リファレンスマニュアル

SAP® Replication Server 15.7.1 SP200

ドキュメント ID： DC37518-01-1571200-01
改訂： 2014 年 3 月
Copyright©2014 by SAP AG or an SAP affiliate company. All rights reserved.
このマニュアルの内容を SAP AG の明示的許可を得ずに、いかなる手段によっても、複製、転載することを禁じます。 ここに記載された情報は事前の通知なしに変更されることがあります。
SAP AG およびディストリビュータが販売しているソフトウェア製品には、他のソフトウェアベンダー独自のソフトウェアコンポーネントが含まれているものがあります。 国内製品の仕様は変わることがあります。
これらの資料は SAP AG および関連会社 (SAP グループ) が情報のみを目的として提供するものであり、いかなる種類の表明または保証も行うものではなく、SAP グループはこの資料に関する誤りまたは脱落について責任を負わないものとします。 SAP グループの製品およびサービスに関する保証は、かかる製品およびサービスに付属している明確な保証文書がある場合、そこで明記されている保証に限定されます。 ここに記載されているいかなる内容も、追加保証を構成するものとして解釈されるものではありません。
ここに記載された SAP および他の SAP 製品とサービス、ならびに対応するロゴは、ドイツおよび他の国における SAP AG の商標または登録商標です。 その他の商標に関する情報および通知については、http://www.sap.com/corporate-en/legal/copyright/index.epx#trademark を参照してください。

目次

表記の規則

ここでは、SAP® マニュアルで使用しているスタイルおよび構文の表記規則について説明します。

表記の規則

構文要素	定義
等幅 （固定幅）	• SQL およびプログラムコード • 表示されたとおりに入力する必要のあるコマンド • ファイル名 • ディレクトリ名
斜体等幅	SQL またはプログラムコードのスニペット内では、ユーザ指定の値のプレースホルダ (以下の例を参照)
斜体	• ファイルおよび変数の名前 • 他のトピックまたはマニュアルとの相互参照 • 本文中では、ユーザ指定の値のプレースホルダ (以下の例を参照) • 用語解説に含まれているテキスト内の用語
太字体 sans-serif	• コマンド、関数、ストアドプロシージャ、ユーティリティ、クラス、メソッドの名前 • 用語解説のエントリ (用語解説内) • メニューオプションのパス • 番号付きの作業または手順内では、クリックの対象となるボタン、チェックボックス、アイコンなどのユーザインタフェース (UI) 要素

必要に応じて、プレースホルダ (システムまたは設定固有の値) の説明が本文中に追加されます。次に例を示します。

次のコマンドを実行します。

```
installation directory/start.bat
```

installation directory はアプリケーションがインストールされた場所です。

構文の表記規則

構文要素	定義
{ }	中カッコで囲まれたオプションの中から必ず 1 つ以上を選択する。コマンドには中カッコは入力しない。
[ ]	角カッコは、オプションを選択しても省略してもよいことを意味する。コマンドには角カッコは入力しない。
( )	このカッコはコマンドの一部として入力する。
|	縦線はオプションのうち 1 つのみを選択できることを意味する。
,	カンマは、表示されているオプションを必要な数だけ選択でき、選択したものをコマンドの一部として入力するときにカンマで区切ることを意味する。
...	省略記号 (...) は、直前の要素を必要な回数だけ繰り返し指定できることを意味する。省略記号はコマンドには入力しない。

