SYBASE®
An SAP Company

组件集成服务用户指南

Adaptive Server® Enterprise

15.7

文档 ID：DC32968-01-1570-01

最后修订日期：2011 年 9 月

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

目录

第 1 章

简介

组件集成服务扩展了 Adaptive Server® 的功能，并提供了更高的互操作性。

它还提供了位置透明度和功能补偿。

位置透明度是指组件集成服务允许 Adaptive Server 为客户端应用程序提供统一的企业数据视图。在企业范围内，可以从异构源获得数据，如同它是本地数据源。

功能补偿允许组件集成服务模拟 Transact-SQL 的所有功能，且仅在需要实际数据时才与数据源进行交互。此功能使 Transact-SQL 的所有用途和功能均可应用于任何数据源，无论此数据源是否支持 Transact-SQL 的某一特定功能。内置函数和 Java 函数即是此功能的例子。即使函数中使用的数据是从不能支持这些函数的外部源获得，组件集成服务也允许语句使用这些函数。

组件集成服务与 Adaptive Server Anywhere、Adaptive Server IQ 和各种 DirectConnect 接口一起，实现了从企业的任何地方都可透明地访问数据库管理系统，从而扩展了 Adaptive Server 的范围。这种透明的、扩展的 Adaptive Server Enterprise 应用范围使 Enterprise Portal 组件能轻松地：

- 从任何地方访问数据并把它作为动态内容提供给 Web 页
- 执行跨越异构边界的事务
- 通过存储在 Adaptive Server/ 组件集成服务系统目录下的全局元数据所提供的单一视图来查看整个企业状况

组件集成服务允许用户访问不同服务器上的 Sybase® 数据库和非 Sybase 数据库。这些外部数据源包括数据库系统 （如 Adaptive Server 和 Oracle）中的主机数据文件、表、视图和 RPC （远程过程调用）。

使用组件集成服务可以：

- 如同访问本地表一样访问远程服务器中的表。
- 连接多个远程异构服务器中的表。例如，可以连接 Oracle 数据库管理系统 (DBMS) 和 Adaptive Server 中的表，也可以连接多个 Adaptive Server 中的表。
- 使用 select into 语句，将一个表中的内容传送到所支持的任何远程服务器上的新表中。
- 保证各种异构数据源的参照完整性。
- 使用组件集成服务的直通模式访问本机远程服务器功能。

任何需要访问多个数据源或遗留数据的用户都可使用组件集成服务。任何需要将数据从一个服务器迁移到另一个服务器的用户也可以使用它。

通常情况下，使用单个服务器来访问多个外部服务器上的数据。组件集成服务可管理位于任意位置的外部服务器上的数据。数据管理对客户端应用程序是透明的。

组件集成服务与 EnterpriseConnect™ 和 MainframeConnect™ 一起，提供对多种数据源的透明访问，这些数据源包括：

- Oracle
- Informix
- Microsoft SQL Server
- Adaptive Server Enterprise
- Adaptive Server Anywhere
- Adaptive Server IQ
- 大型机数据，包括：
  - ADABAS
  - IDMS
  - IMS
  - VSAM

启动组件集成服务：

- 为您选择访问的外部数据源 （例如 Oracle、 Informix、 Microsoft SQL Server）安装 DirectConnect 服务器或网关。
- 按照中的说明配置服务器以访问远程对象。第 2 章 “了解组件集成服务”。

第 2 章

了解组件集成服务

本章说明如何使用组件集成服务。目的是帮助您了解 Adaptive Server 如何使用配置的 “组件集成服务”选项。

基本概念

如同访问本地表一样访问远程 （或外部）表的功能是组件集成服务的一个特点。组件集成服务将表提供给客户端应用程序，如同在本地存储表中的所有数据一样。远程表映射到存储元数据的本地代理表。在内部，执行涉及远程表的查询时，确定存储位置并访问远程位置以检索数据。

用于检索远程数据的访问方法由外部对象的两个属性确定：

- 与远程对象相关联的服务器类
- 对象类型

若要达到位置透明度，必须首先将表映射到它们相应的外部位置。

访问方法

访问方法形成了服务器和外部对象间的接口。对于每个服务器类，应使用单独的访问方法来处理 Adaptive Server 与相同类和对象类型的远程服务器之间的所有交互。

服务器类

使用 sp_addserver 添加各服务器时，必须给每个服务器指派服务器类。服务器类决定了用于与远程服务器交互的访问方法。服务器类包括：

- *ASEnterprise* — 在服务器为 Adaptive Server 时使用。这是缺省服务器类。
- *ASAnywhere* — 在服务器为 Adaptive Server Anywhere 6.0 版或更高版本时使用。应当将此服务器类用于 Adaptive Server IQ 12.5 之前的 Adaptive Server IQ 版本。
- *ASIQ* — 在服务器为 Adaptive Server IQ 12.5 版及更高版本时使用。
- *local* — 本地服务器。只能有一个。
- *direct_connect* — 表示服务器是符合 DirectConnect™ 服务器接口要求的 Open Server™ 应用程序。为了访问 Microsoft SQL Server、DB2、 Oracle 或 Informix，您必须使用 DirectConnect 服务器。
- *sds* — 表示服务器符合 Specialty Data Store 的接口要求。
- RPCServer — 表示服务器只能处理 RPC。它不接受 SQL 语句、事务控制语句或任何其它命令。

对象类型

服务器将大量对象类型 （如同它们是本地表一样）提供给客户端应用程序。支持的对象类型包括：

- table — 任何类的远程服务器中的对象均是关系表。这是缺省类型。
- view — 任何类的远程服务器中的对象均是视图。组件集成服务将这些视图作为没有任何索引的本地表来处理。
- remote procedure — 任何类的远程服务器中的对象均是远程过程。组件集成服务将远程过程的结果集作为只读表来处理。
- file — 对象是文件系统中的单个文件。
- directory — 对象是一个文件系统目录。

远程服务器的接口

本地服务器和远程服务器间的接口由 Open Client 软件 Client-Library™ 来处理。用于实现接口的 Client-Library 功能取决于与组件集成服务交互的服务器的类。

例如，如果服务器类是 *direct_connect*，则使用大量的功能，如游标和动态请求。

在本地服务器能够与远程服务器交互之前，必须配置以下各项：

- 附加到 *directory services* 的远程服务器
- 远程服务器定义
- 远程服务器登录信息
- 远程对象定义

目录服务

使用组件集成服务访问远程表前，必须具有对 LDAP 目录服务或 *interfaces* 文件 （Windows 平台上的 *sql.ini* 文件）的访问权限。有关访问远程表的详细信息，请参见第 27 页的 “连接管理”。有关设置目录服务的信息，请参见所用平台的配置文档。请参见附录 A “教程”它可作为组件集成服务用户的基础教程。

远程服务器定义

通过存储过程 sp_addserver 来定义远程服务器。《参考手册》中介绍了这一过程。

登录到远程服务器

配置完远程服务器后，必须提供登录信息。缺省情况下，每当组件集成服务代表 Adaptive Server 客户端与远程服务器建立连接时，均使用这些客户端的名称和口令。不过，可以使用 sp_addexternlogin 替换这一缺省设置，以便系统管理员为连接到远程服务器的每个用户定义登录名和口令。

使用 connect to *server_name*，可以检验服务器配置是否正确。此命令建立与远程服务器的直通模式连接。直通模式允许客户端用本机语法与远程服务器通信。除非发出 disconnect 命令，否则此直通模式将一直有效。

定义远程对象

在配置了远程服务器后，除非已经在该远程服务器中的对象和本地对象（代理表）之间建立了映射，否则不能将这些对象作为表进行访问。

可以在远程服务器上创建新表，以及为远程服务器中的现有对象定义模式。这两个过程十分类似。

代理表

代理表是实现位置透明度的关键所在。代理表是包含指向远程对象的元数据的本地表。有关远程对象的信息，请参见第 4 页的 “对象类型”。远程表映射到代理表，以使该远程表以本地表的形式出现。

第 3 章 “SQL 参考”中完整描述了如何执行这一操作。

使用 *create table* 命令

create table 命令使用以下语法同时创建代理表和远程表：

create table table_name (column_list) [[external {table | file}] at “pathname”]]

远程位置是使用 at pathname 子句指定的。create table 允许使用外部类型 table 和 file。每列的数据类型传递给远程服务器而不进行转换。

使用 *create existing table* 命令

create existing table 命令允许对现有表 （代理表）进行定义。此选项的语法与 create table 命令类似：

```
create existing table table_name (column_list)
[[external {table | procedure | file}] at pathname]
```

但接收到此命令时服务器所采取的操作与接收到 create table 命令时所采取的操作明显不同。在远程位置上不创建新表；而是检查表映射并检验基础对象是否存在。如果对象不存在 （主机数据文件或远程服务器对象），则拒绝该命令，并显示错误消息。

如果对象存在，则获取其属性并将其用于更新系统表 sysobjects、syscolumns 和 sysindexes。

- 确定现有对象的性质。
- 对于远程服务器对象 （而不是 RPC），将找到的表或视图的列属性与 *column_list* 中定义的属性进行比较。列名必须匹配 （区分大小写），列类型和长度必须匹配 （或至少是可转换的），并且列的 NULL 属性必须匹配。
- 提取主机数据文件或远程服务器表中的索引信息，并将其用于创建系统表 sysindexes 的行。这将用服务器的术语来定义索引和键，并允许查询优化程序考虑该表中可能存在的任何索引。

成功定义现有表后，对该表发出 update statistics 命令。这使查询优化程序能够对索引选择和连接顺序做出正确的选择。

数据类型转换

使用 create table 或 create existing table 命令时，必须使用已识别的 Adaptive Server 数据类型来指定所有数据类型。如果远程服务器表位于异构服务器类中，则在检索数据时，自动将远程表的数据类型转换为指定的 Adaptive Server 类型。如果无法进行转换， create table 或 create existing table 命令将不允许创建或定义该表。

远程表定义的示例

以下示例从定义服务器开始定义远程 Adaptive Server 表 authors：

1　定义名为 SYBASE 的服务器。它的服务器类为 *ASEnterprise*，它在 interfaces 文件中的名称为 SYBASE：

```
exec sp_addserver SYBASE, ASEnterprise, SYBASE
```

2　定义远程登录别名。如果两个服务器上的 *username* 和 *password* 相同，则此步骤为可选步骤。用户 “sa” 在远程服务器 SYBASE 上使用用户名 “sa” 和口令 “timothy” 标识身份：

```
exec sp_addexternlogin SYBASE, sa, sa, timothy
```

3　定义远程 authors 表：

```
create existing table authors
(
au_id        varchar(11)      not null,
au_lname     varchar(40)      not null,
au_fname     varchar(20)      not null,
phone        char(12)         not null,
address      varchar(40)      null,
city         varchar(20)      null,
state        char(2)          null,
country      varchar(12)      null,
postalcode   char(10)         null
)
EXTERNAL TABLE at "SYBASE.pubs2.dbo.authors"
```

4　更新表的统计信息以确保查询优化程序进行合理的选择：

```
update statistics authors
```

5　执行查询以测试配置：

```
select * from authors where au_lname = 'Carson'
```

使用 create proxy_table 命令

使用 create proxy_table 命令不要求列列表。组件集成服务从它自远程表获得的元数据中派生出列的列表。

如果对象确实存在，则 create proxy_table 更新 sysobjects、 syscolumns 和 sysindexes。

将远程过程作为代理表

可以在 create existing table 语句中添加一个可选子句，表明该远程对象实际上是一个存储 （或其它）过程，而不是一个表。如果没有此子句，则将该远程对象假定为表或视图：

```
create existing table t1
    (
                column_1    int,
                column_2    int
    )
    EXTERNAL PROCEDURE AT "SYBASE.mydb.dbo.p1"
```

如果远程对象是类型过程，则会出现几种不同的处理方式：

- 不为此类型的对象创建任何索引。
- 必须提供与远程过程结果集的说明相匹配的列的列表。不对该列表的准确性进行任何检验。
- 可以使用以下划线 （‘_’）开头的列名来指定不属于远程过程结果集的列。这些列称为参数列。例如：

  ```
  create existing table t1
  (
      a       int,
      b       int,
      c       int,
      _p1     int null,
      _p2     int null
  )
  external procedure
  at “SYBASE.sybsystemprocs.dbo.myproc”

  select a, b, c from t1
  where _p1 = 10 and _p2 = 20
  ```

- 在此示例中，参数列 *_p1* 和 *_p2* 不应出现在结果集中，但可以在查询中引用。组件集成服务通过名为 *@p1* 和 *@p2* 的参数将搜索参数传递给远程过程。

- 如果参数列包含在 select 列表中，并且是作为参数传递给远程过程的，那么其值应等于在 where 子句中为其指定的值。如果参数列未出现在 where 子句中，或者不能作为参数传递给远程过程，而是包含在 select 列表中，则其值为 NULL。
- 如果 Adaptive Server 查询处理器认为某个参数列是可搜索的参数，则可以将该参数列作为参数传递给远程过程。如果它没有包括在任何 “or” 谓词中，那么它通常是可搜索的参数。例如，以下查询禁止将参数列用作参数。

  ```
  select a, b, c from t1
  where _p1 = 10 OR _p2 = 20
  ```

- 在 create existing table 语句中用于参数列定义的规则：
  - 参数列必须允许 NULL。
  - 参数列不能位于常规结果列之前 （它们必须出现在列的列表的末尾）。

  由于允许将远程过程作为本地表来定义，组件集成服务可以将远程过程的结果集作为 “虚拟表” 来处理。该表可以进行排序，与其它表连接，或通过 insert/select 语法插入到另一个表中。但是，虚拟表被视为只读：
  - 不能对 *procedure* 类型的表发出 delete、 update 或 insert 命令；
  - 不能对虚拟表发出 create index、truncate table 或 alter table 命令。

如果已经在服务器中定义了 procedure 类型的对象，将不会向该对象所在的远程服务器发出查询。相反，组件集成服务将发出一个 RPC，并将该 RPC 的结果作为只读表来处理。

示例

```
create existing table rtable
 ( col1   int,
   col2   datetime,
   col3   varchar(30)
 )
external procedure at “SYBASE...myproc“

select * from rtable
```

当发出此查询时，组件集成服务将名为 *myproc* 的 RPC 发送到服务器 SYBASE。行结果的处理方式与来自任何其它表的结果类似；可以将它们排序、与其它表连接、分组或插入到另一个表中，等等。

RPC 参数应表示限制结果集的参数。如果发出不带参数的 RPC，则返回该对象的整个结果集。如果发出带有参数的 RPC，则每个参数将进一步限制结果集。例如：

```
select * from rtable where col1 = 10
```

导致将名为 @col1 的单个参数与 RPC 一起发送。其值为 10。

组件集成服务尝试将尽可能多的搜索参数传递给远程服务器，但根据正在执行的 SQL 语句，组件集成服务可能会自行计算结果集。每个参数都表示一个完全匹配的搜索，例如 = 运算符。

以下规则定义发送到 RPC 的参数。如果 RPC 用作组件集成服务对象，则在开发期间应该牢记这些规则。

- 组件集成服务将 where 子句中的 = 运算符作为参数发送。例如，此查询导致组件集成服务发送两个参数：

  ```
  select * from rpc1 where a = 3 and b = 2
  ```

  参数 a 的值为 3，参数 b 的值为 2。RPC 应只返回结果行，其中列 a 的值为 3，列 b 的值为 2。

- 如果没有确切的搜索条件，组件集成服务不会发送 where 子句的任何参数或 where 子句的任何部分。例如：

  ```
  select * from rpc1 where a = 3 or b = 2
  ```

  由于是 or 子句，组件集成服务不会发送 a 或 b 的参数。

  另一个示例：

  ```
  select * from rpc1 where a = 2 and b < 3
  ```

  组件集成服务发送 *a* 和 *b* 的参数，并过滤包含 *b* 值小于 3 的行。

服务器限制

Adaptive Server 配置允许使用 2K、 4K、 8K 或 16K 个字节的页大小。另外，取消了 char/binary 列的 255 个字节的限制。Adaptive Server 支持 char、 varchar、 univarchar、 unichar、 binary 和 varbinary 数据类型的扩展大小。新限制取决于服务器的页大小。对于不同的页大小，新限制如下：

表 2-1：新限制

页大小	最大列大小
2048	2048
4096	4096
8192	8192
16384	16384

这些大小是近似值。基本规则指定限制是允许一页可容纳的单行的最大大小。这些限制还取决于创建表时所指定的锁定方案。假定批量代理表是使用缺省锁定方案 （即所有页锁定）创建的。

- Transact-SQL 变量和参数的长度限制 — char、 varchar、 binary 和 varbinary 变量的大小扩展为等于给定服务器中相同数据类型列的最大大小。这允许将变量传递给长度超过 255 个字节的当前限制的存储过程 （或 RPC）。
- 每个表的列数限制 — 只要页面仍可以容纳列，则每个表允许有多达 1024 列。可变长度列数的限制为 254 个 （空列也被视为可变长度列）。
- 索引的宽度限制 — 根据服务器页大小， Adaptive Server 中索引的总宽度可以大于以前版本中的宽度。表 2-2 按照页大小列出了最大索引宽度：

表 2-2：最大索引宽度

页大小	索引宽度
2048	600
4096	1250
8192	2600
16384	5300

- 每个索引的列数限制 — 每个索引 31 列。
- 表名、列名和索引名的长度最多可为 255 个字节。
- 标识符名现在最多可为 255 个字节。

create new proxy table

如上所述， create table 允许以扩展长度指定数据类型为 char、 varchar、 binary 和 varbinary 的列。这些数据类型和长度转移到创建表的远程服务器上。

create existing proxy table

create existing table 命令同样允许以大于 255 个字节的长度指定列。这允许组件集成服务将远程数据库中以前必须作为 text 或 image 列处理的列作为 char、 varchar、 binary 或 varbinary 来处理。

仍有可能会出现列大小不匹配的错误。例如，当远程数据库包含列长度为 5000 个字节的表，而处理 create existing table 的 Adaptive Server 只支持长度最多为 1900 个字节的列时，将发生大小不匹配错误。在这种情况下，必须重新将该列指定为 text 或 image 列。

当代理表的列大小超过远程表中相应列的大小时，将检测到大小不匹配错误并中止该命令。

create proxy_table

create proxy_table 从远程服务器导入元数据，将列信息转换为内部 create existing table 命令，并从导入的元数据派生一个列列表。获得列元数据时，必须从远程 DBMS 类型转换为内部 Adaptive Server Enterprise 类型。

如果远程 char、 varchar、 binary 或 varbinary 列的大小超过本地服务器所允许的最大值，则将其长度截短到可能的最大大小 （这取决于页大小）。如果大小超过 16K 字节，则将类型从 char 或 varchar 转换为 text，或者从 binary 或 varbinary 转换为 image。

alter table

如果对代理表执行 alter table 命令，则首先在本地对其进行处理，然后再转移到远程服务器上执行。如果远程执行失败，则返回本地更改并中止该命令。

远程服务器必须相应地处理命令，否则提交错误。如果出现错误，则中止并回退命令的组件集成服务端。

select、 *insert*、 *delete*、 *update*

当数据操作语言 (DML) 操作涉及代理表时，组件集成服务将处理较大的列值。组件集成服务使用以下策略之一来处理 DML：

- Tabular data stream (TDS)™ 语言命令 — 如果可以将整个 SQL 语句转移到远程服务器，则组件集成服务使用 CT-Library ct_command (CS_LANG_CMD) 生成的 TDS 语言命令来执行此操作。

  语言缓冲区的文本可能包含超过 255 个字节的 long char 或 binary 值数据，远程服务器必须处理这些命令缓冲区的语法分析。

- TDS 动态命令 — 如果组件集成服务不能将整个 SQL 语句转移到远程服务器 （例如，强制组件集成服务为语句提供功能性补偿），则可以使用 TDS 动态命令以及所需的参数或使用 CT-Library 函数 ct_dynamic (CS_PREPARE_CMD, CS_EXECUTE_CMD, CS_DEALLOC_CMD) 来处理 insert、 update 或 delete。

  该动态命令的参数可能为 CS_LONGCHAR_TYPE 或 CS_LONGBINARY_TYPE。

- TDS 游标命令 — 如果必须执行功能性补偿，则可以使用 CT-Library 游标操作来处理 select、 update 和 delete 的代理表操作。例如，如果更新代理表，而 from 子句中包含多个表，则组件集成服务可能需要从多个数据源读取行，并且为每个限定行对目标表应用 update。在这种情况下，组件集成服务使用 ct_cursor ({CS_DECLARE_CMD, CS_OPEN_CMD, CS_CURSOR_UPDATE_CMD, CS_CLOSE_CMD, CS_DEALLOC_CMD})。

  准备好游标之后，请指定参数。这些参数可能包含 CS_LONGCHAR 或 CS_LONGBINARY 类型的参数。

- 批量插入命令 — 执行 select/into 操作时，如果目标服务器支持批量接口 （仅适用于远程 Adaptive Server 和 DirectConnect for Oracle），则必须准备好远程服务器以处理大于 255 的 char 和 binary 值（通过 CS_LONGCHAR、 CS_LONGBINARY 值）。

可以将来自远程服务器的列作为 CS_LONGCHAR_TYPE 或 CS_LONGBINARY_TYPE 类型返回到组件集成服务。

RPC 处理

发送到远程服务器的 RPC 可包含 CS_LONGCHAR 和 CS_LONGBINARY 类型的参数。组件集成服务命令 cis_rpc_handling 支持这些类型。

由于早期版本的 Adaptive Server 不支持 CS_LONGCHAR 或 CS_LONGBINARY 数据，因此不允许将长参数发送给早于 12.5 版的 Adaptive Server。在发送 RPC 以前，组件集成服务检查远程服务器的 TDS 功能，如果远程服务器不能接受这些数据类型，则出现错误。

sp_tables

Adaptive Server Anywhere 或 ASIQ 存储过程 sp_tables 仅返回用户表。

级联代理表

Adaptive Server 允许在任意数量的组件集成服务实例之间进行级联代理表配置。

在某些条件下，这可能会出现问题，如循环引用或事务 （其中，第二个代理表在与第一个代理表相同的服务器上引用本地表）。在这种情况下，可能会导致组件集成服务检测不到的应用程序死锁。必须对系统进行配置以避免这些潜在的缺陷。

代理数据库

共有两种类型的代理数据库：用户和系统。

用户代理数据库

创建用户代理数据库时，将自动从包含实际表的远程位置导入代理表的元数据。然后，使用这些元数据在代理数据库中创建代理表。

若要创建代理数据库，请使用：

```
create database <dbname>
    [create database options]
    [with default_location = 'pathname'
    [for proxy_update]]
```

