sybase 入门 CD、产品文档站点和联机帮助来了解关于此产品版本的更多信息。

- Getting Started CD（或下载） – 包含 PDF 格式的发行公告和安装指南，也可能包含其它文档或更新信息。
- 位于 http://sybooks.sybase.com/ 上的产品文档 – 是 Sybase 文档的在线版本，您可以使用标准 Web 浏览器进行访问。您可以在线浏览文档，也可以采用 PDF 格式进行下载。除产品手册外，该网站还包含指向 EBF/维护、 技术文档、案例管理、已解决的案例、社区论坛/新闻组和其它资源的链接。
- 产品中的联机帮助（如果有）。

要阅读或打印 PDF 文档，您需要 Adobe Acrobat Reader，可以从 Adobe Web 站点免费下载。

注意： 产品文档网站可能会提供更新的发行公告，其中包含在产品发布后增加的重要产品或文档信息。

技术支持部门

获得 Sybase 产品支持。

如果贵组织为此产品购买了支持合同，则您的一个或多个同事将被指定为授权支持联系人。如果您有任何问题，或者在安装过程中需要帮助，请指定专人联系您所在地区的 Sybase 技术支持部门或 Sybase 子公司。

下载 Sybase EBF 和维护报告

可以从 Sybase 网站获得 EBF 和维护报告。

1. 将 Web 浏览器定位到 http://www.sybase.com/support。
2. 从菜单栏或滑出菜单中的**“支持”**下，选择**“EBF/维护”**。
3. 如果出现提示，请输入您的 MySybase 用户名和密码。
4. （可选）从**“显示”**下拉列表中选择过滤器，然后选择时间范围并单击**“开始”**。
5. 选择产品。

   挂锁图标表示您不具有特定 EBF/维护版本的下载权限，因为您未注册成为授权支持联系人。如果您尚未注册，但拥有您的 Sybase 代表提供的或通过您的支持联系人提供的有效信息，请单击**“我的帐户”**向您的 MySybase 配置文件添加“技术支持联系人”。

6. 单击“信息”图标以显示 EBF/维护报告，或者单击产品说明以下载该软件。

Sybase 产品和组件认证

认证报告检验 Sybase 产品在特定平台上的性能。

查找有关认证的最新信息：

- 有关合作伙伴产品认证，请转至 http://www.sybase.com/detail_list?id=9784
- 有关平台认证，请转至 http://certification.sybase.com/ucr/search.do

创建 MySybase 配置文件

MySybase 是一项免费服务，它允许您创建 Sybase 网页的个人化视图。

1. 转至 http://www.sybase.com/mysybase。
2. 单击“立即注册”。

辅助功能特性

辅助功能可确保所有用户（包括残障人士）都能访问电子信息。

Sybase 产品文档采用设计为实现辅助功能的 HTML 版本。

视力受损的用户可以使用自适应技术（如屏幕阅读器）浏览在线文档，或者使用屏幕放大器查看文档。

Sybase HTML 文档已经过测试，符合《美国康复法》第 508 条的辅助功能要求。符合第 508 条的文档一般也符合非美国地区的辅助功能指导原则，如针对网站的 World Wide Web 协会 (W3C) 原则。

注意： 为优化使用性能，您可能需要对辅助工具进行配置。某些屏幕阅读器按照大小写来辨别文本，例如将“ALL UPPERCASE TEXT”看作首字母缩写，而将“MixedCase Text”看作单词。您可能会发现按语法约定来配置工具更为方便。有关工具的信息，请查阅相关文档。

有关 Sybase 如何支持辅助功能的信息，请参见“Sybase 辅助功能”网站： http://www.sybase.com/products/accessibility。该网站包括有关第 508 条和 W3C 标准的信息的链接。

您可以在产品文档中找到更多有关辅助功能特性的信息。

索引

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Y

Z