大文字と小文字の区別

- すべてのコマンド構文およびコマンドの例は、小文字で表記しています。ただし、複写コマンド名では、大文字と小文字が区別されません。たとえば、**RA_CONFIG**、**Ra_Config**、**ra_config** は、すべて同じです。
- 設定パラメータの名前では、大文字と小文字が区別されます。たとえば、**Scan_Sleep_Max** は、**scan_sleep_max** とは異なり、パラメータ名としては無効になります。
- データベースオブジェクト名は、複写コマンド内では、大文字と小文字が区別されません。ただし、複写コマンドで大文字と小文字が混在したオブジェクト名を使用する場合 (プライマリデータベースの大文字と小文字が混在したオブジェクト名と一致させる場合)、引用符でオブジェクト名を区切ります。次に例を示します。**pdb_get_tables** "*TableName*"
- 識別子および文字データでは、使用しているソート順によっては大文字と小文字が区別されます。
  - “binary” などの大文字と小文字を区別するソート順を使用する場合には、識別子や文字データは、大文字と小文字を正しく入力してください。
  - “nocase” などの大文字と小文字を区別しないソート順を使用する場合には、識別子や文字データは、大文字と小文字をどのような組み合わせでも入力できます。

用語

SAP® Replication Server® はさまざまなコンポーネントと連携して、SAP Adaptive Server Enterprise (SAP ASE)、SAP HANA® データベース、SAP® IQ、Oracle、IBM

DB2 UDB、Microsoft SQL Server など、サポートされているデータベース間の複製を実現します。 SAP Replication Server では SAP ASE を Replication Server システムデータベース (RSSD) に使用します。または、SAP® SQL Anywhere® を Embedded Replication Server システムデータベース (ERSSD) に使用します。

Replication Agent™ は、SAP ASE、SAP HANA データベース、Oracle、IBM DB2 UDB、Microsoft SQL Server 用の Replication Agent を表現するために使用される一般的な用語です。 具体的な名前は、次のとおりです。

- RepAgent - SAP ASE 用の Replication Agent スレッド
- Replication Agent for Oracle
- Replication Agent for Microsoft SQL Server
- Replication Agent for UDB — Linux、Unix、Windows 用の IBM DB2
- Replication Agent for DB2 for z/OS

複写コマンド言語

各カテゴリのコマンドについて説明します。複数のカテゴリに含まれているコマンドもあります。

コマンド構文と使用法の詳細については、『SAP Replication Server コマンド』を参照してください。

Replication Command Language (RCL) を使用するときには、以下のフォーマット規則に従ってください。

- 行にコマンドを入力するときは、キーワードや識別子の途中を除いてどの位置でも改行できます。
- 1 つの文字列を次の行に続けるときには、行の終わりに円記号 (¥) を入力します。行内の余分な空白文字は、円記号の後ろ以外は無視されます。
- 円記号のあとには、空白文字を入力しないでください。特に指示がないかぎり、バッチには複数のコマンドを入力できます。
- RCL コマンドはトランザクション指向ではありません。 SAP Replication Server は、他のコマンドの完了ステータスには関係なく、バッチ内のコマンドを 1 つずつ実行します。 ただし、1 つのコマンドに構文エラーが存在すると、バッチ内のその後にあるコマンドは SAP Replication Server で解析されなくなります。

データ型、識別子、予約語、および SAP ASE でのサポートの詳細については、「トピック」を参照してください。

SAP Replication Server アーキテクチャについては、『管理ガイド 第 1 巻』の「Replication Server の技術的概要」を参照してください。

一部の Replication Server プロシージャでは、**sp_setreptable** や **sp_setrepproc** などの SAP ASE システムプロシージャを実行する必要があります。 構文と使用法の詳細については、「SAP ASE コマンドとシステムプロシージャ」を参照してください。

データ複写コマンド

データ複写コマンドは、テーブルまたはストアドプロシージャを複写するための、複写定義、パブリケーション、サブスクリプションを作成し管理します。

テーブル複写定義コマンド

テーブル複写定義には、複写するテーブルとそのカラムのリストを記述します。プライマリテーブルが複写の送信元で、レプリケートテーブルが送信先です。各プライマリテーブルにつき、1 つまたは複数の複写定義を作成できます。