可以使用 with default_location 子句来指定任何新表的存储位置，如果还指定了 for proxy_update 子句，则还会指定在自动创建代理表时从中导入元数据的位置。 for proxy_update 将数据库指定为代理数据库； with default_location 定义从中导入代理表的位置。如果未指定 for proxy_update，with default_location 的行为与 sp_defaultloc 所提供的行为相同 — 使用缺省的存储位置来创建新表和现有表，但在处理 create database 命令的过程中不会自动导入代理表定义。

路径名的值是具有如下格式的字符串标识符：*servername.dbname.owner*。

- *servername* — 必需字段；表示拥有代理表要引用的对象的服务器的名称。必须存在于 *master.dbo.sysservers.srvname* 中。
- *dbname* — 可选。 *servername* 中包含代理表要引用的对象的数据库的名称。

- *owner* — 可选。代理表要引用的对象的所有者名。这可能是有限制性的，因此，如果多个用户拥有 *dbname* 中的对象，则指定所有者将仅选择该用户所拥有的那些对象。不会为其它用户所拥有的对象创建代理表。

如果指定 for proxy_update 时不指定 default_location，则报告错误。

创建代理数据库 （使用 for proxy_update 选项）后，将调用组件集成服务函数来执行以下操作：

- 提供包含所有代理表所需的数据库大小的估计值，这些代理表表示在主服务器的数据库中找到的实际表/视图。此估计值根据包含所有代理表和索引所需的数据库页数来提供。如果未指定大小和数据库设备，则使用此大小。

  注释　如果用指定的特定大小来创建数据库 [on device_name = nn]，或者指定设备名而未指定大小 [on device_name]，则不估计代理数据库的大小要求；在这种情况下，假定用户或数据库管理员要替换为代理数据库计算的缺省大小。

  如果正从其它 Adaptive Server 导入元数据，则在创建代理表前导入远程数据库用户。每个导入数据库的用户必须在 syslogins 中拥有相应的系统用户名。

- 创建所有表示在协同服务器的数据库中找到的实际表/视图的代理表。不为系统表创建代理表。
- 将代理表的所有权限授予 “public”。
- 将 “guest” 用户添加到代理数据库。
- 从远程节点导入数据库用户 （如果是 Adaptive Server）。
- 将 create table 权限授予 “public”。
- 设置数据库状态以表示此数据库是用户代理数据库。通过设置 *master.dbo.sysdatabases.status*3 (0x0001, DBT3_USER_PROXYDB) 中的状态字段来完成此操作。

数据库创建完毕后，它包含在缺省位置中找到的每个表或视图的代理表。用户代理数据库的行为与以前数据库行为完全相同。用户可以创建额外的对象，如 procedure、 views、 rules、 defaults 等，对代理表执行操作的 DDL 和 DML 语句均按本文中所描述的方式工作。

唯一的例外是 alter database 命令。将在下一节中介绍该命令的语法和功能。

用户代理数据库模式同步

有时，DBA 可能必须强制对代理数据库中包含的代理表进行重新同步。可以使用 alter database 命令来完成这一过程：

```
alter database <dbname>
        [alter database options]
        [for proxy_update]
```

如果输入 for proxy_update 子句时没有指定其它选项，则不会扩展数据库的大小；而是从代理数据库中删除代理表 （如果有）并用元数据重新创建，该元数据是从执行 create database ... with default_location = 'pathname' 时所指定的 *pathname* 中获取的。

如果使用 create database 和其它选项来扩展数据库的大小，则在扩展大小后将执行代理表同步。

此 alter database 扩展的目的是为 DBA 提供使用方便的单步操作，通过该操作可获得单个远程节点上所有表的精确且最新的代理表示。

所有的外部数据源都支持此重新同步，而不仅仅是 HA 集群环境中的主服务器。而且，不需要首先使用 for proxy_update 子句来创建数据库。如果通过 create database 命令或使用 sp_defaultloc 指定了缺省存储位置，则可以将数据库中的元数据与远程存储位置上的元数据同步。

使用 create/alter database 指定代理数据库时，将隐式启用某些行为：

- 对使用 create database 命令指定的缺省位置的修改不允许使用 alter database。
- 不能在代理数据库中创建本地表。 create table 命令将创建代理表，并且在缺省位置创建实际表。
- 可以在 create table 命令中使用 at 'pathname' 语法来指定表的缺省位置。如果路径名与缺省位置不同，则 alter database 命令无法同步该表的元数据。
- 若要更改缺省位置，请删除该数据库，然后使用 default_location = 'pathname' 子句中指定的新路径名重新创建该数据库。如果使用 sp_defaultloc 更改了位置，则使用新位置来提供元数据同步，但不会对用以前位置创建的代理表进行同步，如果其名称与新位置上表的名称冲突，则可能会删除和替换代理表。

系统代理数据库

系统代理数据库的行为与用户代理数据库类似，但有一些明显的改进和例外情况。系统代理数据库只用于 HA 配置中。

系统代理数据库允许在高可用性集群系统的任一节点上运行客户编写的应用程序。这并不表示 “单系统映像” 功能；它表示这样一种环境：在这种环境中，可以在集群系统的任一节点上执行大多数用户编写的应用程序。这意味着，数据库和用户创建的对象应对所有节点可见。

系统代理数据库与其引用的主节点中的数据库具有相同的名称，并且包含对支持此应用程序所需的用户定义对象的处理。将为主数据库中找到的每个用户表和视图创建代理表，将存储过程转换为 RPC 并转移到代理数据库引用的节点。

系统代理数据库创建

在以下情况下，自动创建系统代理数据库：

- 通过使用存储过程 sp_companion ServerName, ’configure’, with_proxydb 来配置 HA 集群。

  在这种情况下，为在以 ServerName 表示的服务器中找到的每个用户数据库创建一个系统代理数据库。

- 在其 HA 状态为 MODE_APNC、MODE_SNC 或 MODE_ASNC 之一的服务器中发出 create database 命令。

创建完系统代理数据库后，将调用组件集成服务函数来执行以下操作：

- grant create table to public — 该命令允许在主服务器上创建表时导致在系统代理数据库中创建代理表。

当前数据库具有系统代理数据库时的模式同步

在 HA 集群中，必须将对主服务器的数据库所做的某些更改转移到协同服务器，以使两个服务器保持同步。

在具有系统代理数据库的数据库中执行某些 DDL 命令时，将导致通知协同服务器并自动对相应更改进行同步：

- create table 和 drop table — 执行本地操作，导致创建或删除本地表。此命令随即转移到协同服务器以在系统代理数据库中执行，因此可以创建或删除代理表。

- create index 和 drop index — 执行本地操作，导致创建或删除索引。随即通知拥有系统代理数据库的服务器，并且删除和重新创建代理表，允许在代理表中反映对索引的更改。
- create view 和 drop view — 成功执行本地操作，导致创建或删除本地视图。随即通知拥有系统代理数据库的服务器，并创建或删除代理表。

如果在系统代理数据库中执行这些命令，将发生以下类似行为：

- create table 和 drop table — 创建或删除本地代理表。该命令随即转移到主服务器，因此可创建或删除代理表所引用的本地表。
- create index 和 drop index — 在代理表上执行本地操作，导致创建或删除索引。随即通知拥有主数据库的服务器，并在代理表引用的本地表上创建或删除索引。
- create view 和 drop view — 不允许在系统代理数据库中使用。

在系统代理数据库中执行存储过程

当前数据库为系统代理数据库时，如果遇到系统存储过程请求，组件集成服务将首先尝试在本地 sybsystemprocs 数据库中找到该存储过程，然后执行它。如果在 sybsystemprocs 中未找到该存储过程，组件集成服务将搜索 master 数据库。如果该过程不是系统存储过程，或者它虽然是系统存储过程，但无法在本地找到它，则会将存储过程请求转换为 RPC 并将其传输到系统代理数据库缺省位置引用的服务器。

系统代理数据库的其它行为

在系统代理数据库中执行某些命令时，将拒绝执行命令并显示错误：

- create procedure 和 drop procedure
- create view 和 drop view
- create trigger 和 drop trigger
- create rule 和 drop rule
- create default 和 drop default

这些情况下生成的错误为：`Msg 12818, Severity 16: Cannot create an object of this type in system-created proxy database`。

文件系统访问

注释　映射到代理表的目录和文件现在具有 255 个字节的文件路径限制。

Adaptive Server 通过 SQL 语言提供对文件系统的访问。通过文件系统访问，可以创建映射到文件系统目录或各个文件的代理表。

若要创建映射到目录或文件的代理表，必须拥有系统管理员或系统安全员特权。

安全注意事项

只有具有系统管理员 (sa) 或系统安全员 (sso) 角色的 Adaptive Server Enterprise 用户才允许创建映射到文件或目录的代理表。这些要求考虑了从 Adaptive Server Enterprise 服务器进程（可能在运行时具有根权限）内部访问文件系统数据的安全方面的问题。

目录访问

可以创建代理表来引用文件系统目录。支持的语法为：

```
create proxy_table <table_name>
external directory at "directory pathname[;R]"
```

目录路径名必须引用对 Adaptive Server Enterprise 进程可见并可由其搜索的文件系统目录。将创建一个代理表，它将列名映射到该目录中存在的文件的属性。如果路径名的末尾附加有 ‘;R’（表示 “递归”）扩展，则组件集成服务包括所有下级目录中的条目。表 2-3 说明了在成功完成此命令时创建的代理表列：

表 2-3：代理表列

列名	数据类型	说明
id	numeric(24) — 在 32 位计算机上 numeric(36) — 在 64 位计算机上	由 st_dev 和 st_ino 中的值组成的标识值。首先将这两个值转换为单个字符串 （格式为：“%d%014ld”），然后将字符串转换为数字值。
filename	varchar(n)	at ‘pathname’ 指定的目录或路径名的下级目录中的文件的名称。文件名的总长度限制为 255 个字节。
size	int	对于常规文件 — 指定文件中的字节数。 对于目录文件 — 块特殊或字符特殊文件，未定义。
filetype	varchar(4)	文件类型 – 合法值为：管道文件为 FIFO ；目录为 DIR ； 字符特殊文件为 CHRS ； 块特殊文件为 BLKS ； 普通文件为 REG ； 所有其它文件类型为 UNKN。将自动扩展链接，并且链接不作为单独的文件类型出现。
access	char(10)	访问权限，以较 “标准” 的 UNIX 格式提供：“drwxrwxrwx”
uid	varchar(n)	文件所有者的名称。 n 的值由系统定义 L_cuserid 指定，该值在所有 UNIX 系统上均为 9。此值在 Windows 系统上为 0。
gid	varchar(n)	文件所有组的名称。 n 的值由系统定义 L_cuserid 指定，该值在所有 UNIX 系统上均为 9。此值在 Windows 系统上为 0。
atime	datetime	上次访问文件数据的日期 / 时间。
mtime	datetime	上次修改文件的日期 / 时间。
ctime	datetime	上次更改文件状态的日期 / 时间。
content	image	文件的实际物理内容 （仅适用于常规文件）。如果文件为非常规文件，则为 NULL。

映射到文件系统目录的代理表可以支持下列 SQL 命令：

- select — 可使用 select 命令从代理表中获得文件属性和内容。内容列完全支持用于处理文本值的内置函数 （例如 textptr、 textvalid、 patindex、 pattern）。
- insert — 可使用 insert 命令创建新的文件或目录。只有 filename、 filetype 和 content 列有意义。应该在 insert 语句中省略其余列，如果找到这些列，则将其忽略。如果文件类型是 *DIR* （指示将创建新目录），则忽略 content 列。

若要创建新的目录，请输入：

```
insert D1 (filename, filetype) values ("newdir",
"DIR")
```

若要创建新的文件，请输入：

```
insert D1 (filename, content) values
("newdir/newfile", "This is an example.")
```

- delete — 可使用 delete 命令删除文件或目录。只有在目录为空时，才能将其删除。例如：

```
/* delete the files only */
delete D1 where filename = 'newdir/newfile'
/* deletes the directory (if empty) */
delete D1 where filetype = 'DIR' and filename =
'newdir'
```

- update — 只能使用 update 命令更改文件的名称。

  注释　有些文件系统可能执行 update 作为删除，然后创建一个新的目录条目，因此，在不对 update 命令进行限制的情况下，相同的文件名可能被更新多次。Sybase 建议您通过将文件名包含在 update 命令的 where 子句中来限定对特定文件执行 update。例如，下面的语句可能会导致多次更新：

  ```
  update t1 set filename=filename + 'old' where filetype
  = 'REG'
  ```

  可以通过添加如 `"and filename like "%.c"` 这样的子句来避免该问题。

- readtext — 可以使用 readtext 命令检索文件的内容。
- writetext — 可以使用 writetext 命令修改文件的内容。

不能对代理表执行其它 SQL 命令。

只有在 Adaptive Server 进程具有足够的特权来访问和读取文件，并且文件类型表明是 “普通”文件时，常规文件内容才可用。在所有其它情况下，内容列为 NULL。例如：

```
select filename, size, content
    from directory_table
    where filename like '%.html'
```

如果 Adaptive Server 进程拥有常规文件的访问特权，则返回具有 “.html” 后缀的常规文件的名称、大小和内容。否则，内容列将为 NULL。

如果目录路径名引用的路径名不是目录或者 Adaptive Server Enterprise 进程不能对其进行搜索， create proxy_table 将失败。

如果已打开跟踪标志 11206，则将消息写入错误日志中，日志包含有关目录内容的信息以及获得该信息所需的查询处理步骤。

通过下级目录递归

如果在 create proxy_table 语句中指定的路径名包含 *;R* 扩展，则组件集成服务将遍历路径名的所有下级目录，并返回每个下级目录内容的信息。此过程完成后，查询返回的文件名包含相对于该路径名的完整文件名。换言之，所有下级目录名称均出现在文件名中。例如，如果路径名指定 *"/work;R"*：

```
create proxy_table d1 external directory at "/work;R"
select filename, filetype from d1
```

返回下级目录中文件的值如表 2-4 所示：

表 2-4：文件值

文件名	文件类型
dir1	DIR
dir1/file1.c	REG
dir1/file2.c	REG
dir2	DIR
dir2/file1.c	REG

文件访问

Adaptive Server 中允许使用的另一个代理表类可使 SQL 访问文件系统中的各个文件。支持的语法为：

```
create proxy_table <table_name>
    external file at " pathname" [column delimiter "<string>"]
```

使用此命令时，将创建具有一列 （名为 "record"，类型为 varchar(255)）的代理表。在这种情况下，假定文件的内容为可读字符，并且以换行符 (\n) 来分隔文件中的各个记录。

也可以使用 create [existing] table 命令，指定您自己的列名和数据类型：

```
create existing table fname (
    column1 int  null,
    column2 datetime null,
    column3 varchar(1024)  null
    etc. etc.
) external file at "pathname" [column delimiter “<string>”]
```

列可以为除 text、 image 或 Java ADT 以外的任何数据类型。 existing 关键字的使用是可选的，并且不影响语句的处理。如果路径名引用的文件不存在，则创建该文件。如果文件存在，则不会覆盖其内容。 create table 和 create existing table 命令的行为没有区别。

如果将代理表映射到文件，则对该文件及其内容作出以下假定：

- 该文件不是目录文件、块特殊文件或字符特殊文件。
- Adaptive Server 进程至少具有文件的读取访问权限。如果要创建文件，服务器进程必须对要在其中创建文件的目录具有写入访问权限。
- 现有文件的内容为人工可读格式。
- 文件中的记录以换行符分隔。
- 支持的最大记录大小为 32767 个字节。
- 各个列 （最后一列除外）以 column delimiter 字符串分隔，最大长度为 16 个字节；缺省为单个制表符。
- 文件每个记录中的分隔值和代理表中的列之间是对应的。

通过将代理表映射到文件，您可以：

- 使用 select/into 或 insert/select 将数据库表备份到文件系统。处理 insert 语句时，每列均被转换为服务器缺省字符集中的字符。转换结果被存入缓冲区，并且所有列 （最后一列除外）均以单个制表符分隔。最后一列以换行符终止。随后将缓冲区写入文件，表示一行数据。
- 提供使用 bcp in 和 bcp out 的一种 SQL 替代形式。可以使用 select/into 语句，方便地将表备份到文件或将文件内容复制到表中。
- 使用 select 语句查询文件内容，并根据需要用搜索参数或函数限定行。例如，可以读取 Adaptive Server 错误日志文件中的各个记录：

```
create proxy_table errorlog
    external file at "/usr/sybase/ASE15_0/install/errorlog"
select record from errorlog where record like "%server%"
```

此查询将从与 like 模式相匹配的文件返回所有行。如果行长度大于 255 个字节，则将它们截断。可以通过输入以下命令指定更长的行：

```
create existing table errorlog
(
    record    varchar(512) null
)
external file at "/usr/sybase/ASE15_0/install/errorlog"
```

在这种情况下，将返回最长为 512 个字节的记录。由于代理表只包含一列，每列的实际长度将由换行符的位置来确定。

文件访问只支持 select、 insert 和 truncate table 语句。如果文件代理是这些命令的目标， update 和 delete 将出现错误。

将值插入文件时，首先将所有数据类型转换为 char 值，然后再由列分隔符分隔。

警告！ truncate table 将文件大小设置为 0。

跟踪标志 11206 用于将消息记录到错误日志中。这些消息包含涉及文件访问的查询处理阶段的信息。

远程服务器

对于本地服务器和要调用的每个远程服务器，使用 sp_addserver 将条目添加到 sysservers 表中。 sp_addserver 语法为：

sp_addserver *server_name [,server_class [,network_name]]*

其中：

- *server_name* 是用于标识服务器的唯一名称。

- *server_class* 是服务器的类型。以下几节中说明支持的服务器类和每个类中服务器的类型。缺省设置为服务器类 ASEnterprise。

  注释　组件集成服务不再支持服务器类 db2。

- *network_name* 是 interfaces 文件中的服务器名。此名称可以与 *server_name* 相同，也可以不同。 *network_name* 有时称为*物理名*。缺省值为与 *server_name* 相同的名称。

  注释　在各服务器上需要有相同的排序顺序和大小写。

服务器类 ASEnterprise

Adaptive Server 使用服务器类 ASEnterprise。当组件集成服务与此类中的服务器首次建立连接时，组件集成服务确定 Adaptive Server 版本并基于找到的版本确定服务器功能。

服务器类 ASAnywhere

具有服务器类 ASAnywhere 的服务器是 Adaptive Server Anywhere 的一个实例：

- Adaptive Server Anywhere 9.0 或更高版本

服务器类 ASIQ

具有服务器类 ASIQ 的服务器为 Adaptive Server IQ 12.5 版或更高版本。

服务器类 *direct_connect*

具有服务器类 direct_connect 的服务器是一个基于 Open Server 的应用程序，该应用程序符合 direct_connect 接口规范。

使用服务器类 direct_connect 且基于 Open Server 的应用程序是访问所有外部非 Sybase 数据源的首选方式。

已启用组件集成服务的 Adaptive Server 可与客户端和基于 Open Server 的应用程序进行交互。例如，客户端应用程序可与 Adaptive Server 上的 CIS 交互， CIS 通过网络与 DirectConnect （如 Oracle 或 Informix）交互。 DirectConnect 也可以直接访问客户端应用程序，客户端应用程序也可以直接访问 DirectConnect。

服务器类 *sds*

具有服务器类 sds 的服务器符合 Adaptive Server Specialty Data Store Developer's Kit （《Adaptive Server Specialty Data Store 开发人员工具包》）手册中所描述的 Specialty Data Store™ 接口要求。 Specialty Data Store 是用于与 Adaptive Server 进行交互的 Open Server 应用程序。

服务器类 RPCServer

配置了 RPCServer 类的服务器无法处理 RPC 以外的任何命令。此类中的服务器无法参与分布式事务，组件集成服务也不会尝试向配置了此类的服务器发送 SQL 语句。

要向 Backup Server 或 XP Server 发送 RPC，必须启用 sp_serveroption negotiated logins 和 server logins。

通常，此类中的服务器是客户编写的 Open Server 应用程序，这些应用程序用于执行自定义操作或使作为一次或多次 RPC 的结果生成的数据对 Adaptive Server 应用程序可用。此类型的应用程序仅需要以下 Open Server 事件处理程序：

- SRV_CONNECT — 处理和鉴定来自 CIS 或客户端应用程序的登录请求。
- SRV_DISCONNECT — 处理来自 CIS 或客户端应用程序的断开连接请求。
- SRV_ATTENTION — 处理来自 CIS 或客户端应用程序的取消请求。

- SRV_RPC — 处理来自 CIS 的 RPC。处理 RPC 可能生成一个结果集，CIS 会将该结果集转移到 ASE 客户端（CIS 代表此客户端转移 RPC）。

使用此服务器类，可以编写支持映射到 RPC 代理表的 CIS 代理表的 Open Server 应用程序：

```
create existing table myRPCtable
      (
            <column description(s)
      )
      external procedure at 'myRpcServerName...rpcname'
```

连接管理

代表客户端连接到远程服务器时，组件集成服务使用 Client-Library 函数。为给定客户端首次建立与远程服务器的连接后，除非客户端与组件集成服务断开连接，否则连接将始终保持打开。

注释　当连接第一次使用组件集成服务功能时，它会与一个 Adaptive Server 引擎密切连接。 PSS 标志 POMNI_AFFINITY_SET 会被设置并且不会被自动清除。

不通过 interfaces 文件连接到远程服务器

不使用目录服务 *ldap* 中的相应条目或 *interfaces* 文件即可建立与远程服务器的连接。组件集成服务通过使用 CT-Library 连接属性 CS_SERVERADDR 来实现这一过程，该属性允许按以下格式指定服务器：

```
"hostname.domain.com:99999"
"hostname:99999"
"255.255.255.255:99999"
```

其中 99999 为端口号，*hostname* 表示为简单名称、IP 地址或完整的域名。

按如下格式输入名称 （使用带网络名参数的 sp_addserver）：

```
sp_addserver S1, ASEnterprise,
"myhost.sybase.com:11222"
```

或：

```
sp_addserver S1, ASEnterprise, "192.123.321.101:11222"
```

网络名的这种用法具有一些限制：

- Adaptive Server 节点处理器无法识别此语法。
- Replication Agent 线程无法识别此语法。

如果使用了此语法，则无论配置了 interfaces 文件还是 LDAP 服务器，CT-Library 都不会尝试从目录服务中查找连接信息。

如果配置了 SSL 而且拥有指向服务器文档中 SSL 部分的指针，则可使用可选的 SSL 语法：

```
"hostname.domain.com:99999:SSL"
"hostname:99999:SSL"
"255.255.255.255.99999:SSL"
```