複写定義は、プライマリテーブルが格納されているデータベースを管理する Replication Server で作成してください。

複写定義には、次の内容を含めます。

- 複写定義の名前
- プライマリテーブルおよびレプリケートテーブルの名前 (両者が互いに異なり、複写定義名も異なる場合)
- プライマリテーブルのロケーション
- 複写するプライマリカラムの名前とそのデータ型、および対応するレプリケートカラムの名前
- テーブルのプライマリキーとなるカラムの名前

複写定義には、任意で次の内容を含めることができます。

- サブスクリプションの **where** 句で参照できるカラムの名前
- 複写定義とそのカラムを、スタンバイデータベースへの複写で使用するかどうか
- **update** および **delete** オペレーションで、すべてのカラムを複写するか、必要な最小限のカラムを複写するか
- *text*、*unitext*、*image*、*rawobject* カラムの複写ステータス
- 複写した値のデータ型を、プライマリデータベースのデータ型からレプリケートデータベースのデータ型に変更するかどうか

複写定義を作成するだけでは、データは分配されません。分配するには、個々のレプリケートデータベースでテーブルのコピーを作成してから、サブスクリプションを作成してデータの複写を開始してください。

テーブル複写定義に関連するコマンドには、次のものがあります。

- **create replication definition** - テーブルの複写定義を作成する。
- **alter replication definition** - 複写定義を変更する。
- **drop replication definition** - 複写定義を削除します。

参照：

- サブスクリプションコマンド (9 ページ)

ファンクション複写定義コマンド

ファンクション複写定義には、複写するストアドプロシージャについての情報を指定します。

ファンクション複写定義は、プライマリデータベースを管理する Replication Server で作成してください。

ファンクション複写定義には次の内容を含めます。

- ファンクション複写定義の名前。
- プライマリデータのロケーション。
- 複写するストアドプロシージャパラメータの名前とデータ型。

ファンクション複写定義にはオプションで次の内容を含めることができます。

- 送信元データベースで実行するストアドプロシージャの名前と送信先データベースで実行するストアドプロシージャの名前 (ストアドプロシージャ名がファンクション複写定義の名前と異なる場合)。
- サブスクリプションの **where** 句で参照できるパラメータの名前。
- ファンクション複写定義とそのパラメータを、スタンバイデータベースへの複写で使用するかどうか。

ファンクション複写定義に関連するコマンドには、次のものがあります。

- **create applied function replication definition** - ストアドプロシージャ用の適用ファンクション複写定義を作成します。
- **alter applied function replication definition** - 適用ファンクション複写定義を変更します。
- **create request function replication definition** - ストアドプロシージャ用の要求ファンクション複写定義を作成します。
- **alter request function replication definition** - 要求ファンクション複写定義を変更します。
- **drop function replication definition** - ファンクション複写定義を削除します。

ファンクション複写定義を作成するだけでは、データは分配されません。プライマリデータベースとレプリケートデータベースの両方でストアドプロシージャを作成してから、レプリケート Replication Server でサブスクリプションを作成する必要があります。

複写定義のサブスクリプションの作成に使用するコマンドについては、「サブスクリプションコマンド」を参照してください。

データベース複写定義コマンド

データベース複写定義には、複写するデータベースまたはデータベースオブジェクトを記述します。データベース全体を複写するか、そのデータベース内の特定

のテーブル、ファンクション、トランザクション、DDL、システムストアドプロシージャなどを複写するか複写しないかを選択できます。

データベース複写定義には、次の内容を含めます。

- データベース複写定義の名前
- 複写するデータベースが置かれてるプライマリサーバの名前
- 複写するデータベースの名前

データベース複写定義は、オプションで次の内容を含めることができます。

- サブスクリプションを作成しているデータベースに DDL を複写するかどうかを示すインジケータ
- サブスクリプションを作成しているデータベースにテーブル、ストアドプロシージャ、ユーザ定義ファンクション、トランザクション、またはシステムプロシージャを複写するかどうかを示すインジケータ

データベース複写定義に関連するコマンドには、次のものがあります。

- **create database replication definition** - データベースまたはデータベースオブジェクトを複写するための複写定義を作成します。
- **alter database replication definition** - 既存のデータベース複写定義を変更します。
- **drop database replication definition** - 既存のデータベース複写定義を削除します。

パブリケーションコマンド

Replication Server のパブリケーション機能を使用すると、サブスクリプションを作成するテーブルとプロシージャおよびそれらの複写定義を 1 つのグループにまとめ、そのグループに対して 1 つのサブスクリプションを作成できます。

「*パブリケーション*」は、同じプライマリデータベースからのアーティクルの集まりです。各「*アーティクル*」はテーブルまたはストアドプロシージャと、どのローが処理対象であるかを指定した一連の **where** 句の複写定義です。1 つのアーティクルに、ゼロ、1 個、または複数の **where** 句を含めることができます。複数指定する場合には、句と句の間を **or** キーワードで区切ります。

パブリケーションとアーティクルに関連するコマンドには、次のものがあります。

- **create publication** - パブリケーションを作成する。
- **drop publication** - パブリケーションとそのアーティクルを削除する。*drop_repdef* オプションを指定すると、関連する複写定義も削除される。
- **validate publication** - パブリケーションに少なくとも 1 つのアーティクルが含まれていることを確認し、そのパブリケーションにマークを付けて、新しいサブスクリプションを作成できるようにする。
- **check publication** - パブリケーションにサブスクリプションを作成できるかどうかを示し、含まれるアーティクルの数をレポートする。

- **create article** - アーティクルを作成し、パブリケーションに割り当てる。
- **drop article** - パブリケーションからアーティクルを削除する。*drop_repdef* オプションは、関連する複写定義も削除する。

参照：
- パブリケーションサブスクリプションコマンド (12 ページ)

サブスクリプションコマンド

サブスクリプションは、データまたはストアドプロシージャの複写を開始します。サブスクリプションには、テーブル複写定義かファンクション複写定義の名前、またはパブリケーション、およびデータの複写先であるデータベースを指定します。

- テーブル複写定義のサブスクリプションは、データを複写する。
- ファンクション複写定義のサブスクリプションは、ストアドプロシージャを複写する。
- データベース複写定義のサブスクリプションは、データベースまたはデータベースオブジェクトを複写する。
- パブリケーションのサブスクリプションは、パブリケーション内の各アーティクルによって表されるデータを複写する。パブリケーションには、ストアドプロシージャのアーティクルも含めることができる。

テーブル複写定義またはファンクション複写定義のサブスクリプションには、複写するローや、ストアドプロシージャを複写するかどうかを決定する **where** 句を含めることができます。

データベース複写定義へのサブスクリプションは、すべてのデータに対してサブスクリプションを作成します。**where** 句を使用して、サブスクリプションを作成するデータの条件を設定することはできません。特定のテーブルまたはファンクションに対してサブスクリプションを作成する必要がある場合は、テーブルサブスクリプションまたはファンクションサブスクリプションを追加できます。『Replication Server 管理ガイド 第 1 巻』の「MSA を使用した複写オブジェクトの管理」の「データベースサブスクリプション、テーブルサブスクリプション、ファンクションサブスクリプションの併用」を参照してください。

注意： パブリケーションのサブスクリプションに **where** 句を含めることはできません。**where** 句はパブリケーションのアーティクルに含まれます。

参照：
- データベース複写定義コマンド (7 ページ)

サブスクリプションマテリアライゼーション

テーブル複写定義のサブスクリプションを作成すると、そのサブスクリプションに該当するローが、「*マテリアライゼーション*」と呼ばれる処理の中でプライマ

リテーブルからレプリケートテーブルにコピーされます。マテリアライゼーションが完了した後、Replication Server の通常の複写によってプライマリデータベースにローの変更が分配されます。