有关为 SSL 配置 Adaptive Server 的详细信息，请参见 《系统管理指南：第一卷》中的第 19 章 “数据的保密性”。

LDAP 目录服务

LDAP 目录服务意味着不需要在客户端和服务器中使用 interfaces 文件。Adaptive Server 和组件集成服务均支持 LDAP 服务以获得服务器信息。当尝试建立与远程服务器的连接时，组件集成服务指示 Open Client 软件引用 interfaces 文件或 LDAP 服务器，除非 sp_addserver 的网络名参数包含冒号 (:)。

仅当配置文件 (*libtcl.cfg*) 指定 LDAP 服务时，组件集成服务才会使用它。可在 *$SYBASE/$SYBASE_OCS/config/libtcl.cfg* 或 *$SYBASE/$SYBASE_OCS/config/libtcl64.cfg*（对于 64 位应用程序）中找到 *libtcl.cfg*。

注释 当在 *libtcl.cfg* 中指定了 LDAP 服务器时，只能从 LDAP 服务器中获得服务器信息，并且 Adaptive Server 和组件集成服务忽略任何 （传统的） interfaces 文件。

与 SSL 的安全通信

通过使用 SSL，可以建立从组件集成服务到支持 SSL 协议 （Adaptive Server 和某些 DirectConnect）的任何数量的远程服务器的安全连接。

组件集成服务按以下方式处理 SSL 连接：

- 受托根文件的位置已确定。如果当前服务器已启用 SSL，则所有外发组件集成服务连接将使用与 Adaptive Server Enterprise 相同的受托根文件。
- 如果当前服务器已启用 SSL，将建立连接属性以定义 Open Client 回调，该回调用于响应来自启用 SSL 的远程服务器的询问。如果当前服务器未启用 SSL，则所使用的回调无法建立与启用 SSL 的远程服务器的任何连接。

受托根文件

当将受托根文件正确地添加到系统时，受托根文件包含对其它服务器的认证，本地服务器将这些服务器作为受托服务器对待。如果已定义了 $SYBASE_CERT，则本地服务器 (Adaptive Server) 可以在以下文件中访问受托根文件：

```
$SYBASE_CERT/trusted.txt
```

否则它位于：

```
$SYBASE/$SYBASE_ASE/certificates/servername.txt
```

在 UNIX 平台上：

```
%SYBASE%\%SYBASE_ASE%\certificates\servername.txt
```

在 Windows 平台上：

其中 *servername* 是当前 Adaptive Server 的名称。

使用 Kerberos 的安全通信

Kerberos 的基于网络的鉴定是一种单点登录功能，能让 Kerberos 客户端通过 Kerberos 系统进行鉴定，从而能够连接到任何支持 Kerberos 鉴定的应用程序。由于存储了一个中央口令，因此无需指定口令即可连接到支持 Kerberos 的应用程序。

受 Adaptive Server 支持的 Kerberos 第 5 版还提供一个叫做凭据委托或票据转发的功能，可让 Kerberos 客户端在连接到服务器时委托凭据，从而允许服务器初始化 Kerberos 鉴定以便代表 Kerberos 客户端进一步连接到其它服务器。

凭据委托功能当前仅针对 MIT Kerberos GSSAPI 库 4.x 版和更高版本进行了认证。客户端在连接到 Adaptive Server 之前必须先从 Kerberos 系统获取可委托的凭据 （在 UNIX 系统上使用 kinit -f 选项）。

连接到 Adaptive 服务器的 Kerberos 客户端可以请求对 Adaptive Server 进行远程过程调用 (RPC)，而且在对远程 Adapter Server 进行常规的分布式查询处理请求时，使用 Kerberos 凭据委托功能通过 CIS 实现。基于节点处理器的远程服务器连接不支持 Kerberos 鉴定。

若要使用 Kerberos 统一登录，系统安全员可以使用以下命令为远程 Adaptive Server 的 CIS 启用 Kerberos 安全性机制。

sp_serveroption [*server*, *optname*, *optvalue*]

例如，当当前登录用户使用 Kerberos 机制进行鉴定时，以下在本地服务器 S1 上执行的命令对与远程服务器 S2 的连接启用 Kerberos 鉴定。

```
sp_serveroption s2, "security mechanism", csfkrb5
```

Kerberos 安全服务

一旦对与远程 Adaptive Server 的连接启用了 Kerberos 安全性机制，便可使用 Kerberos 提供的下列一个或多个安全服务：

- 消息保密性

  通过网络对数据进行加密，以防止非授权泄露。

- 消息完整性

  验证通信在传输过程中未被修改。

- 相互鉴定

  验证客户端和服务器的标识。启动远程连接的本地服务器，可以请求对所有针对 Adaptive Server 的远程连接请求进行相互鉴定。这能让客户端验证远程服务器的身份。

注释 Kerberos 提供的可选安全服务在缺省情况下处于禁用状态。

消息保密性

在本地服务器 S1 上执行的以下命令为所有使用 Kerberos 鉴定与远程服务器 S2 建立的连接设置消息保密性。

```
sp_serveroption s2, "use message confidentiality", true
```

消息完整性

在本地服务器 S1 上执行的以下命令为所有使用 Kerberos 鉴定与远程服务器 S2 建立的连接设置消息完整性。

```
sp_serveroption s2, "use message integrity", true
```

消息鉴定

在本地服务器 S1 上执行的以下命令为所有使用 Kerberos 鉴定与远程服务器 S2 建立的连接设置相互鉴定。

```
sp_serveroption s2, "mutual authentication", true
```

远程 Adaptive Server Kerberos 主体名的配置

Adaptive Server 主体名为服务器的缺省名称。由于 Adaptive Server 的主体名可以不同于服务器名，因此，系统安全员可以为每台远程服务器指定服务器主体名。

以下命令为远程服务器 S2 指定远程 Adaptive Server 主体名。

```
sp_serveroption S2, "server principal", ase2@myrealm.com
```

组件集成服务远程过程调用的配置

CIS 使用持久 client-library 连接来处理 RPC 请求。 CIS 通过确定客户端是否已与 RPC 要传递到的服务器建立了 client-library 连接来处理外发 RPC。如果不存在连接，则会建立连接。

若要启用 CIS RPC 处理机制，请将配置选项 cis rpc handling 设置为 1。如果未启用， Kerberos 用户就必须临时启用 CIS RPC，才能让当前会话使用此功能。

以下命令为当前登录会话启用 CIS RPC 处理。

```
set cis_rpc_handling on
```

为组件集成服务远程过程调用建立安全性

下面讲述如何为所有类型的 Adaptive Server 到 CIS 连接启用 Kerberos 鉴定。

在下面的示例中，user1 是一个 Kerberos 用户，他登录到 Adaptive Server S1 并请求对远程 Adaptive Server S2 进行 RPC。

1　向 interfaces 文件或目录服务中添加一个对应于服务器 S1 和 S2 的条目以及对应于安全性机制的 secmech 行。

2 为 Kerberos 用户添加登录名 （如果不存在）。

```
create login user1 with password pwuser1
```

3 通过将配置选项设置为开启以启用安全性机制。

```
sp_configure "use security services", 1
```

4 在本地服务器 S1 上，为对远程服务器 S2 执行的 CIS 启用 Kerberos 鉴定。

注释 这假定远程服务器 S2 仅从 S1 接收 CIS 命令请求。但是，如果 S2 还请求对其它服务器执行的 CIS 命令，并且要求启用 Kerberos 鉴定，则 S2 上需要类似的配置。

a 在本地服务器 S1 上，添加远程服务器 S2。

```
sp_addserver S2
```

b 在 S1 上为对 S2 进行的外发 RPC 请求启用 Kerberos 安全性机制。以下命令为当前登录会话启用 CIS RPC 处理。

```
sp_serveroption S2, "security mechanism", csfkrb5
```

- 只有在使用上述命令为远程服务器 S2 配置了安全性机制时，才会启动对远程服务器 S2 进行的 RPC 请求的安全性机制鉴定。
- 如果配置了针对 S2 的安全性机制，并且用户 'user1' 的转发票据不可用于本地服务器 S1，则不启动 RPC 连接。
- 如果没有为对远程服务器 S2 进行的 RPC 配置安全性机制，意思是未设置服务器选项 security mechanism，则不会为对远程服务器 S2 进行的 RPC 启动安全性会话的建立。这种情况下， CIS 将会针对远程服务器 S2 启动基于口令的鉴定。如果没有指定外部登录名/口令映射，则只要客户端与远程服务器连接， CIS 就会使用客户端的名称和口令。使用 Kerberos 统一登录鉴定登录到 Adaptive Server 的客户端 user1 将没有任何口令与登录会话相关联。系统管理员必须使用用户 user1 的 sp_addexternlogin 来配置登录映射，才能连接到 S2 进行 RPC 请求。如果没有为 user1 指定外部登录名和口令映射，则口令将为空，与远程服务器之间的 CIS 连接将会失败。
- user1 可以通过连接到服务器 S1 并请求凭据委托来请求在 S2 上执行存储过程 sp_who，如下所示：

```
isql -Vd -S S1
S2...sp_who
```

- user1 可以使用以下命令建立与远程服务器 S2 的直通连接：

  ```
  connect to S2
  ```

- user1 可以使用以下命令在远程服务器上执行查询：

  ```
  sp_remotesql S2, "select @@authmech"
  go
  sp_remotesql S2, “sp_who”
  go
  ```

安全问题

建立与远程 Adaptive Server 的连接时，如果 cis_rpc_handling 或 set transactional_rpc 为 on，则使用 Client-Library 函数而不是节点处理器。这种建立连接的方法可以禁止远程服务器区分这些连接和其它客户端的连接。因此，为允许或禁止与给定服务器的连接而在远程服务器上配置的所有远程服务器安全性均不生效。

已启用组件集成服务的其它 Adaptive Server 无法将*受托模式*用于远程服务器连接。如果 Adaptive Server 与组件集成服务一起使用，则强制使用所有可能的用户帐户来配置 Adaptive Server。

口令是以加密的形式内部存储的。

在 CIS 中使用加密列

缺省情况下，加密和解密是由远程 Adaptive Server 处理的。 CIS 在远程 Adaptive Server 上进行一次性加密列检查。如果远程 Adaptive Server 支持加密，则 CIS 会使用如下所示的与加密列相关的元数据更新本地 syscolumns 目录：

- create proxy_table 可自动使用远程表中的任何加密列信息更新 syscolumns。
- create existing table 可自动使用远程表中的任何加密列元数据更新 syscolumns。在 create existing table 的 *column_list* 中不允许使用 encrypt 关键字。如果 CIS 在远程表中找到任何加密的列，它会自动将其标记为已加密。
- 不允许在包含加密列的位置执行 create table。
- 不允许对代理表的加密列执行 alter table。

- select into existing 可选择源表中的纯文本并将其插入目标表中。然后，本地 Adaptive Server 会在将该纯文本插入任何加密的列中之前对其进行加密。

将从远程服务器的 syscolumns 目录更新以下列：

- encrtype — 磁盘上数据的类型。
- encrlen — 加密数据的长度。
- status2 — 指示列已被加密的状态位。

远程服务器登录

为了完全支持远程登录， Client-Library 提供了一些连接属性，使组件集成服务能够请求 *server connection*。此连接在接收服务器上被识别为服务器连接 （而不是普通客户连接），允许该远程服务器通过使用 sysremotelogins 验证连接，如同该连接是由节点处理器创建的一样。

不会自动启用服务器连接。相反， SSO 或 DBA 必须通过执行 sp_serveroption 来请求启用该功能：

```
exec sp_serveroption <server_name>,
    'server login', true | false
```

如果当前服务器的 @@servername 全局变量为 NULL，则无法更改 server login 属性。

如果 server login 选项为 True，组件集成服务将使用 Client-Library 连接属性来建立与指定服务器的连接。

将客户端应用程序指定的远程口令传递到远程服务器，而不做任何更改。服务器登录中的远程口令的使用及其关联的规则与节点处理器连接关联的口令使用和规则相同。

仅在以下情况下能够确定这些连接属性：

- 将服务器选项 server login 设置为 True。
- 使用服务器类 ASEnterprise 来配置远程服务器。
- 为启用组件集成服务的服务器定义了本地服务器名 （即查询 select @@servername 返回非 NULL 的结果）。

受托模式

如果为远程服务器设置了 “服务器登录”，则可将受托模式与组件集成服务连接一起使用。

连接到 Backup Server 和 XP Server

从 Adaptive Server 12.5.1 开始，组件集成服务可以将 RPC 发送到 Backup Server 或 XP Server。在执行此操作之前，必须启用服务器选项 negotiated logins：

```
exec sp_serveroption server_name, "negotiated logins",
true
```

这样，组件集成服务就可以对 Sybase 提供的两种服务器之一发出的登录询问做出响应。

映射外部登录名

调用组件集成服务的 Adaptive Server 用户需要登录名和口令才能访问远程服务器。缺省情况下，组件集成服务用于连接远程服务器的用户名和口令对与客户端用于连接 Adaptive Server 的用户名和口令相同。

组件集成服务支持以一对一的方式将 Adaptive Server 登录名和口令映射到远程服务器登录名和口令。例如，使用存储过程 sp_addexternlogin，可以将 Adaptive Server 用户 steve、口令 sybase 分别映射到 Oracle 登录名 login1、口令 password1：

```
sp_addexternlogin Oracle, steve, login1, password1
```

在 Adaptive Server 12.5 和更高版本中，可以提供多对一的映射，这样可以向需要 Oracle 连接的所有 Adaptive Server 用户指派相同的登录名和口令：

```
sp_addexternlogin Oracle, NULL, login2, password2
```

一对一映射具有优先权，因此如果用户 steve 拥有 Oracle 的外部登录名，则使用一对一映射而不是多对一映射。

此外，您可以将外部登录名指派给 Adaptive Server 角色。对于给定远程服务器，可以使用此功能，为拥有特定角色的任何人指派相应的登录名/口令：

```
sp_addexternlogin Oracle, null, login3, password3, rolename
```

角色名指定角色的名称，而不是用户的名称。当具有此活动角色的用户需要到 Oracle 的连接时，将使用该角色的相应登录名/口令来建立连接。在为具有多个活动角色的用户建立与远程服务器的连接时，将搜索每个角色的外部登录名映射，并使用找到的第一个映射来建立登录。这与 sp_activeroles 所显示的顺序相同。

sp_addexternlogin 的常规语法为：

```
sp_addexternlogin
    <servername>,
    <loginname>,
    <external_loginname>,
    <external_password>
    [, <rolename>]
```

<*rolename*> 是可选的；如果指定该项，则忽略 *loginname*。

这些功能的优先级如下：

- 如果定义了一对一的映射，则使用此映射。
- 如果未定义一对一映射，而角色处于活动状态并且可以找到其映射，则使用该角色映射来建立远程连接。
- 如果未定义一对一映射，而角色处于活动状态并且可以找到其映射，则使用该角色映射来建立远程连接。
- 如果上述条件均不为真，则使用 Adaptive Server 登录名和口令来建立连接。

如果完成了角色映射，并且更改了用户的角色 （通过 set role），则断开任何使用角色映射的远程服务器连接。

已对 sp_helpexternlogin 进行了更新，以便允许查看使用 sp_addexternlogin 添加的不同类型的外部登录。 sp_helpexternlogin 的语法为：

```
sp_helpexternlogin [<servername> [,<loginname> [,<rolename>]]]
```

所有三个参数均为可选参数，并且均可为 NULL。

存储过程 sp_dropexternlogin 还接受 <*rolename*> 参数。如果指定了 <*role name*>，则忽略第二个参数 <*login name*>。

远程服务器连接故障切换

如果设置了 interfaces 文件 （或 LDAP 目录服务）来定义故障切换配置，则当与主服务器的连接失败时，组件集成服务会利用故障切换功能自动连接到故障切换服务器。

执行以下配置步骤后，可以设置远程服务器来实现故障切换功能：

1　启用新的服务器选项 cis hafailover：

```
exec sp_serveroption server_name, 'cis hafailover',
true
```

2　修改目录服务 （interfaces 文件或 LDAP 服务器中的服务器条目）以指定故障切换服务器

例如，可通过在 interfaces 文件中添加以下命令，将服务器 S2 配置为 S1 的故障切换服务器 （反之亦然），如下例所示：

```
S1
        master tcp ether host1 8000
        query tcp ether host1 8000
        hafailover S2
S2
        master ether host2 9000
        query ether host2 9000
        hafailover S1
```

有关 CS_HAFAILOVER 连接属性的详细讨论，请参见附录 C，在 《高可用性系统中使用 Sybase 故障切换》。如果 cis hafailover 服务器选项为 true，组件集成服务器将使用 ct_con_props() API 来设置此属性。

远程服务器功能

Adaptive Server 首次与 sds 或 direct_connect 类的远程服务器建立连接时，它发出名为 sp_capabilities 的 RPC 并等待返回结果集。此结果集描述远程服务器的功能，使组件集成服务能够调整它与该远程服务器的交互以利用可用的功能。组件集成服务根据其功能将尽可能多的语法转移到远程服务器。

查询处理

本节说明在组件集成服务内进行的查询处理。

处理步骤

启用组件集成服务时采用的查询处理步骤与 Adaptive Server 采用的步骤类似，但以下情况除外：

- 如果以直通模式建立客户端连接，则绕过 Adaptive Server 查询处理，并将 SQL 文本转移到远程服务器上执行。
- 在将 select、 insert、 delete 或 update 语句提交到服务器上执行时，如果引用了本地代理表，组件集成服务可能会采取其它步骤来提高查询的性能。

后面将对查询处理步骤进行概述。

查询语法分析

SQL 语法分析程序检查进来的 SQL 语句的语法，如果 Transact-SQL 语法分析程序无法识别被提交执行的 SQL，则出现错误。

查询规范化

查询规范化期间，将检验 SQL 语句中引用的每个对象。查询规范化检验语句中引用的对象是否存在，及数据类型与语句中的值是否兼容。

示例

```
select * from t1 where c1 = 10
```

查询规范化阶段检验系统目录中是否存在带有名为 c1 列的表 t1。它还检验列 c1 的数据类型是否与值 10 兼容。例如，如果列的数据类型为 datetime，则拒绝该语句。

查询预处理

查询预处理准备进行优化的查询。它可能会更改语句的表示形式，这样，组件集成服务生成的 SQL 语句与初始语句在语法上会有所不同。

预处理执行视图扩展，因此查询可对视图引用的表进行操作。预处理还采用如对表达式重新排序和转换子查询等步骤提高处理效率。例如，子查询转换可能将一些子查询转换为连接。

决策点

预处理后，将确定是由组件集成服务还是由标准 Adaptive Server 查询优化程序来执行优化。

在以下情况下，组件集成服务使用称为 “快速传递模式”的功能来处理优化：

- SQL 语句中表示的每个表均位于单独的远程服务器中。
- 远程服务器能够处理语句所表示的所有语法。

  组件集成服务使用其服务器类来确定远程服务器的查询处理功能。例如，组件集成服务假定任何配置为服务器类 sql_server 的服务器都能够处理所有的 Transact-SQL 语法。

  对于具有服务器类 direct_connect 的远程服务器，当首次建立与服务器的连接时，组件集成服务发出 RPC 询问远程服务器的功能。根据服务器对 RPC 的响应，组件集成服务确定将转移到远程服务器的 SQL 的语法。
- 必须满足 SQL 语句的下列条件：
  - 它是 select、 insert、 delete 或 update 语句。
  - 如果它为 insert、 update 或 delete 语句，则没有 identity 或 timestamp 列或参照约束。
  - 它不包含 text 或 image 列。
  - 它不包含 compute by 子句。
  - 它不包含 for browse 子句。
  - 它不是 select...into 语句。
  - 它不是与游标相关的语句 （例如，包含 where current of cursor 的 fetch、 declare、 open、 close、 deallocate、 update 或 delete 语句）。
- 远程连接不包括在远程服务器上打开的游标。

如果不满足以上条件，则无法使用快速传递模式，而由标准 Adaptive Server 查询优化程序来处理优化。

组件集成服务计划生成

如果可以使用快速传递模式，组件集成服务将生成简化的查询计划。当语句包含代理表时，由远程服务器对它们进行处理的速度要快于通过 Adaptive Server 计划生成阶段的处理速度。

Adaptive Server 优化和计划生成

Adaptive Server 优化和计划生成评估执行查询的最佳路径，并生成说明 Adaptive Server 如何执行查询的查询计划。

如果已为查询中的表运行 update statistics 命令，则优化程序具有足够多的数据来进行有关优化连接顺序的决策。如果尚未运行 update statistics，则应用 Adaptive Server 缺省值。

有关 Adaptive Server 优化的详细信息，请参见《性能和调优指南》中的第 7 章 “Adaptive Server 查询优化程序”。

组件集成服务计划生成

如果可以使用快速传递模式，组件集成服务将生成简化的查询计划，在其中整个语句被传递到远程服务器。

如果无法使用快速传递模式， Adaptive Server 优化程序将生成一个计划以执行整个语句。然后检查此计划，选择计划的某些部分来脱离远程服务器。初始计划被脱离的程度与基于表的位置和远程服务器的功能的可能性相同。对于具有完整功能的远程服务器，远程语句可能非常接近于原始语句。对于用本地 Adaptive Server 执行无法发送的部分计划的其它服务器，可能生成更小的语句。

例如，如果客户端输入以下语句：

```
select a,b from table1 where cos(a) > 0 and sin(b) > 0
```

如果拥有 table1 的远程服务器支持 cos() 但不支持 sin()，则发送到远程服务器的语句将为：

```
select a,b from table1 where cos(a) > 0
```

然后本地服务器将有一个计划，该计划将 sin(b) > 0 的检查应用到由远程服务器返回的结果集。

组件集成服务远程位置优化程序

Adaptive Server 生成包含多表查询最佳连接顺序的查询计划，而不考虑每个表的存储位置。如果在查询中表示远程表，则组件集成服务将执行附加的优化并考虑位置，还可能重新安排允许远程执行部分连接的连接顺序的计划。

统计信息

要做出明智的计划选择，需要查询中涉及的所有表（包括代理表）的统计信息。通过对特定表执行 update statistics 可获得这些信息。

如果尚未运行 update statistics，则应用 Adaptive Server 缺省值。有关 Adaptive Server 优化的详细信息，请参见《性能和调优指南》中的第 7 章 “Adaptive Server 查询优化程序”。

代理表的优化程序开销模型

Adaptive Server 优化程序合并了基于 “网络交换”单元对远程服务器进行网络访问所需的开销，该单元指定执行序列所需的时间：

- 打开游标
- 读取 50 行
- 关闭游标

单个交换的开销由用户来控制，并通过 sp_serveroption 基于每个服务器来指定，其缺省值为 1000 毫秒：

```
sp_serveroption <servername>, "server cost", "nnnn"
```

其中 *nnnn* 是一个数位字符串，表示在优化程序计算网络开销的过程中每个交换所使用的毫秒数。

注释　服务器开销限制为 32767。如果超过该限制，则发生算术溢出错误。

在使用 sp_addserver 将新的服务器添加到 sysservers 时，就会将缺省开销 1000 毫秒存储在该服务器的 sysattributes 中。可以使用 sp_serveroption 为给定服务器指定较大或较小的开销。 sp_helpserver 显示与服务器关联的当前网络开销。

查询计划执行

服务器检查所有能够影响表的命令，以确定对象是拥有本地存储位置还是远程存储位置。如果存储位置是远程的，在执行查询计划以便为远程对象应用请求操作时，就会调用相应的访问方法。如果对映射到远程存储位置的对象执行以下命令，这些命令将受到影响：

- alter table
- begin transaction
- commit

- create index
- create table
- create existing table
- deallocate table
- declare cursor
- delete
- drop table
- drop index
- execute
- fetch
- insert
- open
- prepare transaction
- readtext
- rollback
- select
- set
- setuser
- truncate table
- update
- update statistics
- writetext

RPC 处理和组件集成服务

在启用了组件集成服务时，可以选择节点处理器或组件集成服务来处理外发的远程过程调用 (RPC)。以下各节中对这些机制进行了说明。

节点处理器和外发 RPC

在 Adaptive Server 内，通过节点处理器来传输外发 RPC，节点处理器对通过到远程服务器的单个物理连接发出的多个请求进行多路复用。RPC 作为多步操作的一部分进行处理：

1　建立连接 — Adaptive Server 节点处理器建立与远程服务器的单个物理连接。每个 RPC 均需要在这个物理连接上建立一个逻辑连接。该逻辑连接通过将要使用的远程服务器进行传递。

   这些连接请求的连接检验过程与常规客户端连接的检验过程不同。首先，远程服务器必须确定是否在服务器的 sysservers 表中配置了来自起始连接请求的服务器。如果是这样，检查系统表 sysremotelogins，确定应如何处理连接请求。如果配置了受托模式，则不执行口令检查。（有关受托模式的详细信息，请参见第 35 页的 “受托模式”）。

2　传输 RPC — 通过逻辑连接传输 RPC 请求。

3　处理结果 — 将 RPC 的所有结果从逻辑连接传递到客户端。

4　断开连接 — 终止逻辑连接。

逻辑连接和断开步骤可能会降低节点处理器 RPC 的速度。

组件集成服务和外发 RPC

如果已启用组件集成服务，客户端可以使用两种方法之一来请求组件集成服务处理外发 RPC 请求：

- 通过发出以下命令配置组件集成服务，将外发 RPC 作为所有客户端的缺省设置来处理：

  ```
  sp_configure "cis rpc handling", 1
  ```

  如果使用此方法设置 cis rpc handling 配置参数，则所有新客户端连接将继承此行为，并且由组件集成服务来处理外发 RPC 请求。这是由所有未来连接继承的一个服务器属性。必要时，客户端可以通过发出以下命令恢复缺省的 Adaptive Server 行为：

  ```
  set cis_rpc_handling off
  ```

- 通过发出以下命令，将组件集成服务配置为仅处理当前连接的外发 RPC：

  ```
  set cis_rpc_handling on
  ```

  此命令仅为当前线程启用 cis rpc handling，而不会影响其它线程的行为。

在启用了 cis rpc handling 时，不通过 Adaptive Server 节点处理器来传递外发 RPC 请求。而是通过组件集成服务来传递，组件集成服务使用持久 Client-Library 连接来处理 RPC 请求。使用此机制，组件集成服务按如下方式来处理外发 RPC：

1 确定客户端是否已与 RPC 要传递到的服务器建立了 Client-Library 连接。如果没有，将建立一个连接。

2 使用 Client-Library 函数将 RPC 发送到远程服务器。

3 将结果从远程服务器传递回使用 Client-Library 函数发出 RPC 的客户端程序。

RPC 可以包含在用户定义的事务中。实际上，组件集成服务代表其客户端执行的所有工作都是在单个连接环境中完成的。这样，就可以将 RPC 包含在工作的事务单元中，而 RPC 执行的工作可以与在该事务中执行的其它工作一起提交或回退。

使用组件集成服务处理外发 RPC 请求的好处如下：

- Client-Library 连接是持久的，因此后续 RPC 请求可以使用与远程服务器相同的连接。这可显著提高 RPC 性能，因为除第一个 RPC 外，所有 RPC 均绕过连接和断开连接逻辑。
- RPC 执行的工作可以包含在事务中，并与该事务执行的其余工作一起提交或回退。当前，只有在接收 RPC 的服务器是另一个支持事务性 RPC 的 Adaptive Server 或 DirectConnect 时，才支持此事务性 RPC 行为。
- 对于远程服务器来说，连接请求以普通客户端连接的形式出现。除非启用了 server logins，否则远程服务器无法区分该连接和常规应用程序的连接。这将影响 Adaptive Server 的远程服务器管理功能，因为未对 sysremotelogins 执行任何检验，并且所有连接在发出连接请求之前必须建立了有效的 Adaptive Server 登录帐户 （在此情况下，不能使用受托模式）。

RPC 的文本参数

Adaptive Server 可以在单个远程过程调用中发送大量数据。可以使用以下方法来实现这种功能：将某些参数作为文本指针来处理，然后取消引用这些文本指针以获得与其相关的文本值。然后，将 text 数据打包为 16K 块 （对于 Adaptive Server）和 32K 块 （对于所有其它服务器），并作为 RPC 参数传递给 Client-Library。

文本指针可被识别为 binary(16) 或 varbinary(16) 类型的参数。在执行 RPC 时，将获得每个文本指针参数所引用的文本值，并将其扩展为 16K 的块（对于 Adaptive Server）和 32K 的块 （对于所有其它服务器）中，其中每个块均作为 CS_LONGCHAR_TYPE 类型的参数传递到 Client-Library。

可使用以下 set 命令触发该行为：

```
set textptr_parameters ON
```

当请求 RPC 时 （cis_rpc_handling 必须为 on），在组件集成服务层中取消引用文本指针，并使用获得的文本值来构造一个或多个 Client-Library 参数。

要完成这一过程，文本指针前面必须有一个路径名参数，该参数用于识别从中派生文本指针的表。例如：

```
declare @pathname varchar(90)
declare @textptr1 binary(16)
declare @textptr2 binary(16)
select @pathname = "mydatabase.dbo.t1",
            @textptr1 = textptr(c1),
            @textptr2 = textptr(c2)
      from mydatabase.dbo.t1
      where ... (whatever)
set textptr_parameters ON
exec SYBASE...myrpc @pathname, @textptr1, @textptr2
set textptr_parameters OFF
```

- 当名为 ‘myrpc’ 的 RPC 被发送到服务器 SYBASE 时，实际上并不发送 *@pathname* 参数，而是使用该参数来帮助定位 textptr 的 *@textptr1* 和 *@textptr2* 所引用的文本值。

varchar 参数 *@pathname* 必须紧靠在 binary(16) 参数前面，否则，*@textptr1* 将被视为普通参数并被作为常规 binary(16) 值传输到服务器 SYBASE。

文本将被拆分为 16K 或 32K 块，每个块都是 CS_LONGCHAR_TYPE 类型的单独参数。

忽略 *@@textsize* 的当前值。

此方案旨在用于映射到远程过程的代理表。例如：

```
create existing table myrpctable
(
      id int,      -- result column
      crdate datetime, -- result column
      name varchar(30), -- result column
      _pathname varchar(90), -- parameter column
      _textptr1 binary(16), -- parameter column
      _textptr2 binary(16), -- parameter column
) external procedure at 'SYBASE...myrpc'
go
declare @textptr1 binary(16)
declare @textptr2 binary(16)
select @textptr1 = textptr(c1), @textptr2 = textptr(c2)
from mydatabase.dbo.t1 where <whatever>
set textptr_parameters ON
select id, crdate, name
from myrpctable
where_pathname = "mydatabase.dbo.t1" and
      _textptr1 = @textptr1 and
      _textptr2 = @textptr2
```

在处理对代理表 myrpctable 的查询时，组件集成服务向服务器 'SYBASE' 发送名为 'myrpc' 的 RPC。将从包含在查询的 where 子句中的搜索参数中派生这些参数。由于 textptr_parameter 选项已被设置为 ON，因此 textptrs 将扩展为 CS_LONGCHAR_TYPE，这与上述 RPC 示例所示的情况相同。

XJS/390 支持文本参数

由于能够将大文本块作为 RPC 参数转移，因此现在组件集成服务可以使用 XJS/390 与 IBM 大型机进行交互。 XJS/390 脚本 （JavaScript 式语法）可以存储在 Adaptive Server 表 （或可通过代理表访问的文件）中，并使用 RPC 转移到大型机。用 XJS/390 功能分析和执行该脚本的语法，并根据脚本的过程逻辑生成结果集。

启用了以下几项功能：

- Adaptive Server 中的数据库事件可以生成 MQ Series 消息。由于 XJS/390 Mscript 支持生成消息，因此可以将 RPC 发送到大型机以请求为响应数据库中的触发事件而生成的此类消息。
- 组件集成服务用户现在无需安装第三方中间件 （如 InfoHub 等），即可访问 VSAM、 IMS 和 MQSeries 数据。

要将脚本作为 RPC 参数进行处理，需要安装 XJS/390 2.0 版本或更高版本。有关详细信息，请参见 XJS/390 规范。

分布式事务管理

Adaptive Server 中的分布式事务管理跟踪本地 Adaptive Server/组件集成服务中事务的状态，还跟踪参与事务的所有远程服务器中事务的状态。在用户应用程序提交事务时，使用 Adaptive Server 事务协调程序 (ASTC) 将提交内容传播到所有参与的远程服务器。由 ASTC 协同组件集成服务来完成多节点事务的管理。组件集成服务为每个事务注册新的参与服务器，然后将事务协调的控制权转交给 ASTC， ASTC 回调组件集成服务来执行用于事务管理的各项命令。

服务器类和 ASTC

在内部， ASTC 将服务器视为以下两种类型之一：

- DTM-enabled
- Pre-DTM

这些类型映射到所使用的三个回调集合，并映射到服务器类 （如表 2-5 所示）：

表 2-5：事务功能

ASTC 服务器类型	组件集成服务服务器类
DTM-enabled	ASEnterprise （12.x 或更高版本） DC/Oracle 12.5 或更高版本
Pre-DTM	ASEnterprise （12.0 之前的版本） ASAnywhere ASIQ 其它 Direct Connect sds

注释　启动分布式事务以前，必须命名本地服务器。 *@@servername* 不能为 NULL。

DTM-enabled 服务器

“DTM-enabled” 远程服务器支持 ASTC 所启用的完全两阶段提交服务。支持此服务的服务器必须允许使用不同的连接 （或会话）来提交或回退其它会话开始的事务。如果需要使用提交协调器 (ASTC) 连接到远程节点并提交或回退有问题的事务，则必须使用此功能。与 DirectConnect for Oracle 12.5 或更高版本相同，Adaptive Server 12.0 和更高版本提供此支持。

Pre-DTM 服务器

归类为 “pre-DTM” 的远程服务器支持事务管理语句 （如 begin tran、commit tran、 rollback tran），但不支持某个会话提交或回退另一个会话启动的事务。

组件集成服务尽最大可能为 pre-DTM 服务器可靠地管理用户事务。但是，服务器中包含的不同访问方法对此功能提供的支持是不同的。如果 Specialty Data Store 支持事务管理，则服务器类 ASEnterprise （12.0 之前的版本）、 ASAnywhere、 ASIQ、 direct_connect 和 sds 采用下述的常规逻辑。这种涉及远程服务器的事务管理方法使用两阶段提交协议。Adaptive Server 执行一种策略，确保在大多数情形下的事务完整性。但是，分布式工作单元仍可能处于不确定的状态。即，虽然使用两阶段提交协议，但未包括恢复进程。管理用户事务的常规逻辑如下：

组件集成服务使用 begin transaction 通知远程服务器开始工作。当准备好提交事务时，组件集成服务向属于该事务的每个远程服务器发送 prepare transaction 通知。 prepare transaction 强制回应远程服务器，以确定连接是否仍然可用。如果 prepare transaction 请求失败，则指示所有远程服务器回退当前事务。如果所有 prepare transaction 请求均成功，服务器向该事务涉及的每个远程服务器发送 commit transaction 请求。以 begin transaction 开始的任何命令均可以开始一个事务。其它命令被发送到远程服务器，并作为单个远程工作单元执行。

strict DTM enforcement

为确保实现两阶段提交功能， ASTC 使用 strict dtm enforcement 的概念。启用时，如果试图在事务中包含 pre-DTM 服务器， strict dtm enforcement 将导致事务中止。

enable xact coordination

ASTC 使用配置选项 enable xact coordination。缺省情况下启用此选项，此选项允许 ASTC 管理涉及远程服务器的所有事务。必须启用组件集成服务才能启用 xact coordination。启用 xact coordination 时，不能禁用组件集成服务。启用 xact coordination 时，将隐式启用 transactional_rpcs。

启用组件集成服务

ASTC 依靠组件集成服务来处理与远程服务器的所有通信。由于在缺省情况下启用 ASTC (enable xact coordination)，因此在缺省情况下也会启用组件集成服务。

事务性 RPC

服务器允许将 RPC 包含在由当前事务启动的工作单元中。

在使用事务性 RPC 之前，发出 set transactional_rpc on 命令。

假定远程服务器能够支持在事务中包含 RPC，以下语法显示如何使用此功能：

```
begin transaction
 insert into t1 values (1)
 update t2 set c1 = 10
 execute @status = SYBASE.pubs2.dbo.myproc
 if @status = 1
      commit transaction
 else
      rollback transaction
```

在此示例中，服务器 SYBASE 中由过程 *myproc* 所执行的工作包含在以 begin transaction 命令开始的工作单元中。此示例要求远程过程 *myproc* 对成功返回状态 “1”。应用程序控制是否将工作作为一个完整单元提交或回退。

要接收 RPC 的服务器必须允许将 RPC 作为数据操作语言 (DML) 命令（select、 insert、 delete、 update）包含在相同的事务性环境中。对于 Adaptive Server 也是如此，并且 Sybase 发行的大多数 DirectConnect 产品也是如此。但是，有些数据库管理系统可能不支持此功能。

对事务管理的限制

如果嵌套的 begin transaction 和 commit transaction 语句包含在涉及远程服务器的事务中，则只处理最外层的一组语句。不会将包含 begin transaction 和 commit transaction 语句的最里层一组语句传输到远程服务器。

Adaptive Server 到 Adaptive Server 的 update statistics

对远程服务器代理表执行 update statistics 命令时，如果相关的表、索引和列统计信息可用，则将表目录导入本地 systabstats 和 sysstatistics。

缺省情况下，对代理表执行 update statistics 命令会总是尝试导入需要的统计信息数据。但如果远程表上的统计信息数据不可用或不完整，则组件集成服务 (CIS) 还原为以前的统计信息数据收集机制。

您也可以通过打开跟踪标志 11229 强制 CIS 还原为以前的统计信息数据收集机制。这样，您便可以从数据库获取所有数据，然后计算统计信息。

注释 这是在尚未对远程表运行 update statistics 命令且没有可用的统计信息时的行为。

限制

关键限制：

- 代理表必须映射到另一个 Adaptive Server 11.9 版或更高版本。
- 排除映射到 RPC、外部文件和系统目录的代理表。
- 如果远程服务器不是 Adaptive Server Enterprise 11.9 版或更高版本，或属于其它服务器类，则 CIS 继续使用以前的机制获取统计信息数据。

对非 Adaptive Server 后端运行 update statistics

update statistics 命令通过提供关于索引中键值的分布信息，帮助服务器在处理查询时做出关于要使用哪些索引的最佳决策。在已包含数据的表上创建或重新创建索引时，并不自动运行 update statistics。当索引列中已添加、更改或删除大量数据时，可以应用它。查询优化中至关重要的元素是分布步长的精确程度。如果索引中的键值发生重要更改，则在该索引上重新运行 update statistics。

只有表所有者或系统管理员可以发出 update statistics 命令。

语法为：

update statistics *table_name* [*index_name*]

因为运行 update statistics 相当耗费资源，因此，请尝试在未频繁使用指定的表时运行 update statistics。 update statistics 在读取数据时获取对远程表和索引的锁定。如果您使用跟踪标志 11209，则不锁定表。

可以将 update statistics 设置为在最适合您的站点的时间自动运行，并避免在妨碍您系统的时间运行它。有关详细信息，请参见 《性能和调优指南：监控和分析》中的第 4 章 “利用统计信息来提高性能”。

服务器对 update statistics 命令中指定的每个索引执行表扫描。

由于 Transact-SQL 不要求索引名在数据库中唯一，因此您必须给出与索引相关联的表的名称。

运行 update statistics 后，请运行 sp_recompile ，以便使用这些索引的触发器和过程使用新的分布：

```
sp_recompile authors
```

数据库中的 Java

通过组件集成服务进行的远程数据访问支持数据库中的 Java。

但有以下限制：

- 只有远程 Adaptive Server 12.x 和更高版本支持 Java。
- 仅语言事件支持 Java （动态 SQL 不能用于远程表）。

在使用 Java 进行远程数据访问之前，请阅读第 53 页的 “Java 类定义”。然后，当在本地服务器上安装 Java 类文件后，在远程服务器上安装所需的 Java 类文件。

@@textsize

使用 image 数据类型格式，将数据作为序列化的 Java 对象返回，然后在本地服务器上进行非序列化。 @@textsize 必须设置得足够大以容纳序列化的对象。如果 @@textsize 设置太小，则将对象截断，并且非序列化也将失败。

@@stringsize

@@stringsize 表示要从 toString() 方法返回的 character 数据的量。这与 @@textsize 的行为相类似，只是它只应用通过 Java Object.toString() 方法返回的 char 数据。缺省值为 50。最大值为 16384。值为零表示 “使用缺省值”。可使用 set 命令修改此值：

```
set stringsize n
```

其中 *n* 是 0 到 16384 之间的整数值。该值将立即出现在全局变量 @@stringsize 中。

Java 类列的约束

在远程表的 Java 列上定义的约束必须在远程服务器上进行检查。如果试图在本地服务器上进行约束检查，检查将失败。因此，当对进行完 Java 数据类型约束检查的数据执行 insert、update 或 delete 时，必须启用跟踪标志 11220。请参见第 66 页的 “跟踪标志”。

错误消息

对于使用 Java 进行的远程数据访问，共有两种特定的错误消息：

- 错误 11275 — 引用扩展数据类型的语句包含禁止将其发送到远程服务器的语法。可重新编写语句或删除扩展数据类型引用。
- 错误 11276 — 无法非序列化列 '<colname>' 中的对象，可能的原因是对象被截断。检查 @@textsize 的值是否足够大以容纳序列化对象。

Java 抽象数据类型 (ADT)

SQL 中的 Java 类 (JCS) 是在 Adaptive Server 内存储和使用 Java 对象的方法。在此实现中，需要使用组件集成服务交互来支持远程服务器上的 Java 对象和 Java 函数。

在远程 Adaptive Server 12.0 版本或更高版本上，组件集成服务支持 JCS。

对象以序列化格式在本地和远程服务器之间传递，这种格式是用于重新实例化对象的二进制表示形式。组件集成服务将序列化对象作为 image blob 处理，并使用 text 和 image 处理函数在服务器之间传递对象。在继续处理之前，将在目标服务器上重新实例化该对象。

当处理包含对远程服务器上 Java 对象和函数的引用的查询时，组件集成服务尝试将尽可能多的语法转移到远程服务器。无法传递到远程服务器的任何查询部分均在本地服务器上进行处理，这要求对所有必需的远程对象进行序列化和非序列化。由于与序列化和非序列化 Java 对象相关联的开销很大，这些查询的性能明显低于相应的本地访问。

为便于服务器间交换 Java 对象，组件集成服务发出以下命令：

```
set raw_object_serialization ON
```

（对于每个启用 Java 的 ASEnterprise 服务器。）这使组件集成服务能够方便地非序列化从远程节点获取的对象。

Java 类定义

本地和远程服务器上的 Java 类定义必须兼容以便于在服务器间传递对象。由于这一原因，组件集成服务假定存在兼容性，并且在非序列化工作期间中将检测到对象定义中的任何错误。如果远程服务器上对象的序列化格式可成功地用于实例化本地服务器上的对象 （反之亦然），则认为这些对象是兼容的。同样，还必须在远程对象上定义与远程映射对象一起使用的本地服务器中引用的任何 Java 方法。

数据库管理员应该保证本地和远程服务器上的类定义相兼容。不兼容的对象和无效的方法引用将导致非序列化错误或 Java 例外，它们将取消请求的查询。

要提高总体性能，应增加 cis packet size 配置变量，以更加便于在服务器间传递序列化对象。可在服务器间传递具有 image 数据类型的序列化对象，其大小可从几个字节到 2GB。

数据类型

本节讨论组件集成服务如何处理不同数据类型的问题。

Unicode 支持

Adaptive Server 包含对 Unicode 字符集的正式支持。所提供的数据类型为 unichar、univarchar 和 unitext。它们构成 Unicode 中表示的 2 个字节字符。 Adaptive Server 提供 Unicode 数据和所有其它数据类型之间的转换函数，以便与当前的 char 和 varchar 数据类型处理保持一致。通过支持这些数据类型，组件集成服务可以提供以 Unicode 表示的所有企业字符数据的视图。在对代理表中定义的列使用 unichar 或 univarchar 类型的列时，大型机和所有其它外部或遗留系统的字符数据均转换为 Unicode。

以下组件集成服务功能受这些新数据类型的影响：

create table

create table 可能包含使用新 Unicode 数据类型进行说明的列。如果要创建的表为代理表，则组件集成服务将整个命令转移到要在其中创建新表的远程服务器，整个命令中包括 Unicode 数据类型名称 （unichar、univarchar 和 unitext）。如果远程服务器无法处理这些数据类型，则出现错误。

create existing table

在将 Adaptive Server 列类型和长度与从远程服务器获得的元数据进行比较时，以下情况下允许代理表中使用 Unicode 数据类型：

- 列的远程服务器数据类型是具有相同长度 （以字符而不是字节表示）的 unichar、 unitext 或 univarchar。
- 给定列的远程服务器数据类型为 char 或 varchar。在这种情况下，组件集成服务将从远程服务器读取的数据转换为 Unicode，并将作为 DML 命令 （select、 insert、 delete、 update）的一部分传输的数据从 Unicode 转换为缺省 Adaptive Server 字符集 (UTF8)。
- Unicode 列的远程服务器数据类型为 binary 或 varbinary。远程服务器列的长度必须是 Unicode 列长度的两倍。在与远程服务器之间传输数据时，组件集成服务将根据需要对数据进行转换。