1 つのサブスクリプションがたくさんのローに関係している場合、マテリアライゼーションで長時間ロックがホールドされて、ネットワークが過負荷状態になる可能性があります。また、Replication Server のキューも、データで満杯になる恐れがあります。この問題を避けるために、Replication Server にはサブスクリプションをマテリアライズする 4 つの方法があります。

テーブル複写定義またはパブリケーションのサブスクリプションには、すべての方法を使用できます。ファンクション複写定義やデータベース複写定義のサブスクリプションには、非マテリアライゼーションまたはバルクマテリアライゼーションを使用してください。

- 「*アトミックマテリアライゼーション*」は、テーブル複写定義用のデフォルトの方法です。ホールドロックを使ってプライマリテーブルでローが選択され、ネットワーク経由でコピーされます。マテリアライゼーション中はプライマリテーブルはロックされ、プライマリテーブルとレプリケートテーブルの間のデータの一貫性は保たれます。
- 「*ノンアトミックマテリアライゼーション*」では、ホールドロックを使わずにプライマリテーブルでローが選択され、ネットワーク経由でコピーされます。プライマリテーブルはロックされないため、マテリアライゼーションの処理中に、当初は存在しなかった複写がプライマリテーブルで行われる可能性があります。
- 「非マテリアライゼーション」では、プライマリデータとレプリケートデータは、すでに同期しています。ネットワーク経由でデータをコピーしたり、メディアからデータをロードする必要はありません。このようなサブスクリプションが作成されている間は、更新を行うことができません。
- 「*バルクマテリアライゼーション*」は、メディアから手動でデータをロードまたはアンロードする方法です。大量のデータを対象とするサブスクリプションをマテリアライズするには、この方法が一番適しています。

サブスクリプションマテリアライゼーションメソッドの詳細については、『Replication Server 管理ガイド 第 1 巻』を参照してください。

アトミックおよびノンアトミックマテリアライゼーションコマンド

サブスクリプションを作成してレプリケートデータベースでデータの初期化を行うには、次のコマンドを使用します。

- **create subscription -** アトミックマテリアライゼーションを使用して、サブスクリプションの作成およびマテリアライズを行う。
- **create subscription ... without holdlock** - ノンアトミックマテリアライゼーションを使用して、サブスクリプションを作成してマテリアライズします。

ホールドロックなしでプライマリデータを選択するノンアトミックマテリアライゼーションを使用する場合は、次のコマンドも使用する必要があります。

- **set autocorrection** - レプリケートテーブルでローが重複または消滅するエラーを防止します。ホールドロックなしでプライマリデータを選択すると、マテリアライゼーションが完了して通常のトランザクションによる複写が始まる前に、データが更新されることがあります。

非マテリアライゼーションコマンド
レプリケートデータベースでデータがすでに同期している状態でサブスクリプションを作成するには、次のコマンドを使用します。

- **create subscription ... without materialization** - レプリケートデータベースでデータをマテリアライズしないでサブスクリプションを作成します。

バルクマテリアライゼーションコマンド
バルクマテリアライゼーションは、サブスクリプションのステータスを手動で調整し、ファンクション複写定義またはデータベース複写定義のデータを転送するために使用します。

バルクマテリアライゼーションでは、次のコマンドを使用します。

- **define subscription** - プライマリおよびレプリケート Replication Server のシステムテーブルに、サブスクリプションを追加する。
- **activate subscription** - プライマリデータベースからレプリケートデータベースへの更新の配信を開始し、サブスクリプションステータスを ACTIVE に設定します。
  このコマンドを使用してステータスを確認したら、初期データを手動でメディアからレプリケートデータベースにロードしてください。メディアからのロードが完了するまでレプリケートデータベースにデータが適用されないようにするには、**with suspension** オプションを使用します。
- **validate subscription** - バルクマテリアライゼーションを完了して、サブスクリプションステータスを VALID に変更します。マテリアライゼーションの完了が Replication Server に通知されます。