当将代理表映射到远程表时，不允许使用 Unicode 数据类型的其它映射数据类型。其它类型导致类型不匹配错误。通过创建代理表，将 Unicode 列映射到现有 char 或 varchar 列，即可将遗留系统中的数据转换为 Unicode。

注释　只能使用 create existing table 命令将 Unicode 映射到 unitext 列。

create proxy_table

通过使用 create proxy_table，Adaptive Server 用户不必指定与代理表相关联的列列表。相反，将从由实际表所在的远程服务器导入的列元数据派生列的列表。仅当远程列的数据类型为 unichar、unitext 或 univarchar 时，远程服务器的 Unicode 列才映射到代理表中的 Unicode 列。

alter table

alter table 允许修改列类型。在 Adaptive Server 12.5 版和更高版本中，可以将列的类型修改为 Unicode 数据类型，以及将 Unicode 数据类型修改为其它类型。如果在代理表上执行该命令，则重新构建该命令，并将其转移到拥有实际表的远程服务器。如果远程服务器 （或 DirectConnect）无法处理该命令，将出现错误并中止 Adaptive Server 命令。

如果跟踪标志 11221 为 ON， alter table 将不转移到远程服务器；仅在代理表上本地完成列的添加、删除或修改。

通过使用 alter table 命令，可以将 unitext 更改为 char、 varchar、 nchar、 nvarchar、 unichar、 univarchar、 binary 和 varbinary。所有这些数据类型均可更改为 unitext。

select、 *insert*、 *update* 和 *delete* 语句

当涉及代理表时，Unicode 数据类型以两种方法影响 select 语句的处理。第一种方法涉及构造传递到远程服务器的 SQL 语句和参数；第二种方法在组件集成服务读取非 Unicode 数据时将数据转换为 Unicode。

在与远程服务器进行交互时，将使用 TDS 语言请求或 TDS 游标请求来处理涉及代理表的 DML 命令。如果 select 语句的 where 子句中包含涉及 Unicode 列和常量的谓词，则必须按以下两种方式之一对 Unicode 常量进行处理 （具体取决于是使用语言还是游标命令来处理语句）：

1 TDS 语言 — 生成可包含在语言文本缓冲区中的明文值。它涉及将常量 Unicode 值转换为明文值，可将明文值作为语言请求的一部分进行传输。

2 TDS 游标 — 生成 CT-Library 游标请求的 Unicode 参数。参数值可以是 Unicode 数据，但要求组件集成服务使用 CS_UNICHAR_TYPE 参数类型。

组件集成服务使用 TDS 语言请求或 TDS 动态请求来处理涉及代理表的 insert 命令。

如果可以通过快速传递模式来处理 insert 命令，则使用 TDS 语言请求。如果无法通过快速传递模式来处理该命令，则使用 TDS 动态请求来处理 insert。

在语言请求的情况下，问题与 select 相同 — 必须将 Unicode 值转换为明文格式，以便与 SQL 语句的其它部分一起传输。在动态请求情况下，将 Unicode 数据 （同所有其它数据值一起）作为动态命令的参数进行传输。接收服务器应当可处理 CS_UNICHAR_TYPE 类型的参数。

update 和 delete 命令的问题与 select 和 insert 相同。必须将 Unicode 值转换为明文字符，以便与 SQL 语句的其它部分一起传输；或者必须转换为 CS_UNICHAR_TYPE 类型的参数。

数据类型转换

只要服务器从远程数据源接收到数据，并且远程数据源为 Adaptive Server 或基于 Open Server 的应用程序，就会发生数据类型转换。

根据每列的远程数据类型，将数据从远程服务器上的本机数据类型转换为本地服务器支持的一种格式。

当处理 create table、 alter table 和 create existing table 命令时，将进行数据类型转换。数据类型转换取决于远程服务器的服务器类。有关说明处理命令时各个服务器类所发生的数据库转换的表，请参见第 3 章 “SQL 参考”中的 create table、 alter table 和 create existing table 命令。

text 和 *image* 数据类型

text 数据类型用于存储可打印的字符数据，其列大小取决于 Adaptive Server 的逻辑页大小。 image 数据类型用于存储十六进制编码的二进制数据的字节数，其大小同样取决于 Adaptive Server 的逻辑页大小。 text、 image 和 unitext 数据的最大长度是由列映射到的远程服务器的服务器类定义的。

注释　仅 Adaptive Server 15.0 版本和更高版本支持组件集成服务使用 unitext。

对 *text*、 *image* 和 *unitext* 列的限制

不能以下列方式使用 text、 image 和 unitext 列：

- 用作存储过程的参数 （除非将 set textptr_parameters 设置为 on）
- 作为局部变量
- 在 order by、 compute 或 group by 子句中
- 在索引中
- 在子查询中
- 在 where 子句中 （除非带有关键字 like）
- 用于连接

@@textsize 的限制

select 语句根据在全局变量 *@@textsize 中指定的限制返回 text、image 和 unitext 数据。*可使用 set textsize 命令更改此限制。 *@@textsize* 的初始值为 32K； *@@textsize* 的最大值为 2147MB。

填充的奇数字节

小于 255 个字节并具有奇数字节的 image 值用前导的零填充 （插入的 “0xaaabb” 将变为 “0x0aaabb”）。如果值是奇数字节，则通过执行 insert 来插入超过 255 个字节的 image 值是错误的。

转换 *text* 和 *image* 数据类型

可以使用 convert 函数显式地将 text 值转换为 char 或 varchar，而将 image 值转换为 binary 或 varbinary，但要受 character 和 binary 数据类型的最大长度限制，该限制取决于 Adaptive Server 的逻辑页大小。如果未指定长度，转换后的值将具有缺省的长度 （30 个字节）。不支持隐式转换。

text 和 *unitext* 数据的模式匹配

使用 patindex 函数搜索 text、 unitext、 varchar 或 char 列中指定模式第一次出现的起始位置。通配符 % 必须位于模式之前或之后 （搜索第一个或最后一个字符的情况除外）。

可以使用 like 关键字搜索特定模式。以下示例从 texttest 表的 blurb 列中选择每个包含模式 “Straight Talk%” 的 text 数据值：

```
select blurb from texttest
 where blurb like "Straight Talk%"
```

可使用关键字 like 来搜索特定模式的 unitext 列。但是，在与 unitext 列一起使用时， like 子句不进行优化。与 unitext 匹配的 like 模式依赖于缺省的 Unicode 排序顺序，该顺序也用于与 unichar 和 univarchar 数据类型匹配的 like 模式。

输入 *text* 和 *image* 值

DB-Library™ 函数 dbwritetext 和 dbmoretext 以及 Client-Library 函数 ct_send_data 是输入 text、 unitext 和 image 值的最有效方法。

readtext using bytes

如果在 text 列上使用 readtext using bytes 命令，并且偏移和大小组合导致传输了部分字符，则出现错误。

使用 *bulk copy* 来处理 *text*、 *image* 和 *unitext*

使用 bulk copy 将 text、 unitext 和 image 值复制到远程服务器时，该服务器必须在数据页中存储有这些值才能将它们发送到远程服务器。一旦将值发送到远程服务器，即会释放该数据页。数据页是逐行分配并释放的。这非常重要，因为：

- 分配和释放数据页的开销影响性能。
- 数据页是在表所在的数据库中分配的，因此该数据库必须足够大，才能容纳任何给定行存在的最大 text、unitext 和 image 值的足够多的数据页。

错误记录

可以使用跟踪标志 11207 来记录对 text、 unitext 和 image 数据的处理（仅对远程服务器）。

服务器类为 *ASEnterprise* 的 *text*、 *unitext* 和 *image* 数据

- text、 unitext 或 image 列中的指针是在初始化列时分配的。必须先将列初始化，才能向该列输入 text、 unitext 或 image 数据。这导致在远程服务器或 Adaptive Server 上分配 2K 的页。要初始化 text、 unitext 或 image 列，请使用带有 NULL 的 update 命令或非空的 insert 命令。
- 在使用 writetext 输入 text 或 unitext 数据或者使用 readtext 命令读取 text 或 unitext 之前，必须初始化相应的列。可使用 update 或 insert 非空数据初始化 text 列，然后使用 writetext 和 readtext。
- 使用 update 将现有的 text、unitext 和 image 数据替换为 NULL 可回收除第一页以外的所有已分配数据页。
- 必须使用 writetext、 select into、 DB-Library 函数或 Client-Library 函数来输入大于 16KB 的 text、 unitext 或 image 值。
- readtext 是访问 text、 unitext 和 image 数据的最有效方法。
- 可以使用 insert select 和 select into 将 text、 unitext 和 image 数据插入代理表，但必须具有唯一的索引。

服务器类为 *direct_connect* 的 *text*、 *image* 和 *unitext* 数据

- 特定的 DirectConnect 服务器对 text 和 image 数据提供不同程度的支持。有关 text、 unitext 和 image 支持的信息，请参见 DirectConnect 文档。
- 服务器将在全局变量 *@@textsize* 中定义的长度用于列长度。发出 create table 之前，客户端应用程序应调用 set textsize 将 *@@textsize* 设置为所需的长度。
- 对于支持 text、 unitext 和 image 数据类型但不支持文本指针的 DirectConnect 服务器，存在以下限制：
  - 不支持 writetext 命令。
  - 不支持 readtext 命令。
  - 不支持使用文本指针的 Client-Library 函数。
  - 不支持使用文本指针的 DB-Library 函数。
- 对于支持 text、 unitext 和 image 数据类型但不支持文本指针的 DirectConnect 服务器，需要执行一些额外处理以允许使用下列函数：
  - patindex
  - char_length
  - datalength

  如果支持文本指针，服务器通过向 DirectConnect 服务器发出 RPC 来执行这些函数。
- 对于不支持文本指针的 DirectConnect 服务器，服务器在 sysattributes 系统表中存储数据。数据页是以逐页逐行的方式进行预分配的。列大小由 *@@textsize* 来确定。如果此值不够大，将返回错误。
- 特定的 DirectConnect 服务器可能支持或不支持与 text 数据类型相匹配的模式。如果 DirectConnect 服务器不支持此匹配模式，服务器将 text 值复制到内部数据页并在内部执行模式匹配。当 DirectConnect 服务器执行模式匹配时，可以获得最佳的性能。
- 必须使用 writetext、 select into 或 insert...select 来输入超过 450 个字节的 text、 unitext 或 image 值。
- 可以使用 select into 和 insert...select 来插入 text、 unitext 或 image 值，但表必须具有唯一的索引。

配置与调优

本节提供有关配置、调优、跟踪标志、备份和恢复以及安全问题的信息。

系统管理员或数据库所有者可以选择使用服务器来优化性能或允许由所需数量的客户端来使用。配置选择可能涉及能够检查给定 SQL 命令的读写总数。

一旦应用程序已启动且正在运行，系统管理员应对性能进行监控，并且可以选择对系统进行自定义和微调。服务器提供了用于这些目的的工具。本节说明：

- 使用 sp_configure 过程更改系统参数
- 使用 update statistics 来确保组件集成服务充分利用现有索引
- 使用 dbcc 命令监控服务器活动
- 设置跟踪标志
- 执行 ddlgen 及相关的备份和恢复问题
- 确定数据库大小要求

使用 *sp_configure*

sp_configure 中的配置参数控制资源分配和性能。系统管理员可以重新设置这些配置参数以调优性能和重新定义存储分配。如果没有系统管理员的干涉，服务器将为所有参数提供缺省值。

重新设置配置参数的过程为：

- 执行 sp_configure，这会更新系统表 master..sysconfigures 的值字段。
- 如果已重新设置了任何静态配置参数，请重新启动服务器。下面列出的参数为动态参数：
  - cis rpc handling
  - cis cursor rows
  - cis bulk insert batch size
  - cis bulk insert array size
  - cis packet size

sysconfigures 表

master..sysconfigures 系统表存储所有配置选项。它包括用于标识每个配置参数可能的最小和最大值的列，以及每个参数的配置值和运行值。

用户无法更新 sysconfigures 中的 status 列。状态 1 意味着是动态的，表示这些配置参数的新值立即生效。其余配置参数（状态为 0 的参数）只有在已发出 reconfigure 命令并且重新启动服务器之后才生效。

可通过执行 sp_configure 显示当前使用的配置参数（运行值），而无需为其指定任何参数。

更改配置参数

当使用无参数的sp_configure 时，它显示所有配置值。当使用选项名和值时，服务器重新设置系统表中该选项的配置值。

有关具有语法选项的 sp_configure 的完整讨论，请参见《系统管理指南》。

若要查看组件集成服务选项，请输入：

```
sp_configure "Component Integration Services"
```

要更改配置参数的当前值，请执行 sp_configure，如下所示：

```
sp_configure "parameter", value
```

组件集成服务配置参数

下面的配置参数是组件集成服务所特有的：

- enable cis
- enable file access
- enable full-text search
- max cis remote connections
- cis bulk insert batch size
- cis bulk insert array size
- cis cursor rows
- cis packet size
- cis rpc handling

enable cis

将此参数与 sp_configure 一起使用以启用组件集成服务，如下所示：

1　以系统管理员的身份登录到 Adaptive Server 并发出以下命令：

```
sp_configure "enable cis", 1
```

2　重新启动 Adaptive Server。

如果发出 sp_configure "enable cis", 0，在重新启动服务器后将禁用组件集成服务。

enable file access

此配置参数允许通过代理表来访问 “eXternal 文件系统”。

enable full-text search

此配置参数启用 Enhanced Full-Text Search 服务。需要一个 ASE_EFTS 的许可证。

max cis remote connections

非零值表示在服务器初始化期间预分配的连接数据结构数。缺省值为零。

cis bulk insert batch size

当目标表位于 Adaptive Server 或支持批量复制接口的 DirectConnect 服务器中时，此配置参数确定使用 select into 以单个批处理的形式从源表批量复制到目标表的行数。

如果参数值保留为 0 （缺省），则所有行均作为单个批处理进行复制。否则，按该参数指定的行数将行复制到目标表后，组件集成服务向目标服务器发出批量提交，从而导致提交该批处理。

如果接收到常规客户端生成的批量复制操作 （例如由 bcp 实用程序生成的操作），该客户端将控制批量处理的大小，并且组件集成服务会忽略此配置参数的值。

cis bulk insert array size

在 Adaptive Server 之间进行数据批量传送时，组件集成服务在内部为行分配缓冲区，并要求 Open Client 批量库将它们作为块传送。数组的大小由配置参数 cis bulk insert array size 控制。缺省值为 50 行，且其属性是动态的，允许对其进行更改而无需重新启动服务器。

cis cursor rows

此配置参数允许用户指定用于 cursor open 和 cursor fetch 操作的游标行计数。增加该值意味着，在一次操作中可以读取更多的行。这会加快速度，但要求使用更多的内存。缺省值为 50。

cis packet size

此配置参数允许您指定 Tabular Data Stream™ (TDS) 包的大小，在启动连接后，将在组件集成服务和远程服务器之间交换这些包。

大多数系统上的缺省包大小为 512 个字节，对于大多数应用程序来说，这足够了。但是，较大的包大小可能会显著提高查询性能，特别是当包含 text 和 image 或批量数据时。

如果包大小大于指定的缺省值，则必须配置目标服务器以允许可变长度的包大小。在这种情况下，需要注意以下 Adaptive Server 配置参数：

- additional netmem
- maximum network packet size

有关这些配置参数的完整解释，请参见 《系统管理指南》。

cis rpc handling

此全局配置参数确定在缺省情况下，组件集成服务是否处理外发的 RPC 请求。使用 sp_configure “cis rpc handling” 1 将其启用后，所有外发 RPC 均由组件集成服务进行处理。使用 sp_configure “cis rpc handling” 0 时，将使用 Adaptive Server 节点处理器进行处理。线程无法使用 set cis_rpc_handling on 替换此参数。如果禁用全局属性，线程可以根据需要启用或禁用此功能。

有关使用 Adaptive Server 节点处理器与使用组件集成服务处理外发 RPC 的详细信息，请参见第 43 页的 “RPC 处理和组件集成服务”。

状态的全局变量

已经为组件集成服务用户添加了以下全局变量：

- @@cis_rpc_handling
- @@transactional_rpc
- @@textptr_parameters
- @@stringsize
- @@bulkbatchsize — 包含通过 sp_configure 配置或通过 set bulk batch size 命令设置的当前 cis bulk insert batch size 的值。
- @@bulkarraysize — 包含通过 sp_configure 配置或通过 set bulk array size 命令设置的当前 cis bulk insert array size 的值。

这些全局变量显示了相应配置参数的当前状态。例如，要查看 cis_rpc_handling 的状态，请发出以下命令：

```
select @@cis_rpc_handling
```

它返回 0 （关）或 1 （开）。

第 3 章

SQL 参考

本章提供关于组件集成服务所支持的服务器类的参考资料。

每个服务器类都有一组唯一的特性，系统管理员和程序员需要了解这组特性，以便为远程数据访问配置服务器。这些属性是：

- 每个服务器类所支持的服务器类型
- 针对服务器类的数据类型转换
- 对服务器类应用的 Transact-SQL 语句限制

dbcc 命令

组件集成服务使用的所有 dbcc 命令都可以用一个单独的 dbcc 入口点获取。

dbcc cis 的语法为：

```
dbcc cis ("subcommand"[, vararg1, vararg2...])
```

如果未配置或装载组件集成服务，该命令将导致运行时错误。

dbcc cis 命令的使用不受限制。

dbcc 选项

以下 dbcc 选项是组件集成服务所特有的。

remcon

remcon 显示所有组件集成服务客户端建立的所有远程连接的列表。它不使用任何参数。

srvdes

如果不提任何供参数， srvdes 将返回一个格式化列表，列出内存中的所有 SRVDES 结构。如果提供参数，则此命令将内存中的 SRVDES 版本与在 sysservers 中找到的信息保持同步。此命令使用如下可选参数：

```
srvdes, [ srvid ]
```

showcaps

showcaps 显示一个列表，此列表按功能名、 ID 和值列出 servername 的所有功能，其形式如下：

```
showcaps, servername
```

示例：

dbcc cis(“showcaps”, “servername”)

跟踪标志

dbcctraceon 选项允许系统管理员在组件集成服务内打开跟踪标志。当跟踪标志出现在组件集成服务中时，这些跟踪标志启用某些事件的日志记录。每个跟踪标志都用一个数字唯一地进行标识。对于组件集成服务来说，有些标志是全局性的，而另外一些标志则基于 spid，并且只影响启用此跟踪标志的用户。 dbcc traceoff 可关闭跟踪标志。

语法为：

dbcc traceon (*traceflag* [, *traceflag*...])

表 3-1 显示了跟踪标志及其含义：

表 3-1：组件集成服务跟踪标志

跟踪标志	说明
11201	记录客户端连接事件、断开事件和关注事件。（全局）。仅发送到错误日志的事件。
11202	记录客户机语言、游标声明、动态准备和动态即时执行文本 （全局）。仅发送到错误日志的文本。
11203	记录客户端 RPC 事件 （全局）。仅发送到错误日志的事件。
11204	记录所有传送到客户端的消息 （全局）。仅发送到错误日志的消息。
11205	记录所有与远程服务器的交互信息 （全局）。仅发送到错误日志的交互。
11206	记录文件 / 目录处理步骤。（全局）
11207	记录 text 和 image 的处理信息。（全局）
11208	禁止将 create index 和 drop index 语句传输到远程服务器上。仍将更新 sysindexes。 (spid)
11209	指示 update statistics 从远程表中只获取行数信息，而不是获取全部的分布统计信息。 (spid)
11211	如果该表是使用 create table at location 语法创建的，则会禁止将 drop table 语法转移至远程服务器。
11212	禁止对表名中的下划线 （{1234}_{4321}）进行转义。 (spid)
11213	禁止生成列约束和表约束。 (spid)
11214	禁用启动时的组件集成服务恢复。（全局）
11216	禁用快速传递。 (spid)
11217	禁用快速传递。（全局）
11218	缺省情况下，使涉及组件集成服务表的游标可更新。
11220	禁止在本地服务器对远程表进行约束检查。这样可以避免重复检查。打开此跟踪标志，可保证查询不会由于约束而被快速传递模式拒绝。 (spid)
11221	打开此跟踪标志时，禁止对远程服务器执行 alter table 命令。这将允许用户在本地表中修改列的 *type*、 *length* 和 *nullability*，而无需在远程表中更改列。应小心使用跟踪标志 11221。它可能会使各表之间 “不同步”。 (spid)
11223	在 create existing table 或 create proxy_table 命令执行期间禁用代理表索引的创建功能。如果将此标志打开，则不会从代理表引用的远程节点导入索引元数据，也不会为代理表创建索引。必须小心使用此跟踪标志，当不再需要使用时，应将其关闭。（全局）
11228	禁用映射到 RPC 的代理表。

跟踪标志	说明
11229	指示组件集成服务使用 Adaptive Server 12.5.3 版本之前的方法收集统计信息数据。
11299	当连接到远程服务器失败时，允许记录连接信息。

函数

本节介绍组件集成服务服务器类与内置 Adaptive Server 函数的兼容性。

组件集成服务内对函数的支持

当 SQL 语句 （如 select、 insert、 delete 或 update）包含内置函数时，组件集成服务必须确定该函数是否可以转移到远程服务器，或者是否必须使用远程数据在本地服务器中求出其值。

如果可以用快速传递模式处理包含函数的语句，则只将函数发送到远程服务器。

在下面显示的表中，如果服务器类支持函数，则用字母 ‘Y’ 指示；如果不支持，则用字母 ‘N’ 指示；‘C’ 表示对函数的支持取决于基础 DBMS 的功能 （通常针对 DirectConnect 而言）。

集合函数

集合函数生成摘要值，这些值在查询结果中显示为新列。

表 3-2：服务器类对集合函数的支持

函数	ASE	ASA	ASIQ	dir_con
avg	Y	Y	Y	C
count	Y	Y	Y	C
max	Y	Y	Y	C
min	Y	Y	Y	C
sum	Y	Y	Y	C
count_big	Y	N	N	N

数据类型转换函数

数据类型转换函数将表达式从一种数据类型更改为另一种数据类型，并且为日期/时间信息指定新的显示格式。

表 3-3：服务器类对数据类型转换函数的支持

函数	ASE	ASA	ASIQ	dir_con
convert()	Y	Y	Y	C
inttohex()	Y	N	N	N
hextoint()	Y	N	N	N
biginttohex()	Y	N	N	N
hextobigint()	Y	N	N	N

日期函数

日期函数处理数据类型为 datetime 或 smalldatetime 的值。本地服务器始终扩展 getdate() 函数；但此内置函数的存在不会导致将某个查询从快速传递模式优化中消除。

表 3-4：服务器类对日期函数的支持

函数	ASE	ASA	ASIQ	dir_con
dateadd	Y	Y	Y	C
datediff	Y	Y	Y	C
datename	Y	Y	N	C
datepart	Y	Y	Y	C