その他のサブスクリプションコマンド

その他のサブスクリプションコマンドについて説明します。

サブスクリプションのマテリアライゼーションまたはマテリアライゼーションの解除をモニタするには、次のコマンドを使用します。

- **check subscription** - プライマリデータベースまたはレプリケートデータベースにあるサブスクリプションのステータスを調べます。

レプリケートデータベースからサブスクリプションを削除するには、次のコマンドを使用します。

- **drop subscription** - システムテーブルからサブスクリプション情報を削除します。

drop subscription with purge を使用して、特定のサブスクリプションに関連したレプリケートデータを削除することもできます。この処理を「*マテリアライゼーション解除*」と呼びます。

パブリケーションサブスクリプションコマンド

パブリケーションサブスクリプションでは、複写定義のサブスクリプションと同じコマンドを使用します。

- アトミックマテリアライゼーション、ノンアトミックマテリアライゼーション、または非マテリアライゼーションを使用してパブリケーションサブスクリプションを作成するには、**create subscription** を使用します。
- バルクマテリアライゼーションを使用してパブリケーションサブスクリプションを作成するには、**define subscription** とその他のバルクマテリアライゼーションコマンドを使用します。

サブスクリプションを持つパブリケーションにアーティクルを追加する場合、新しいアーティクルのサブスクリプションを含めるために、パブリケーションサブスクリプションをリフレッシュする必要があります。この処理を「*リマテリアライゼーション*」と呼びます。

- アトミックまたはノンアトミックリマテリアライゼーションには、**for new articles** 句を指定した **create subscription** を使用する。
- プライマリデータベースとレプリケートデータベースでデータが同期している場合、**for new articles** 句と **without materialization** キーワードを指定した **create subscription** を使用する。
- バルクリマテリアライゼーションには、**for new articles** 句を指定した **define subscription** を使用してから、その他のバルクマテリアライゼーションコマンドを使用します。

参照：

- パブリケーションコマンド (8 ページ)

ユーザコマンド

ユーザが Replication Server のコマンドを実行するには、Replication Server のログインアカウントが必要です。アカウントはログイン名とパスワードで構成され、どちらも Replication Server への接続時に指定します。

ユーザのログインアカウントを管理するには、次のコマンドを使用します。

- **create user** - Replication Server に新規ユーザを追加します。
- **alter user** - ユーザのパスワードを変更します。
- **drop user** - Replication Server のユーザアカウントを削除します。

ユーザのパーミッションを管理するには、次のコマンドを使用します。

- **grant** - パーミッションを付与します。
- **revoke** - パーミッションを取り消します。

パーミッションが異なる別のユーザログインアカウントに切り替えるには、**set proxy** コマンドを使用します。

パーミッションが付与されると、コマンドが実行できるようになります。たとえば、複写定義を作成するには、ユーザが **create object** パーミッションが持っていなければなりません。“sa” パーミッションがあるユーザは、Replication Server のすべてのコマンドを実行できます。

データベースインタフェースコマンド

Replication Server には、データベースに接続したり、データベースで実行されるオペレーションをカスタマイズしたりする方法がいくつか用意されています。

Replication Server のオープンアーキテクチャでは、Adaptive Server およびその他のデータサーバなど、種類の異なる複数のデータサーバでプライマリデータベースまたはレプリケートデータベースを管理できます。

各データベースで次の操作を実行できます。

- データベースへの Replication Server コネクションの作成または修正。「データベースコネクションコマンド」を参照してください。
- エラー処理方法のカスタマイズ。「エラークラスコマンド」を参照してください。
- データベースオペレーションのカスタマイズ。「ファンクションおよびファンクション文字列コマンド」を参照してください。
- ウォームスタンバイアプリケーションで使用されている論理データベースコネクションの作成または変更。「ウォームスタンバイデータベースコマンド」を参照してください。
- コネクションまたは論理コネクションの設定パラメータの設定。「設定コマンド」を参照してください。

複写トランザクションまたはストアドプロシージャの発信元になる各データベースには、それぞれ専用の Replication Agent が必要です。詳細については、『Replication Server 管理ガイド 第 1 巻』を参照してください。