数学函数

数学函数返回在数学数据运算中通常所需要的值。数学函数名不是关键字。

每种函数还可接受隐式地转换为指定类型的参数。例如，接受近似数值类型的函数还可接受整数类型。 Adaptive Server 自动将参数转换成所需的类型。

表 3-5：服务器类对数学函数的支持

函数	ASE	ASA	ASIQ	dir_con
abs	Y	Y	Y	C
acos	Y	Y	N	C
asin	Y	Y	N	C
atan	Y	Y	N	C
atn2	Y	Y	N	C
ceiling	Y	Y	Y	C
cos	Y	Y	N	C
cot	Y	Y	N	C
degrees	Y	Y	N	C
exp	Y	Y	N	C
floor	Y	Y	Y	C
log	Y	Y	N	C
log10	Y	Y	N	C
pi	Y	Y	N	C
power	Y	Y	N	C
radians	Y	Y	N	C
rand	Y	Y	Y	C
round	Y	Y	N	C
sign	Y	Y	N	C
sin	Y	Y	N	C
sqrt	Y	Y	Y	C
tan	Y	Y	N	C

安全性函数

安全性函数返回与安全性相关的信息。

表 3-6：服务器类对安全性函数的支持

函数	ASE	ASA	ASIQ	dir_con
ic_sec_service_on()	N	N	N	N
show_sec_services()	N	N	N	N

字符串函数

字符串函数对二进制数据、字符串和表达式进行计算。字符串函数有：

表 3-7：服务器类对字符串函数的支持

函数	ASE	ASA	ASIQ	dir_con
ascii	Y	Y	N	C
char	Y	Y	N	C
charindex	Y	Y	N	C
char_lengt	Y	Y	N	C
difference	Y	Y	Y	C
lower	Y	Y	Y	C
ltrim	Y	Y	Y	C
patindex	N	N	N	N
replicate	Y	Y	N	C
reverse	Y	N	N	Y
right	Y	Y	Y	C
rtrim	Y	Y	Y	C
soundex	Y	N	Y	C
space	Y	Y	N	C
str	Y	Y	N	C
stuff	Y	Y	N	C
substring	Y	Y	Y	C
upper	Y	Y	Y	C

系统函数

系统函数返回数据库中的特定信息。

表 3-8：服务器类对系统函数的支持

函数	ASE	ASA	ASIQ	dir_con
col_length	Y	Y	N	C
col_name	Y	Y	N	C
curunreservedpgs	N	N	N	N
data_pgs	N	N	N	N
datalength	Y	Y	N	C
db_id	N	N	N	N
db_name	N	N	N	N
getdate	Y	N	N	N
getutcdate	Y	N	N	N
host_id	N	N	N	N
host_name	N	N	N	N
index_col	N	N	N	N
isnull	Y	Y	N	N
lct_admin	N	N	N	N
mut_excl_roles	N	N	N	N
object_id	N	N	N	N
object_name	N	N	N	N
proc_role	N	N	N	N
ptn_data_pgs	N	N	N	N
reserved_pgs	N	N	N	N
role_contain	N	N	N	N
role_id	N	N	N	N
role_name	N	N	N	N
rowcnt	N	N	N	N
show_role	N	N	N	N
suser_id	N	Y	Y	N
suser_name	N	Y	Y	N
tsequal	Y	Y	N	N
used_pgs	N	N	N	N
user	Y	Y	Y	N
user_id	Y	Y	Y	N
user_name	Y	Y	Y	N
valid_name	N	N	N	N
valid_user	N	N	N	N

Text 和 image 函数

Text 和 image 函数对 text 和 image 数据进行计算。

表 3-9：服务器类对 Text 和 image 函数的支持

函数	ASE	ASA	ASIQ	dir_con
textptr()	Y	Y	N	C
textvalid()	Y	Y	N	C

Transact-SQL 命令

以下各页按字母顺序介绍直接或间接影响外部表并进而影响组件集成服务的 Transact-SQL 命令。对于每个命令，将提供它对组件集成服务的影响的说明以及组件集成服务处理该命令的方式。有关每个命令的完整说明，请参见 《参考手册》。

如果组件集成服务并未将某个命令的所有语法传递到远程服务器 （例如， select 语句的所有子句），则为每个服务器类说明所传递的语法。

每个命令都被分为几个部分来加以描述：

- 说明 — 包含对命令的简要说明。
- 语法 — 包含命令的完整 Transact-SQL 语法的说明。
- 用法 — 包含一般性的、独立于服务器类的组件集成服务处理说明。
- 服务器类 ASEnterprise — 包含专门针对服务器类 ASEnterprise 的处理说明。其中包括被转移到 ASEnterprise 类远程服务器的语法。
- 服务器类 ASAnywhere — 包含专门针对服务器类 ASAnywhere 的处理说明。其中包括被转移到 ASAnywhere 类远程服务器的语法。
- 服务器类 ASIQ — 包含专门针对服务器类 ASIQ 的处理说明。其中包括被转移到 ASIQ 类远程服务器的语法。
- 服务器类 direct_connect — 包含专门针对服务器类 direct_connect 的处理说明。其中包括被转移到 direct_connect 类远程服务器的语法。在这个版本中，所有适用于服务器类 direct_connect 的注释，同样适用于服务器类 sds。

alter table

服务器类 *ASEnterprise*

组件集成服务将以下语法转移到配置为类 ASEnterprise 的服务器：

```
alter table [database.[owner].]table_name
    {add column_name datatype [{identity | null}]
        {[, next_column]}...}
    | [drop column_name [, column_name]}
    | modify column_name [data_type] [NULL] |
        [not null]] [, column_name]}
```

- 当用户使用 alter table 命令添加列时，组件集成服务将每列的数据类型传递到远程服务器，而不进行类型名的转换。
- 仅限于 ASEnterprise 类服务器：如果 Adaptive Server 为 11.9.2 或更高版本，且初始查询中包含 lock 子句，则还会转移此子句。

服务器类 *ASAnywhere*

此类服务器对 alter table 的处理与 ASEnterprise 服务器相同。

服务器类 *ASIQ*

- 此类服务器对 alter table 的处理与 ASEnterprise 服务器相同。
- 服务器类 ASIQ 完全支持 text 和 image 数据类型。

服务器类 *direct_connect*

- 组件集成服务将以下语法转移到配置为 direct_connect 类的远程服务器：

```
alter table [database.[owner].]table_name
 add column_name datatype [{identity | null}]
        {[, next_column]}...
```

- 虽然组件集成服务从 direct_connect 类服务器中请求功能响应，但对 alter table 的支持不是可选的。无论功能响应是什么，组件集成服务都会将 alter table 转移到远程服务器。
- direct_connect 类服务器的行为取决于数据库。转移 Transact-SQL 语法时可能会出现错误，也可能不出现错误，具体情况取决于远程数据库处理此语法的能力。
- 服务器类 direct_connect 不支持 bigint、 unsigned tinyint、 unsigned smallint、 unsigned int、 unsigned bigint。
- 如果远程服务器的语法功能表明是 Sybase Transact-SQL，则将 Adaptive Server 数据类型发送到远程服务器。如果语法功能表明是 DB2 SQL，则发送 DB2 数据类型。

  Direct Connect 不支持 bigint、 unsigned tinyint、 unsigned smallint、 unsigned int、 unsigned bigint。

表 3-10 显示了这些数据类型的映射情况：

表 3-10：alter table 的 DirectConnect 数据类型转换

Adaptive Server 数据类型	DirectConnect 缺省数据类型
binary(*n*)	binary(*n*)
bit	bit
char	char
date	date
datetime	datetime
decimal(*p*, *s*)	decimal(*p*, *s*)
float	float
image	image
int	int
money	money
numeric(*p*, *s*)	numeric(*p*, *s*)
nchar(*n*)	nchar(*n*)
nvarchar(*n*)	nvarchar(*n*)
real	real
smalldatetime	smalldatetime
smallint	smallint
smallmoney	smallmoney
time	time
timestamp	timestamp
tinyint	tinyint
text	text
unichar	unichar
unitext	unitext
varbinary(*n*)	varbinary(*n*)
varchar(*n*)	varchar(*n*)

用法

当服务器接收到 alter table 命令时，它将命令传递给相应的访问方法，如果：

- 要执行该命令的对象已与远程或外部存储位置相关联。
- 命令由一个 add column 请求组成。没有将添加或删除约束的请求传递给访问方法；而是进行本地处理。

alter table 作为语言请求传递到远程服务器。

另请参见

《参考手册》中的 alter table

case

服务器类 *ASEnterprise*

如果初始查询语法中存在 case 表达式，并不会导致查询优化程序拒绝快速传递模式。

服务器类 *ASAnywhere*

如果初始查询语法中存在 case 表达式，并不会导致查询优化程序拒绝快速传递模式。

服务器类 *ASIQ*

没有为此类服务器设置处理 case 表达式的功能。在优化包含 case 表达式的 SQL 语句时，如果存在 case 表达式，则会导致组件集成服务快速传递优化拒绝该语句。一旦发生这种情况，本地 Adaptive Server 在从远程服务器检索数据后必须计算出 case 表达式的值。

服务器类 *direct_connect*

处理 case 表达式的能力由 RPC sp_capabilities 的结果集决定。如果 direct_connect 指示它可以处理 case 表达式，则当使用快速传递模式处理查询时，组件集成服务就会将它们转移到 direct_connect。

另请参见

《参考手册》中的 case。

connect to...disconnect

服务器类 ASEnterprise

当发出 disconnect 时，组件集成服务将 disconnect 转移到远程服务器，以便从直通模式中将其关闭。如果未处于直通模式，则可能会发生语法错误，但组件集成服务忽略这些错误，并且不会将它们转移到客户端。

服务器类 *ASAnywhere*

当发出 connect 或 disconnect 时，不会与 ASAnywhere 发生交互。

服务器类 *ASIQ*

发出 connect 或 disconnect 时，不会与 ASIQ 发生交互。

服务器类 direct_connect

使用 direct_connect 类服务器发出 connect 时，将向 direct_connect 发送一个 RPC：

```
sp_thread_props “passthru mode”, 1
```

发出 disconnect 时，如果已建立直通模式连接的服务器为 direct_connect，则将向 direct_connect 发送一个 RPC：

```
sp_thread_props “passthru mode”, 0
```

另请参见

《参考手册》中的 commit

create existing table

服务器类 *ASEnterprise*

- 表 3-11 描述了将远程 Adaptive Server 列映射到本地代理表的列时可以使用的数据类型：

表 3-11：create existing table 的 Adaptive Server 数据类型转换

远程 Adaptive Server 数据类型	允许使用的 Adaptive Server 数据类型
binary(*n*)	image、 binary(*n*) 和 varbinary(*n*) ； 如果不为 image，则长度必须匹配
bit	bit、 decimal、 float、 int、 money、 numeric、 real、 smallint、 smallmoney 和 tinyint
char(*n*)	text、 nchar(*n*)、 nvarchar(*n*)、 char(*n*)、 varchar(*n*)、 unichar、 univarchar ； 如果不为 text，则长度必须匹配
datetime	datetime、 smalldatetime、 char 和 varchar
decimal(*p*, *s*)	bit、 decimal、 float、 int、 money、 numeric、 real、 smallint、 smallmoney 和 tinyint
float	bit、 decimal、 float、 int、 money、 numeric、 real、 smallint、 smallmoney 和 tinyint
image	image
int	bit、 decimal、 float、 int、 money、 numeric、 real、 smallint、 smallmoney 和 tinyint
money	bit、 decimal、 float、 int、 money、 numeric、 real、 smallint、 smallmoney 和 tinyint
nchar(*n*)	text, nchar(*n*), nvarchar(*n*), char(*n*), varchar(*n*) ； 如果不为 text，则长度必须匹配
numeric(*p*, *s*)	bit、 decimal、 float、 int、 money、 numeric、 real、 smallint、 smallmoney 和 tinyint
nvarchar(*n*)	text、 nchar(*n*)、 nvarchar(*n*)、 char(*n*)、 varchar(*n*)、 unichar、 univarchar ； 如果不为 text，则长度必须匹配
real	bit、 decimal、 float、 int、 money、 numeric、 real、 smallint、 smallmoney 和 tinyint
smalldatetime	datetime、 smalldatetime、 char 和 varchar
smallint	bit、 decimal、 float、 int、 money、 numeric、 real、 smallint、 smallmoney 和 tinyint
smallmoney	bit、 decimal、 float、 int、 money、 numeric、 real、 smallint、 smallmoney 和 tinyint
text	text、 unitext
timestamp	timestamp
tinyint	bit、 decimal、 float、 int、 money、 numeric、 real、 smallint、 smallmoney 和 tinyint
unichar	char、 varchar、 unichar、 univarchar、 text、 datetime 和 smalldatetime

远程 Adaptive Server 数据类型	**允许使用的 Adaptive Server 数据类型**
univarchar	char、varchar、unichar、univarchar、text、datetime 和 smalldatetime
unitext	unitext
varbinary(*n*)	image、binary(*n*) 和 varbinary(*n*)；如果不为 image，则长度必须匹配
varchar(*n*)	text、nchar(*n*)、nvarchar(*n*)、char(*n*)、varchar(*n*)、unichar、univarchar；如果不为 text，则长度必须匹配
date	
time	
bigint	隐式：binary、varbinary、bit、tinyint、smallint、int、decimal、numeric、float、real、money、smallmoney 显式：char、varchar、unichar、univarchar
unsigned tinyint	隐式：binary、varbinary、bit、tinyint、smallint、unsigned smallint、int、unsigned int、bigint、unsigned bigint、decimal、numeric、float、real money、smallmoney 显式：char、varchar、unichar、univarchar 不支持：text、image、date、time、datetime、smalldatetime
unsigned smallint	隐式：binary、varbinary、bit、tinyint、smallint、unsigned smallint、int、unsigned int、bigint、unsigned bigint、decimal、numeric、float、real money、smallmoney 显式：char、varchar、unichar、univarchar 不支持：text、image、date、time、datetime、smalldatetime
unsigned int	隐式：binary、varbinary、bit、tinyint、smallint、unsigned smallint、int、unsigned int、bigint、unsigned bigint、decimal、numeric、float、real money、smallmoney 显式：char、varchar、unichar、univarchar 不支持：text、image、date、time、datetime、smalldatetime

远程 Adaptive Server 数据类型	允许使用的 Adaptive Server 数据类型
unsigned bigint	隐式：binary、 varbinary、 bit、 tinyint、 smallint、 unsigned smallint、 int、 unsigned int、 bigint、 unsigned bigint、 decimal、 numeric、 float、 real money、 smallmoney 显式：char、 varchar、 unichar、 univarchar 不支持：text、 image、 date、 time、 datetime、 smalldatetime

注释　组件集成服务仅支持对 Adaptive Server 15.0 和更高版本执行 unitext。

服务器类 *ASAnywhere*

- 表 3-12 描述了将远程 Adaptive Server 列映射到本地代理表的列时可以使用的数据类型：

表 3-12：create existing table 的 Adaptive Server Anywhere 数据类型转换

远程 Adaptive Server Anywhere 数据类型	允许使用的 Adaptive Server Anywhere 数据类型
binary(*n*)	image、 binary(*n*) 和 varbinary(*n*)；如果不为 image，则长度必须匹配
bit	bit、 decimal、 float、 int、 money、 numeric、 real、 smallint、 smallmoney 和 tinyint
char(*n*)	text、 nchar(*n*)、 nvarchar(*n*)、 char(*n*)、 varchar(*n*)、 unichar、 univarchar；如果不为 text，则长度必须匹配
datetime	datetime 和 smalldatetime
decimal(*p*, *s*)	bit、 decimal、 float、 int、 money、 numeric、 real、 smallint、 smallmoney 和 tinyint
float	bit、 decimal、 float、 int、 money、 numeric、 real、 smallint、 smallmoney 和 tinyint
image	image
int	bit、 decimal、 float、 int、 money、 numeric、 real、 smallint、 smallmoney 和 tinyint
money	bit、 decimal、 float、 int、 money、 numeric、 real、 smallint、 smallmoney 和 tinyint
numeric(*p*, *s*)	bit、 decimal、 float、 int、 money、 numeric、 real、 smallint、 smallmoney 和 tinyint
real	bit、 decimal、 float、 int、 money、 numeric、 real、 smallint、 smallmoney 和 tinyint
smalldatetime	datetime 和 smalldatetime
smallint	bit、 decimal、 float、 int、 money、 numeric、 real、 smallint、 smallmoney 和 tinyint
smallmoney	bit、 decimal、 float、 int、 money、 numeric、 real、 smallint、 smallmoney 和 tinyint
text	text
timestamp	timestamp
tinyint	bit、 decimal、 float、 int、 money、 numeric、 real、 smallint、 smallmoney 和 tinyint
varbinary(*n*)	image、 binary(*n*) 和 varbinary(*n*)、 unichar、 unitext、 univarchar；如果不为 image，则长度必须匹配
varchar(*n*)	text、 nchar(*n*)、 nvarchar(*n*)、 char(*n*)、 varchar(*n*)、 unichar、 univarchar；如果不为 text，则长度必须匹配

远程 Adaptive Server Anywhere 数据类型	允许使用的 Adaptive Server Anywhere 数据类型
date	
time	
bigint	隐式：binary、 varbinary、 bit、 tinyint、 smallint、 unsigned smallint、 int、 unsigned int、 bigint、 unsigned bigint、 decimal、 numeric、 float、 real money、 smallmoney 显式：char、 varchar、 unichar、 univarchar 不支持：text、 image、 date、 time、 datetime、 smalldatetime
unsigned tinyint	隐式：binary、 varbinary、 bit、 tinyint、 smallint、 unsigned smallint、 int、 unsigned int、 bigint、 unsigned bigint、 decimal、 numeric、 float、 real money、 smallmoney 显式：char、 varchar、 unichar、 univarchar 不支持：text、 image、 date、 time、 datetime、 smalldatetime
unsigned smallint	隐式：binary、 varbinary、 bit、 tinyint、 smallint、 unsigned smallint、 int、 unsigned int、 bigint、 unsigned bigint、 decimal、 numeric、 float、 real money、 smallmoney 显式：char、 varchar、 unichar、 univarchar 不支持：text、 image、 date、 time、 datetime、 smalldatetime
unsigned int	隐式：binary、 varbinary、 bit、 tinyint、 smallint、 unsigned smallint、 int、 unsigned int、 bigint、 unsigned bigint、 decimal、 numeric、 float、 real money、 smallmoney 显式：char、 varchar、 unichar、 univarchar 不支持：text、 image、 date、 time、 datetime、 smalldatetime
unsigned bigint	隐式：binary、 varbinary、 bit、 tinyint、 smallint、 unsigned smallint、 int、 unsigned int、 bigint、 unsigned bigint、 decimal、 numeric、 float、 real money、 smallmoney 显式：char、 varchar、 unichar、 univarchar 不支持：text、 image、 date、 time、 datetime、 smalldatetime

远程 Adaptive Server Anywhere 数据类型	允许使用的 Adaptive Server Anywhere 数据类型
nchar(n)	text、 nchar(n)、 nvarchar(n)、 char(n)、 varchar(n)、 unichar、 univarchar ； 如果不为 text，则长度必须匹配
nvarchar(n)	text、 nchar(n)、 nvarchar(n)、 char(n)、 varchar(n)、 unichar、 univarchar ； 如果不为 text，则长度必须匹配

服务器类 *ASIQ*

- text 和 image 数据类型受 ASIQ 12.6 版支持并需要一个许可证。
- 与服务器类 ASAnywhere 的行为相同。

服务器类 *direct_connect*

- RPC sp_columns 查询现有表中列的数据类型。
- 本地列的数据类型不必与远程列的数据类型相同，但它们必须可以互相转换 （如表 3-13 所示）。否则，将出现列类型错误，而且中止该命令。

表 3-13：create existing table 的 DirectConnect 数据类型转换

DirectConnect 数据类型	允许使用的 Adaptive Server 数据类型
binary(*n*)	image、binary(*n*)、varbinary(*n*)；如果长度不匹配，则中止该命令
binary(16)	timestamp
bit	bit、decimal、float、int、money、numeric、real、smallint、smallmoney 和 tinyint
char(*n*)	text、nchar(*n*)、nvarchar(*n*)、char(*n*) 和 varchar(*n*)、unichar、univarchar；如果长度不匹配，则中止该命令
datetime	datetime、smalldatetime
decimal(*p*, *s*)	bit、decimal、float、int、money、numeric、real、smallint、smallmoney 和 tinyint
float	bit、decimal、float、int、money、numeric、real、smallint、smallmoney 和 tinyint
image	image
int	bit、decimal、float、int、money、numeric、real、smallint、smallmoney 和 tinyint
money	bit、decimal、float、int、money、numeric、real、smallint、smallmoney 和 tinyint
nchar(*n*)	text、nchar(*n*)、nvarchar(*n*)、char(*n*) 和 varchar(*n*)、unichar、univarchar；如果长度不匹配，则中止该命令
numeric(p, s)	bit、decimal、float、int、money、numeric、real、smallint、smallmoney 和 tinyint
nvarchar(*n*)	text、nchar(*n*)、nvarchar(*n*)、char(*n*) 和 varchar(*n*)、unichar、univarchar；如果长度不匹配，则中止该命令
real	bit、decimal、float、int、money、numeric、real、smallint、smallmoney 和 tinyint
smalldatetime	datetime、smalldatetime
smallint	bit、decimal、float、int、money、numeric、real、smallint、smallmoney 和 tinyint
smallmoney	bit、decimal、float、int、money、numeric、real、smallint、smallmoney 和 tinyint
text	text
timestamp	timestamp、binary(8)、varbinary(8)
unichar	text、nchar(*n*)、nvarchar(*n*)、char(*n*)、varchar(*n*)、unichar、univarchar；如果不为 text，则长度必须匹配

DirectConnect 数据类型	允许使用的 Adaptive Server 数据类型
univarchar	text、nchar(*n*)、nvarchar(*n*)、char(*n*)、varchar(*n*)、unichar、univarchar；如果不为 text，则长度必须匹配
date	
time	
bigint	UDB 和 DC/Microsoft 支持 bigint。

- 数据类型信息在与该参数相关联的 CS_DATAFMT 结构中传递。下面的结构字段包含数据类型信息：
  - *datatype* — 表示 Adaptive Server 数据类型的 CS_Library 数据类型。例如， CS_INT_TYPE。
  - *usertype* — 本机 DBMS 数据类型。在执行 create existing table 命令的过程中， sp_columns 将此数据类型作为其结果集的一部分传回到组件集成服务 （参见 《参考手册》中的 sp_columns）。Adaptive Server 在参数的 *usertype* 字段中返回此数据类型，以帮助 DirectConnect 进行数据类型转换。