データベースコネクションコマンド

物理データベース接続は、プライマリデータまたはレプリケートデータが格納されているローカルデータベースに SAP Replication Server を接続します。 SAP Replication Server は、コネクションを介してデータベース間でメッセージを分配します。

データベース接続を管理するには、次のコマンドを使用します。

- **create connection** - SAP Replication Server から SAP 以外のデータベースへのデータベース接続を作成します。 SAP ASE へのデータベース接続は、**rs_init** で追加します。
- **create connection using profile** 句 - 事前に定義された情報を使用して、SAP Replication Server と SAP ASE 以外のレプリケートデータベース間のコネクションを設定し、必要に応じて RSSD およびレプリケートデータサーバとデータベースを修正します。
- **alter connection** - データベースコネクションを変更または設定する。
- **drop connection** - データベースコネクションを削除する。
- **suspend connection** - データベースコネクションをサスペンドする。
- **resume connection** - サスペンドされていたコネクションを再開します。

エラークラスコマンド

「*エラークラス*」は、リトライ (**retry**) や無視 (**ignore**) など、データサーバの特定のエラーに割り当てられたエラー処理アクションの名前です。

データベースにエラークラスを関連付けるには、**create connection** コマンドを使用します。エラークラスを変更するには、**alter connection** コマンドを使用します。特定のデータサーバのすべてのデータベースに対して、1 つのエラークラスを作成できます。

注意： **rs_init** を使用してコネクションを追加すると、Adaptive Server データベースのデフォルトのエラークラスが割り当てられます。

これらのコマンドを使用してエラー処理アクションとエラークラスを管理します。

- **create error class** - エラークラスを作成する。
- **alter error class** - エラーアクションを別のエラークラスからコピーして、既存のエラークラスを修正します。
- **move primary** - エラークラスまたはファンクション文字列クラスと、ファンクション文字列クラスの派生クラスを、別のプライマリサイトに転送します。
- **drop error class** - エラークラスを削除する。
- **assign action** - データサーバのエラーコードにアクションを割り当てます。

新しく作成したエラークラスを既存のエラークラスからのエラーアクションで初期化するには、ストアドプロシージャの **rs_init_erroractions** を使用します。詳細については、「Adaptive Server コマンドとシステムプロシージャ」を参照してください。

参照：

- SAP ASE コマンドとシステムプロシージャ (595 ページ)

ファンクションおよびファンクション文字列コマンド

ファンクション文字列を使用して、カスタマイズされたコマンドを送信先データベースで実行するように、Replication Server をプログラムすることができます。

「*ファンクション*」は、データサーバのオペレーションに対応する名前です。たとえば、**rs_insert** はテーブルにローを挿入するシステムファンクションであり、**rs_begin** はトランザクションを開始するシステムファンクションです。システムファンクションは、**rs_insert** のようにデータを操作したり、**rs_begin** のようにトランザクションを制御したりすることができます。

Replication Server では、「*ファンクション文字列*」というテンプレートを使用して、データベースに送信するコマンドが構成されます。ファンクション文字列内の変数は、実行時に、ファンクションからの値と置き換えられます。

「*ファンクション文字列クラス*」は、データベースで使用するファンクション文字列をグループ化したものです。たとえば、ベンダのデータサーバ用のすべてのファンクション文字列や、部署のテーブル用のすべてのファンクション文字列を、ファンクション文字列クラスを使ってグループ化できます。Replication Server には、Adaptive Server データベースおよび DB2 データベース用のファンクション文字列クラスが用意されています。

ファンクション文字列クラスをデータベースに関連付けるには、**create connection** を使用します。ファンクション文字列クラスを変更するには、**alter connection** を使用します。

注意： **rs_init** を使用してコネクションを追加すると、デフォルトのファンクション文字列クラス **rs_sqlserver_function_class** が割り当てられます。