用法

当接收到 create existing table 命令时，将该命令解释为请求从对象远程或外部位置导入元数据，以更新系统目录。通过将三个 RPC 发送到与对象关联的远程服务器来导入此元数据：

- sp_tables — 检验远程对象实际是否存在。
- sp_columns — 获得远程对象的列属性，以便与 create existing table 中定义的列属性相比较。
- sp_statistics — 获得索引信息以更新本地系统表 sysindexes。

另请参见

《参考手册》中的 create existing table

create index

服务器类 *ASEnterprise*

组件集成服务将除 on *segment_name* 以外的所有子句都转移到远程服务器。

服务器类 *ASAnywhere*

组件集成服务将除 on *segment_name* 以外的所有子句都转移到远程服务器。

服务器类 *ASIQ*

组件集成服务将除 on *segment_name* 以外的所有子句都转移到远程服务器。

服务器类 *direct_connect*

- 在将语言功能设置为 “Transact-SQL” 时，组件集成服务将除 max_rows_per_page 和 on *segment_name* 子句以外的所有语法转移到远程服务器。

用法

如果要执行该命令的对象已与远程或外部存储位置相关联，则当服务器接收到 create index 命令时，服务器将该命令传递给相应的访问方法。

该命令使用该类的相应语法重新构建，然后传递到远程服务器上执行。

create index 作为语言请求传递到远程服务器。

另请参见

《参考手册》中的 create index

create table

服务器类 *ASEnterprise*

组件集成服务在不进行转换的情况下，将每一列的数据类型传递到远程服务器。

服务器类 *ASAnywhere*

组件集成服务在不进行转换的情况下，将每一列的数据类型传递到远程服务器。

服务器类 *ASIQ*

组件集成服务在不进行转换的情况下，将每一列的数据类型传递到远程服务器。

服务器类 *direct_connect*

- 组件集成服务重新构造 create table，并将命令传递给目标 DirectConnect。网关将把这些命令转换为基础 DBMS 可以识别的形式。
- Direct Connect 不支持 bigint、 unsigned tinyint、 unsigned smallint、 unsigned int、 unsigned bigint。
- 将 Adaptive Server 数据类型转换为 DirectConnect 语法模式数据类型，（如表 3-14 所示）。

表 3-14：create table 的 DirectConnect 数据类型转换

Adaptive Server 数据类型	DirectConnect 缺省数据类型
binary*(n)*	binary*(n)*
bit	bit
char	char
datetime	datetime
decimal*(p, s)*	decimal*(p, s)*
float	float
image	image
int	int
money	money
numeric*(p, s)*	numeric*(p, s)*
nchar(n)	nchar(n)
nvarchar(n)	nvarchar(n)
real	real
smalldatetime	smalldatetime
smallint	smallint
smallmoney	smallmoney
timestamp	timestamp
tinyint	tinyint
text	text
unichar(n)	unichar
univarchar(n)	char*(n)*，用于 bit 数据
varbinary*(n)*	varbinary*(n)*
varchar*(n)*	varchar*(n)*
date	
time	
bigint	UDB 和 DC/Microsoft 支持 bigint

用法

当服务器接收到 create table 命令时，就会将该命令解释为请求创建新表。服务器调用要创建表的服务器类的相应访问方法。如果它为远程的，则创建表。如果此命令成功，就会对系统目录进行更新，并将对象作为创建它的数据库中的本地表显示到客户端。

使用服务器类的相应语法重新构建 create table。例如，如果服务器类是 direct_connect 并且远程服务器类是 DB2，则该命令在传递到远程服务器之前使用 Adaptive Server Anywhere 语法重新构建。对于 Adaptive Server 环境特有的数据类型，将进行数据类型转换。

有些服务器类对于能支持或不能支持的数据类型有限制。

create table 作为语言请求传递到远程服务器。

另请参见 《参考手册》中的 create table

delete

服务器类 *ASEnterprise* 如果组件集成服务不能按原样转移原始查询，它会采用第 2 种方法执行 delete 命令。

服务器类 *ASAnywhere* 如果组件集成服务不能按原样转移原始查询，它会采用第 2 种方法执行 delete 命令。

服务器类 *ASIQ* 如果组件集成服务不能按原样转移原始查询，则会发生错误，因为 ASIQ 不支持可更新游标。

服务器类 *direct_connect*

- 转移到 direct_connect 类服务器的语法取决于功能协商，功能协商是在组件集成服务第一次连接到远程 DirectConnect 时完成的。可协商功能的示例包括：子查询支持、 group by 支持和内置支持。
- 组件集成服务将数据值作为参数传递给游标或动态 SQL 语句。如果 DirectConnect 支持语言语句，则也可使用语言语句。这些参数采用 Adaptive Server 本机数据类型， DirectConnect 必须将它们转换为适合目标 DBMS 的格式。

另请参见 《参考手册》中的 delete

drop index

服务器类 *ASEnterprise* 组件集成服务将以下 drop index 语法转移到配置为 ASEnterprise 类的远程服务器：

drop index *table_name.index_name*

组件集成服务在执行此语句之前，先执行 use database 命令，因为 drop index 语法不允许指定数据库名。

服务器类 *ASAnywhere*

- 组件集成服务将以下 drop index 语法转移到配置为 ASAnywhere 类的远程服务器：

  drop index *table_name.index_name*

  组件集成服务在执行此语句之前，先执行 use database 命令，因为 drop index 语法不允许指定数据库名。

服务器类 *ASIQ*

组件集成服务将以下 drop index 语法转移到配置为 ASIQ 类的远程服务器：

drop index *table_name.index_name*

组件集成服务在执行此语句之前，先执行 use database 命令，因为 drop index 语法不允许指定数据库名。

服务器类 *direct_connect*

组件集成服务将以下 drop index 语法转移到配置为 direct_connect 类的远程服务器：

drop index *table_name.index_name*

用法

如果要执行该命令的对象已与远程或外部存储位置相关联，则当服务器接收到 drop index 命令时，服务器将该命令传递给相应的访问方法。

drop index 使用该类的相应语法重新构建，并传递到远程服务器上执行。

此命令作为语言请求传递到远程服务器。

另请参见

《参考手册》中的 drop index

fetch

服务器类 *ASEnterprise*

如果此游标是只读的，则在打开游标后，组件集成服务在接收到第一个 fetch 时向远程服务器发送一个语言请求。否则，组件集成服务通过 Client-Library 声明一个面向远程服务器的游标。

服务器类 *ASAnywhere*

对 fetch 语句的处理与 ASEnterprise 相同。

服务器类 *ASIQ*

当打开游标后，组件集成服务在接收到第一个 fetch 请求时，向远程服务器发送一个语言请求。

服务器类 *direct_connect*

组件集成服务处理此类服务器的方式与 ASEnterprise 中服务器的处理方式相同。

另请参见

close、 deallocate cursor、 declare cursor、 open

《参考手册》中的 fetch

insert

服务器类 *ASEnterprise*

- 完全支持使用 values 关键字的 insert 命令。
- 除 text 和 image 之外的所有数据类型都支持使用 select 命令的 insert 命令。只有在 text 和 image 类型的列包含空值时，才支持它们。
- 如果所有 insert 和 select 表位于同一个远程服务器上，则将整个语句转移到远程服务器上执行。这称为 “快速传递模式”。如果 select 语句不符合 select 命令的所有快速传递规则，则不使用快速传递模式。
- 如果 select 表位于一个远程服务器上，而 insert 表位于另一个服务器上，则组件集成服务从源表中选择每一行，然后将其插入目标表中。
- 您无法 insert 到计算列中。

服务器类 *ASAnywhere*

对 insert 语句的处理与 ASEnterprise 相同。

服务器类 *ASIQ*

对 insert 语句的处理与 ASEnterprise 相同。

服务器类 *direct_connect*

- 完全支持使用 values 关键字的 insert 命令。
- 完全支持使用select命令的 insert 命令，但是如果表中存在 text 或 image 列，则此表必须具有唯一的索引。使用带 select 命令的 insert 时，如果符合以下条件，则将整个命令发送到远程服务器：
  - 命令中引用的所有表均位于远程服务器上。
  - 来自 DirectConnect 的功能响应表示支持 insert-select 命令。
  - 如果使用 TopN 功能，则您必须具有 order by 子句。

  如果两种条件都不符合，则组件集成服务从源表中选择每一行，然后将其插入目标表。
- 组件集成服务将数据值作为参数传递给游标或动态 SQL 语句。如果 DirectConnect 支持语言语句，则也可使用语言语句。这些参数采用 Adaptive Server 本机数据类型， DirectConnect 必须将它们转换为适合目标 DBMS 的格式。

另请参见

《参考手册》中的 insert

readtext

服务器类 *ASEnterprise*

当基础表为代理表时，组件集成服务将以下语法转移到远程服务器：

```
readtext [[database.]owner.]table_name.column_name
 text_pointer offset size
[using {chars | characters}]
```

服务器类 *ASAnywhere*

对 readtext 语句的处理与处理 ASEnterprise 相同。

服务器类 *ASIQ*

对 readtext 语句的处理与处理 ASEnterprise 相同。

服务器类 *direct_connect*

- 如果 DirectConnect 不支持文本指针，则不能发送 readtext，使用它会出现错误。
- 如果 DirectConnect 支持文本指针，则组件集成服务将以下语法转移到远程服务器：

  ```
  readtext
   [[database.]owner.]table_name.column_name
  text_pointer offset size
   [using {chars | characters}]
  ```

- 当必须读取 text 或 image 数据时，将发出 readtext。当 select 命令在选择列表中引用了 text 或 image 列时，或者当 where 子句引用了 text 或 image 列时，将调用 readtext 。

  例如，将代理表 books 映射到远程服务器 foo 上的表 books。其中的列分别为 id、 name 和 blurb （text 类型）。当发出下述语句时：

  ```
  select * from books
  ```

  组件集成服务将以下语法发送到远程服务器：

  ```
  select id, name, textptr(blurb) from foo_books
  readtext foo_books.blurb @p1 0 0 using chars
  ```

另请参见

《参考手册》中的 readtext

select

服务器类 *ASEnterprise*

- 支持所有语法。由于假定远程服务器具有处理 Transact-SQL 语法所需的各种能力，因此使用快速传递模式将 select 命令的所有元素（上面提到的除外）转移到远程服务器。
- 发出 select...into 命令且 into 表位于远程 Adaptive Server 上时，使用批量复制传送将数据复制到新表中。必须通过将 dboption 设置为 select into / bulkcopy 来配置本地和远程数据库。

服务器类 *ASAnywhere*

- 支持所有语法。由于假定远程服务器具有处理 Transact-SQL 语法所需的各种能力，因此使用快速传递模式将 select 命令的所有元素（上面提到的除外）转移到远程服务器。
- 如果使用 select...into 格式，且通过 ASAnywhere 接口访问 into 表，则不使用批量插入功能。相反，组件集成服务使用 Client-Library 来准备参数化的动态 insert 命令，并为该命令的 select 部分所返回的每一行执行此命令。

服务器类 *ASIQ*

- 支持所有语法。由于假定远程服务器具有处理 Transact-SQL 语法所需的各种能力，因此使用快速传递模式将 select 命令的所有元素（上面提到的除外）转移到远程服务器。

服务器类 *direct_connect*

- 当组件集成服务首次需要连接到 direct_connect 类服务器时，将对 DirectConnect 发出功能请求。组件集成服务根据响应来决定将 select 命令的哪些部分转移到 DirectConnect。大多数情况下，这取决于与 DirectConnect 交互的 DBMS 的功能。
- 如果不能使用快速传递模式将整个语句转移到 DirectConnect，组件集成服务将对不能转移的功能进行补偿。例如，如果远程服务器不能处理 order by 子句，则不使用快速传递，并且组件集成服务对结果集执行排序。
- 组件集成服务将数据值作为参数传递给游标或动态 SQL 语句。如果 DirectConnect 支持语言语句，则也可使用语言语句。这些参数采用 Adaptive Server 本机数据类型， DirectConnect 必须将它们转换为适合目标 DBMS 的格式。
- 支持 select...into 命令，但如果表中存在 text 或 image 列，则此表必须具有唯一的索引。
- 如果使用 select...into 格式，且通过 DirectConnect 访问 into 表，则不使用批量插入功能。相反，组件集成服务使用 Client-Library 来准备动态 insert 命令，然后为该命令的 select 部分返回的每一行执行此命令。

另请参见

《参考手册》中的 select

truncate table

服务器类 *ASEnterprise*

组件集成服务将 truncate table 命令转移到 ASEnterprise 类服务器。

服务器类 *ASAnywhere*

组件集成服务将 truncate table 命令转移到 ASAnywhere 类服务器。

服务器类 *ASIQ*

组件集成服务将 truncate table 命令转移到 ASIQ 类服务器。

服务器类 *direct_connect* 和 *sds*

发送 Transact-SQL 语法：

truncate table [[*database*.]*owner*.]*table_name*

另请参见

《参考手册》中的 truncate table

update

服务器类 *ASEnterprise*

- 如果组件集成服务不能将整个语句传递到远程服务器，则表上必须存在唯一索引。
- 除 text 和 image 以外的所有数据类型都完全支持使用 update 命令。除非将 text 或 image 值设置为空，否则不能使用 update 命令更改 text 和 image 数据。而应使用 writetext 命令。
- 如果未使用快速传递模式，则从源表检索数据，然后使用为处理定位 update 而设计的单独游标来更新目标表中的值。

服务器类 *ASAnywhere*

对 update 语句的处理与处理 ASEnterprise 相同。

服务器类 *ASIQ*

对 update 语句的处理与处理 ASEnterprise 相同。

如果组件集成服务不能按原样转移原始查询，则会发生错误，因为 ASIQ 不支持可更新游标。

服务器类 *direct_connect*

- direct_connect 类服务器支持以下语法：

```
update [[database.]owner.]{table_name | view_name}
 set [[[database.]owner.]{table_name.|view_name.}]
    column_name1 =
       {expression1|NULL|(select_statement)}
    [, column_name2 =
       {expression2|NULL|(select_statement)}]...
 [where search_conditions]
```

如果来自远程服务器的功能响应表示支持命令的所有元素，则符合此语法的 update 命令均使用快速传递模式。可协商功能的示例包括：子查询支持、 group by 支持和内置支持。

- 如果远程服务器不支持此命令的所有元素，或者此命令包含 from 子句，则组件集成服务发出一个查询来获得 set 子句的值，然后向远程服务器发出一个 update 命令。
- 组件集成服务将数据值作为参数传递给游标或动态 SQL 语句。如果 DirectConnect 支持语言语句，则也可使用语言语句。这些参数采用 Adaptive Server 本机数据类型， DirectConnect 必须将它们转换为适合目标 DBMS 的格式。

另请参见　《参考手册》中的 update

update statistics

服务器类
ASEnterprise

- 如果请求检索统计信息的表没有索引，组件集成服务将发出如下命令：

  ```
  select count(*) from table_name
  ```

  这也是在跟踪标志 11209 打开时发出的唯一命令。

- 如果表带有索引，而且在命令中指定了此索引，组件集成服务将发出如下命令：

  ```
  select count(*) from table_name

  select count(*) column_name [,column_name, ...]
   from table_name
   group by column_name [,column_name, ..]
  ```

  列名代表组成索引的一个或多个列。

  例如，发出如下命令时：

  ```
  update statistics customers ind_name
  ```

  组件集成服务发出以下命令：

  ```
  select count(*) from customers

  select count(*) last_name, first_name
   from customers
   group by last_name, first_name
  ```

- 如果表带有一个或多个索引，但语句中未指定索引，组件集成服务将发出 select count (*)，然后为每个索引发出 select/order by 命令。
- 您必须使用远程登录使 sa_role 运行代理表上的 update statistic。

- 在 Adaptive Server 15.0 或更高版本中，如果代理表指向分区表格，则只导入全局统计信息。这些代理表中的被集合的统计信息，在 Adaptive Server 15.0 中不进行分区。

服务器类 *ASAnywhere*

此服务器类中对 update statistics 的处理与 Adaptive Server 15.0 以前的服务器版本相同。

服务器类 *ASIQ*

此服务器类中对 update statistics 的处理与 Adaptive Server 15.0 以前的服务器版本相同。

服务器类 *direct_connect*

- 此服务器类对 update statistics 的处理与上述 ASEnterprise 服务器类的处理方法相同。
- 如果 direct_connect 指示不能处理 group by 或 count(*) 语法，则不会为 direct_connect 收集统计信息。

另请参见

《参考手册》中的 update statistics

writetext

服务器类 *ASEnterprise*

利用与远程服务器的单独连接处理 writetext 命令。

服务器类 *ASAnywhere*

利用与远程服务器的单独连接处理 writetext 命令。

服务器类 *ASIQ*

利用与远程服务器的单独连接处理 writetext 命令。

服务器类 *direct_connect*

如果 DirectConnect 支持文本指针，则组件集成服务将 DirectConnect 当作一个 ASEnterprise 类服务器。

另请参见

《参考手册》中的 writetext

直通模式

组件集成服务中提供了直通模式，使用户能够在以直通模式连接到的服务器上执行本机操作。

例如，通过请求 Oracle 服务器的直通模式，可以将本机 Oracle SQL 语句发送到 Oracle DBMS。结果转换为 Open Client 应用程序可用的格式并传回给用户。

这种模式绕过 Transact-SQL 语法分析程序和编译器，从用户处收到的每批语言都将直接传递给用户以直通模式连接到的服务器。每批的结果将返回给客户端。

可通过以下几种方法使用直通模式：

- connect to
- sp_autoconnect
- sp_passthru
- sp_remotesql

connect to

用户可使用 connect to 命令指定要求直通连接的服务器。此命令的语法如下：

```
connect to server_name
```

其中 *server_name* 是添加到 *sysservers* 表的服务器名称，其服务器类和网络名已定义。请参见《参考手册》中的 sp_addserver。

代表用户与 *server_name* 建立连接时，服务器使用：

- 远程登录别名集（使用 sp_addexternlogin）或者
- 用于与 Adaptive Server 通信的名称和口令。

在两种情况下，如果无法建立与指定服务器的连接，返回给用户的消息中将包含原因。

一旦建立了直通连接，在接收后续语言文本时，将绕过 Transact-SQL 语法分析程序和编译器。服务器接收的任何语句将直接传递给指定的远程服务器。

注释 有些数据库管理系统不能一次识别多个语句，例如，如果将多个 select 语法作为单个语言文本缓冲区的一部分来接收，就会发生语法错误。

在将语句传递到请求服务器后，将所有结果转换为 Open Client 接口可识别的格式并发回到客户程序。

要退出直通模式，请发出 disconnect、或 disc 命令。此客户端的后续语言文本随后将使用 Transact-SQL 语法分析程序和编译器进行处理。

必须由系统管理员显式授予使用 connect to 的权限。语法为：

```
grant connect to user_name
```

若要撤消使用 connect to 的权限，请使用以下语法：

```
revoke connect from user_name
```

connect to 权限存储于 master 数据库中。若要全局授予或撤消 “public” 的权限，系统管理员需在 master 数据库中设置相应的权限；其影响是服务器范围的，即与所使用的数据库无关。只有当用户是 master 数据库的有效用户时，系统管理员才能授予或撤消其权限。

系统管理员可以为任何数据库中的 “public” 组授予或撤消 “all” 权限。如果权限在 master 数据库中，则 “all” 包含 connect to 命令。如果权限在其它数据库中，则 “all” 不包含 connect to 命令。

示例

系统管理员要撤消 “public” 权限，而只想让用户 “fred” 来执行 connect to 命令。“fred” 必须是 master 的有效用户。为此，系统管理员在 master 中发出以下命令：

```
revoke connect from public
 sp_adduser fred
 grant connect to fred
```

sp_autoconnect

有些用户可能总是需要与给定服务器的直通连接。在这种情况下，可以对组件集成服务进行配置，使其在用户连接服务器时自动以直通模式将这些用户连接到指定的远程服务器。可以通过 sp_autoconnect 并使用以下语法来启用和禁用此功能：

sp_autoconnect *server_name*, true|false [,*loginname*]

使用 sp_autoconnect 之前，请使用 sp_addserver 将 *server_name* 添加到 sysservers。

用户可以使用 sp_autoconnect 请求与服务器的自动连接，但只有系统管理员可以启用或禁用与另一用户的自动直通连接。因此，只有系统管理员可以指定此过程的第三个参数。

如果第二个参数为 true， autoconnect 功能将对当前用户 （或在第三个参数中指定的用户）可用。如果第二个参数为 false，则会禁用 autoconnect。

当用户连接服务器时，将在 syslogins 中检查该用户的 autoconnect 状态。如果启用，将对也是在 syslogins 中找到的 *server_name*（由 sp_autoconnect 将其放置于此）进行有效性检查。如果服务器有效，则将用户自动连接到该服务器并设置直通状态。服务器从此用户接收的后续语言语句的处理方式与用户显式输入 connect 命令时的处理方式完全相同。此用户随即查看该服务器，就如同查看远程服务器的直通网关一样。

当 “自动连接的”用户执行 disconnect 时，通常会返回到该服务器。

如果无法到达远程服务器，用户 （除非为该用户指派了 “sa” 角色）将不会连接到本地 Adaptive Server。此时，将返回 “login failed” 错误消息。

sp_passthru

用户可通过 sp_passthru 将 SQL 命令缓冲区传递给远程服务器。假定所传递的 SQL 语句的语法是接收缓冲区的服务器类的本机语法；不执行任何转换或解释。远程服务器的结果被放置在输出参数中 （可选）。sp_passthru 的语法如下：

sp_passthru *server, command, errcode, errmsg, rowcount*
[*, arg1, arg2, ... arg*n]

其中：

- *server* 是要接收 SQL 命令缓冲区的服务器的名称；数据类型为 varchar(255)。
- *command* 是 SQL 命令缓冲区；数据类型为 varchar(255)。

- *errcode* 是远程服务器返回的错误代码；数据类型为 int output。
- *errmsg* 是远程服务器返回的错误消息；数据类型为 varchar(1024) output。
- *rowcount* 是受命令缓冲区中最后一条命令影响的行数；数据类型为 int output。
- *arg1* —*argn* 是可选参数。如果提供了这些输出参数，它们将接收命令缓冲区最后一条命令所返回的最后一行的结果。数据类型可能有所不同。所有数据类型必须为输出参数。

示例

```
sp_passthru ORACLE, "select date from dual",
@errcodeoutput, @errmsg output, @rowcount output,
@oradate output
```

此示例返回输出参数 @oradate 中来自 Oracle 服务器的日期。如果发生 Oracle 错误，则将错误代码放置在 @errcode 中，而将相应消息放置在 @errmsg 中。将 @rowcount 参数设置为 1。

有关 sp_passthru 及其返回状态的详细信息，请参见 《参考手册》。

sp_remotesql

sp_remotesql 允许将本机语法传递到远程服务器。该过程建立与远程服务器的连接，传递查询缓冲区，然后再将结果传递回客户端。
sp_remotesql 的语法如下：

```
sp_remotesql server_name, query_buf1
[, query_buf2, ... , query_buf254]
```

其中：

- *server_name* 是使用 sp_addserver 定义的服务器的名称。
- *server_name* 是一个 varchar(255) 字段。如果 *server_name* 未定义或不可用，则连接失败并中止该过程。该参数是必需的。
- *query_buf1* 是类型为 char 或 varchar 的查询缓冲区，最大长度为 255 个字节。该参数是必需的。

每个附加缓冲区为 char 或 varchar，最大长度为 255 个字节。如果提供了这些可选参数，它们将与 *query_buf1* 的内容一起并置到单个查询缓冲区中。

示例

```
sp_remotesql freds_server, "select @@version"
```

在此示例中，服务器将查询缓冲区传递到 *freds_server*，后者解释 select *@@version* 语法并将版本信息返回客户端。此服务器不解释返回的信息。

有关 sp_remotesql 及其返回代码的详细信息，请参见 《参考手册》。

带引号标识符支持

带引号的标识符将转移到支持它们的远程服务器。由 set 命令触发该过程：

```
set quoted_identifier on
```

如果启用了此线程属性，组件集成服务在向远程服务器发送 SQL 语句之前，用引号将标识符引起来。

远程服务器必须能够支持带引号标识符。在 sp_capabilities 结果集中为此目的保留了一个功能：

- 功能 ID：135
- 功能名：带引号的标识符
- 功能值：0 = 不支持；1 = 支持

对于没有为此功能提供值的 DirectConnect，功能缺省为 0。

分隔标识符支持

带括号的标识符的行为与带引号的标识符的行为相同，不同之处在于不需要借助 set quoted_identifier on 来使用带括号的标识符。

auto identity 选项

当启用 Adaptive Server 的 auto identity 数据库选项时，将 IDENTITY 列添加到在数据库中创建的所有表中。对于代理表，列名为 CIS_IDENTITY_COL；对于本地表，列名为 SYB_IDENTITY_COL。在这两种情况下，可使用 syb_identity 关键字引用该列。

触发器

组件集成服务允许在代理表上使用触发器；但是它们的用途很有限。可以在代理表上使用 create 创建触发器，并按照常规 Adaptive Server 表的方式调用该触发器。但是， image 之前和之后的数据不写入代理表的日志，因为 insert、 update 和 delete 命令将传递到远程服务器。实际上进入日志的视图 inserted 或 deleted 表，不包含代理表的数据。用户无法检查正在插入、删除或更新的行，因此具有代理表的触发器具有有限的值。

Adaptive Server 15.0 版及更高版本不支持使用触发器更新的函数。

附 录 A

教程

本章提供关于设置组件集成服务和访问远程服务器的教程。

注释 此教程假定已安装 pubs2 数据库。

组件集成服务入门指南

本节为配置服务器以访问远程数据源提供了逐步的指导。它包括有关下述内容的说明：

- 添加远程服务器
- 将远程对象映射到本地代理表
- 执行远程表之间的连接

Adaptive Server 文档中阐述了日常系统管理任务（如启动和停止 Adaptive Server、创建登录、创建组、添加用户、授予权限和口令管理）。

添加远程服务器

可用服务器访问远程服务器上的数据。操作之前，必须配置组件集成服务。

概述

1 将远程服务器添加到 interfaces 文件中。

2 将远程服务器的名称、服务器类和网络名添加到系统表中。

3 还可以指派替代登录名和口令。

将远程服务器添加到 interfaces 文件中

使用 dsedit 或 dscp 实用程序编辑 *$SYBASE* 目录下的 interfaces 文件：

- 在 UNIX 中， interfaces 文件称为 *interfaces*。
- 在 Windows 中， interfaces 文件称为 *sql.ini*。

关于 interfaces 文件的详细讨论，请参见所用平台的《Adaptive Server 配置指南》。

在系统表中创建服务器条目

使用 sp_addserver 向 sysservers 表中添加条目。 sp_addserver 为本地服务器创建条目并为要调用的每个远程服务器创建一个条目。 sp_addserver 语法为：

```
sp_addserver server_name [,server_class [,network_name]]
```

其中：

- *server_name* 是用于标识服务器的名称。它必须是唯一的。
- *server_class* 是受支持的服务器类之一。缺省值为 ASEnterprise。如果将 server_class 设置为 local，则忽略 *network_name*。
- *network_name* 是 interfaces 文件中的服务器名。此名称可以与 *server_name* 相同，也可以不同。 *network_name* 有时称为*物理名*。

示例

下面的示例为名为 SYBASE 的本地服务器和具有服务器类 ASEnterprise 的远程服务器 CTOSDEMO 创建条目。

```
sp_addserver SYBASE, local
sp_addserver CTOSDEMO, ASEnterprise, CTOSDEMO
```

添加本地服务器后，您必须重新启动 Adaptive Server。

添加替代登录名和口令

使用 sp_addexternlogin 指派在与远程服务器通信时要使用的替代登录名和口令。此步骤是可选的。 sp_addexternlogin 的语法为：

```
sp_addexternlogin remote_server, login_name, remote_name [,
remote_password]
```

其中：

- *remote_server* 是远程服务器的名称。 *remote_server* 必须通过 master.dbo.sysservers 表中的条目为本地服务器所知。
- *login_name* 是本地服务器所知的帐户。必须用 master.dbo.syslogins 表中的条目来表示 *login_name*。只有使用 “sa” 帐户、 “sso” 帐户和 *login_name* 帐户的用户有权修改给定本地用户的远程访问权限。
- *remote_name* 是 *remote_server* 所知的帐号，它必须是运行 *remote_server* 的节点上的有效帐号。这是用于登录到 *remote_server* 的帐号。
- *remote_password* 是 *remote_name*. 的口令。

示例

```
sp_addexternlogin FRED, sa, system, sys_pass
```

允许本地服务器使用远程名 “system” 和远程口令 “sys_pass”，代表用户 “sa” 访问远程服务器 FRED。

```
sp_addexternlogin OMNI1012, bobj, jordan, hitchpost
```

通知本地服务器当登录名 “bobj” 登录时，通过远程名“jordan” 和远程口令 “hitchpost” 来访问远程服务器 OMNI1012。仅 “bobj” 帐户、 “sa” 帐户和 “sso” 帐户有权添加或修改登录名 “bobj” 的远程登录信息。

检验连接性

使用 connect to *server_name* 命令检验配置是否正确。connect to 要求 “sa” 为非 “sa” 用户显式授予连接权限。 connect to 命令建立到远程服务器的直通模式连接。除非发出 disconnect 命令，否则此直通模式将一直有效。

两个远程表之间的连接

通过组件集成服务，可以执行远程表之间的连接。

将远程服务器添加到 interfaces 文件中

使用 dsedit 编辑 interfaces 文件。

定义远程服务器

使用 sp_addserver 将条目添加到 sysservers 系统表中。在发起调用的服务器上，每个要调用的远程服务器都必须有一个相应的条目。

sp_addserver 语法为：

sp_addserver *server_name [,server_class] [,network_name]*

其中：

- *server_name* 是用于标识服务器的名称。它必须是唯一的。
- *server_class* 是受支持的服务器类之一。缺省值是 sql_server。如果值是 local，则忽略 *network_name*。
- *network_name* 是 interfaces 文件中的服务器名。该名称可能与所指定的 *server_name* 相同，也可能不同。如果未提供 *network_name* ，则缺省值为 *server_name*。

示例

下面的示例为名为 SYBASE 的本地服务器和类 ASEnterprise 的远程服务器 SYBASE 创建条目。

```
sp_addserver SYBASE, local
sp_addserver CTOSDEMO, ASEnterprise, SYBASE
```

将远程表映射到 Adaptive Server

create existing table 用于定义现有 （代理）表。此选项的语法与 create table 命令类似，如下所示：

```
create proxy_table
    table_name
    at “pathname”
```

服务器处理此命令时，不创建新表。不过，它检查表映射和检验基础表的存在。如果对象不存在 （主机数据文件或远程服务器对象），则服务器拒绝该命令并给客户端返回一条错误信息。

当定义现有表后，对该表发出 update statistics 命令。这将帮助查询优化程序在考虑选择索引和连接顺序时做出正确的选择。

示例

要说明示例 pubs2 数据库中映射到本地服务器的远程 Adaptive Server 表 publishers 和 titles，请执行以下步骤：

映射远程表

生成上述映射所需的步骤如下：

1　定义名为 SYBASE 的服务器。它的服务器类为 ASEnterprise，它在 interfaces 文件中的名称为 SYBASE：

```
exec sp_addserver SYBASE, ASEnterprise, SYBASE
```

2　定义远程登录别名。此步骤是可选的。用户 “sa” 在远程服务器 SYBASE 上使用用户名 “sa” 和口令 “timothy” 标识身份：

```
exec sp_addexternlogin SYBASE, sa, sa, timothy
```

3　定义远程 publishers 表：

```
create proxy_table publishers
at "SYBASE.pubs2.dbo.publishers"
```

4　定义远程 titles 表：

```
create proxy_table titles
at "SYBASE.pubs2.dbo.titles"
```

执行连接

使用 select 语句执行连接。

```
select Publisher = p.pub_name, Title = t.title
 from publishers p, titles t
 where p.pub_id = t.pub_id
 order by p.pub_name
```

附 录 B

故障排除

对使用组件集成服务时可能遇到的问题，本附录提供了故障排除提示。本章的目的是：

- 针对某些错误情况提供足够的信息，以使您在没有技术支持部门帮助的情况下也能解决问题
- 提供在与技术支持部门联系之前需收集的信息的列表，这有助于迅速解决问题
- 使您更好地了解组件集成服务

还应使用 《故障排除和错误消息指南》进行故障排除。本附录提供组件集成服务常见问题的故障排除提示，并随联机恢复过程列出了所有错误消息；而 《故障排除和错误消息指南》提供有关 Adaptive Server 问题的提示，这些问题并不针对于组件集成服务。

访问组件集成服务时出现问题

如果发出了一个访问远程对象的命令并且没有找到组件集成服务，则会出现以下错误消息：

```
cis extension not enabled or installed
```

- 检验是否将 enable cis 配置参数设置为 1，方法是运行：

```
sp_configure "enable cis"
```

sp_configure 为 enablecis 参数返回以下行：

```
name           min  max   config value   run value
enable cis     0    1     1              1
```

“config value” 和 “run value” 都应该为 1。如果这两个值都为 0，则将 enable cis 配置参数设置为 1，然后重新启动服务器。使用以下语法：

```
sp_configure "enable cis" 1
```

如果 “config value” 为 1，并且 “run value” 为 0，则说明已设置了 enable cis 配置参数，但是该参数只有在重新启动服务器后才能生效。

注释 从 Adaptive Server 12.0 版开始，缺省情况下将启用组件集成服务。

使用组件集成服务时出现问题

本节提供关于如何解决使用组件集成服务时可能遇到的问题的提示。

无法访问远程服务器

无法访问远程服务器时，会返回以下错误消息：

```
11206 Unable to connect to server server_name.
```

此消息之前将出现以下 Client-Library 消息之一：

```
Requested server name not found
Driver call to connect two endpoints failed
Login failed
```

Client-Library 消息表示不能访问远程服务器的原因，如以下几节所述。

未找到请求的服务器名

在 interfaces 文件中未定义此服务器，这时将显示以下消息：

```
Requested server name not found
11206 Unable to connect to server server_name.
```

使用 sp_addserver 添加远程服务器时，未检查 interfaces 文件。第一次试图与远程服务器连接时，将检查 interfaces 文件。若要更正这一问题，请将远程服务器添加到正由组件集成服务使用的 interfaces 文件中。

连接两个结束点的驱动程序调用失败

如果在 interfaces 文件中定义了远程服务器，但是未收到连接请求的响应，则显示以下消息：

```
Driver call to connect two endpoints failed
11206 Unable to connect to server server_name.
```

- 检验环境设置是否正确。

  若要对其进行测试，请尝试使用 isql 或类似的工具直接连接到远程服务器：

  a 登录到运行组件集成服务的计算机上。

  b 将 SYBASE 环境变量设置为在组件集成服务启动时所使用的相同位置。组件集成服务使用 SYBASE 环境变量所指定的目录中的 interfaces 文件，除非在 *runserver* 文件中用 -i 参数覆盖了该环境变量。

  注释 前两步对于确保测试环境与连接远程服务器失败时组件集成服务所使用的环境保持相同至关重要。

  c 使用 isql 或类似的工具直接连接到远程服务器。

  如果环境设置正确，但连接失败，则按列表中的步骤继续操作。如果已建立连接，则组件集成服务所使用的环境存在问题。

- 检验远程服务器是否已启动并且正在运行。

  登录到远程服务器所在的计算机上，检验服务器是否正在运行。如果服务器正在运行，继续按列表操作。如果服务器未运行，则重新启动服务器，然后重试查询。

- 检验 interfaces 文件中的远程服务器条目是否正确：
  - 检验此计算机名称是否是已装载软件的计算机的正确名称。
  - 检验如果 interfaces 文件为文本文件，查询行和 master 行以制表符开头，而不是以空格开头。
  - 检验端口号是否可用。检查 */etc* 目录下的 *services* 文件，确保没有为另一进程保留端口号。

登录失败

如果能访问远程服务器，但是登录名和口令不正确，则显示以下消息：

```
Login failed
11206 Unable to connect to server server_name.
```

通过执行下列操作，检查是否为远程服务器定义了外部登录名：

```
exec sp_helpexternlogin server_name
```

如果未定义外部登录名，组件集成服务使用用于连接到 Adaptive Server 的用户登录名和口令。例如，如果用户使用 “sa” 帐户连接到 Adaptive Server，那么组件集成服务在建立远程连接时将使用登录名 “sa”。除非远程服务器是另一个 Adaptive Server，否则 “sa” 帐户可能不存在，而且必须使用 sp_addexternlogin 添加外部登录名。

如果定义了外部登录名，则检验用户登录名是否正确。远程服务器登录区分大小写。正在使用的用户登录名和 *externlogins* 中的条目的大小写是否正确？

如果登录名正确，则口令可能是错误的。您不能显示口令。如果用户登录名错误或者口令不正确，则删除现有外部登录名，然后执行下列命令对其重新定义：

```
exec sp_dropexternlogin server_name, login_name
go
 exec sp_addexternlogin server_name, login_name,
remote_login, remote_password
go
```

无法访问远程对象

无法访问远程对象时，将出现以下错误消息：

```
Error 11214  Remote object object does not exist.
```

本地代理表定义或者远程服务器上的表本身可能有问题。

请检验：

- 在组件集成服务中是否已定义了此对象。

  运行以下命令进行确认：

  ```
  sp_help object_name
  ```

  如果该对象不存在，请在组件集成服务中创建该对象。

- 如果已在组件集成服务中定义了该对象，其定义是否正确无误。

  表名可以有四个部分，格式为 *server.dbname.owner.tablename*。*dbname* 部分对于 Oracle 或 InfoHUB 服务器是无效的。

  如果对象定义错误，请用 sp_dropobjectdef 命令将其删除，然后用 sp_addobjectdef 命令进行正确定义。

- 如果本地对象定义正确，则检查远程服务器上的表以验证：
  - 是否将权限设置为既可访问数据库又可访问表。
  - 是否已将数据库标记为可疑。
  - 数据库是否可用。
  - 您是否可以使用本机工具 （如 Oracle 上的 SQL*Plus）访问远程表。

从远程对象检索数据时出现问题

收到关于远程对象不匹配的错误消息时，说明组件集成服务对象定义与远程对象定义不匹配。发生这种情况的原因是：

- 在组件集成服务之外变更了对象定义。
- 在组件集成服务之外添加或删除了索引。

在组件集成服务之外变更了对象

在组件集成服务中定义对象后，在远程服务器对该对象所做的变更不会改变本地代理对象定义。如果在组件集成服务之外变更了对象，解决这个问题的步骤将会有所不同，这取决于定义对象时使用的是 create existing table 还是 create table。

若要确定定义对象时所采用的方法，请运行：

```
sp_help object_name
```

如果对象是通过 create existing table 命令定义的，则在结果集中返回以下消息：

```
Object created with 'existing' option
```

如果未显示此消息，则说明对象是通过 create table 定义的。

如果在组件集成服务中使用 create existing table 创建表，则：

1 在组件集成服务中使用 drop table。

2 在组件集成服务中使用 create existing table 再次创建该表。这样就会在远程服务器上用表的新版本创建该表。

如果该表是在组件集成服务中使用 create table 创建的，则在使用 drop table 时，将删除远程对象。若要防止出现此情况，请执行以下步骤：

1 对远程服务器上的表重命名，这样在使用 drop table 时就不会将此表删除。

2 使用原始名称在远程服务器上创建一个表。

3 在组件集成服务中使用 drop table 删除组件集成服务和远程服务器上的表。

4 在远程服务器上用原始名对第 1 步中保存的表重命名。

5 在组件集成服务中使用 create existing table 再次创建该表。

警告！在远程服务器上对表进行重命名之前，切勿在组件集成服务中使用 drop table，否则将删除远程服务器上的表。

一个应遵循的合理规则是：在远程服务器上创建对象，然后在组件集成服务中执行 create existing table 以创建对象。这可用较少的步骤来解决不匹配问题，并且不会删除远程服务器上的对象。

在组件集成服务之外添加或删除了索引

组件集成服务无法识别在组件集成服务之外添加或删除的索引。检验组件集成服务使用的索引是否与远程服务器上使用的索引相同。使用 sp_help 查看组件集成服务所用的索引。在远程服务器上使用适当的命令检验远程服务器所用的索引。

如果索引不相同，则解决这个问题的步骤会有所不同，这取决于定义对象时使用的是 create existing table 还是 create table。

若要确定定义对象时所采用的方法，请运行：

```
sp_help object_name
```

如果对象是通过 create existing table 命令定义的，则在结果集中返回以下消息：

```
Object created with 'existing' option
```

如果未显示此消息，则说明对象是通过 create table 定义的。

如果使用 create existing table 创建对象，则：

1 在组件集成服务中使用 drop table。

2 在组件集成服务中使用 create existing table 命令重新创建表。这将更新索引以便与远程表中的索引匹配。

如果使用 create table 创建对象：

1 使用 drop index 从远程表中删除索引。

2 在组件集成服务中使用 create index 命令重新创建索引。这就在组件集成服务和远程服务器中创建了索引。

如果对象是用 create table 定义的，一种替代的方法是打开跟踪标志 11208。该跟踪标志能防止将 create index 传送到远程服务器。若要使用跟踪标志 11208，请按以下步骤操作：

1 打开跟踪标志 11208：

```
dbcc traceon(11208)
```

2 用 create index 创建索引。

3 关闭跟踪标志 11208：

```
dbcc traceoff(11208)
```

如果需要帮助

如果遇到无法使用手册解决的问题，可请现场的指定人员与 Sybase 技术支持部门联系。在与技术支持部门联系之前收集以下信息，有助于迅速解决问题。

- 如果是在试图访问远程数据时出现问题，则对本地表执行同一脚本。如果本地表没有问题，则说明问题出在组件集成服务上，应该按此列表继续操作。
- 查看所用的 Adaptive Server 的版本：

  ```
  select @@version
  ```

- 记下可再现问题的 SQL 脚本。包括用于创建表的脚本。
- 查找查询的处理计划。该计划是使用 set showplan 生成的。这样的一个示例如下：

  ```
  set showplan, noexec on
   go
   select au_lname, au_fname from authors
   where au_id = 'A1374065371'
   go
  ```

  此查询的输出如下：

```
set showplan, noexec on
go
select au_lname, au_fname from authors where au_id = 'A1374065371'
 go
The Abstract Plan (AP) of the final query execution plan:
( remote_sql )
To experiment with the optimizer behavior, this AP can be modified and
then
passed to the optimizer using the PLAN clause:
SELECT/INSERT/DELETE/UPDATE ...
PLAN '( ... )

QUERY PLAN FOR STATEMENT 1 (at line 1).

1 operator(s) under root


The type of query is SELECT.

ROOT:EMIT Operator

   |LE_REMSCANOP Operator
```

```
   |    SELECT "au_lname" , "au_fname" FROM pubs2.dbo."authors"
WHERE "au_
   |    id" = 'A1374065371'
```

noexec 选项可编译查询，但不执行查询。在关闭 noexec 命令前，不执行任何后续命令。

- 执行查询时打开跟踪标志 11201 至 11205，可获得事件记录。这些跟踪标志记录如下内容：
  - 11201 — 客户端连接、断开连接和关注事件。
  - 11202 — 客户端语言、游标声明、动态准备和动态即时执行文本。
  - 11203 — 客户端 RPC 事件。
  - 11204 — 路由到客户端的消息。
  - 11205 — 与远程服务器的交互。
  - 11206 — 记录文件和目录处理步骤。
  - 11207 — 记录 text 和 image 的处理信息。

  在打开跟踪标志的情况下执行脚本后，可在 *$SYBASE/install* 目录下的错误日志中找到记录内容。例如：

```
dbcc traceon (11201,11202,11203,11204,11205)
 go
 select au_lname, au_fname from authors
 where au_id = 'A1374065371'
 go
 dbcc traceoff (11201,11202,11203,11204,11205)
 go
```

  错误日志输出如下 （为提高可读性，删除了在每一条目起始处输出的时间戳）：

```
server  TDS_LANG, spid 15: command text:
select au_lname, au_fname from authors where au_id = 'A1374065371'

server  RemoteAccess constructed
server  EXECLANG, spid 15, server huntington0_19442, quickpass statement:

ELECT "au_lname" , "au_fname" FROM pubs2.dbo."authors" WHERE "au_id" =
'A1374065371'

server  BINDCOLS, spid 15: column 1, name au_lname, fmt.type 'CHAR',
fmt.maxlen 40, fmt.stat 16, con.type 'VARCHAR', con.maxlen 40
```

```
server  BINDCOLS, spid 15: column 2, name au_fname, fmt.type 'CHAR',
fmt.maxlen 20, fmt.stat 16, con.type 'VARCHAR', con.maxlen 20
server  BINDCOLS, spid 15: bind array size 50, total memory required is
4304 bytes

server  FETCH   , spid 15: cursor C1; ct_fetch() returned 0 rows; status
-204

server RemoteAccess deleted
```

这种跟踪是全局性的，所以将跟踪标志打开后，将记录所执行的任何查询；因此在获得日志后，将跟踪标志关闭。此外，通过关闭服务器、重命名错误日志、然后重新启动服务器来定期清除错误日志。这样即可创建新的错误日志。

索引

英文

A

B

C

D

F

G

H

J

K

L

M

N

P

Q

R

S

T

W

X

Y

Z