新しいファンクション文字列クラスを作成する場合に、既存のクラスからファンクション文字列を継承できます。そのうえで、使用するデータベースまたはアプリケーションのニーズに応じて、デフォルトと異なる動作が必要なファンクション文字列だけをカスタマイズできます。

ファンクション文字列クラスコマンド

ファンクション文字列クラスのコマンドについて説明します。

ファンクション文字列クラスに関連するコマンドには、次のものがあります。

- **create function string class** - ファンクション文字列クラスを作成する。
- **alter function string class** - ファンクション文字列クラスの継承関係を変更する。
- **move primary** - エラークラスまたはファンクション文字列クラスと、ファンクション文字列クラスの派生クラスを、別のプライマリサイトに転送します。
- **drop function string class** - ファンクション文字列クラスを削除します。

ファンクション文字列コマンド

ファンクション文字列のコマンドについて説明します。

ファンクション文字列クラス内のファクション文字列に関連するコマンドには、次のものがあります。

- **create function string** - ファンクション文字列を作成する。
- **alter function string** - 既存のファンクション文字列を置き換える。
- **drop function string** - ファンクション文字列を削除します。

ファンクションコマンド

ファンクションコマンドは、非同期プロシージャコールにのみ必要です。

ユーザ定義ファンクションに関連するコマンドには、次のものがあります。

- **create function** - ファンクションを作成します。
- **alter function** - ユーザ定義ファンクションにパラメータを追加します。
- **drop function** - ファンクションを削除します。

ウォームスタンバイデータベースコマンド

Replication Server の *「ウォームスタンバイアプリケーション」* は Adaptive Server の 2 つのデータベースを管理し、その 1 つがもう一方のスタンバイ (バックアップコピー) として機能します。Replication Server からアクティブデータベースとスタンバイデータベースへのコネクションを、*「論理コネクション」* と呼んでいます。

論理データベースコネクションを管理するには、次のコマンドを使用します。

- **create logical connection** - 論理コネクションを作成する。
- **alter logical connection** - 論理コネクションの設定を変更する。
- **drop logical connection** - 論理コネクションを削除する。
- **configure logical connection** - 論理コネクションを設定します。

ウォームスタンバイアプリケーションに関連するタスクを実行するには、次のコマンドを使用します。

- **switch active** - アクティブデータベースを変更する。
- **abort switch** - 可能な場合、**switch active** コマンドをアボートする。

- **wait for switch** - 対話型またはスクリプト形式での Replication Server セッションで、新しいアクティブデータベースの切り替えが完了するまでコマンドが実行されないようにする。
- **wait for create standby** - 対話型またはスクリプト形式の Replication Server セッションで、スタンバイデータベースの準備ができるまで Replication Server がコマンドを受け入れないようにします。

ゲートウェイコマンド

Replication Server のゲートウェイを管理するには、ゲートウェイコマンドを使用します。

Replication Server ゲートウェイにより、複数のレプリケーションサーバ、ID サーバ、RSSD の明示的なログインが最小限に抑えられます。Replication Server ゲートウェイは、RSSD のプライマリユーザ名とパスワードを使用して RSSD に、ID サーバのユーザ名とパスワードを使用して ID サーバに、リモートサーバ ID (RSI) を使用してリモート Replication Server に、メンテナンスユーザ ID を使用してリモート Adaptive Server にログインします。Replication Server 自体にアクセスするとき、この情報を複数回提供する必要はありません。

Replication Server ゲートウェイでは、Replication Server と、Replication Server に直接接続されていないサーバとの通信を可能にするカスケードコネクションもサポートされます。また、1 つのクライアントコネクションを使用して複写ドメインを管理することもできます。

Replication Server のゲートウェイを管理するには、次のコマンドを使用します。

- **connect** - Replication Server を、その RSSD、ID サーバ、リモート Replication Server、またはリモートデータサーバのゲートウェイにする。
- **show connection** - コネクションスタックの内容を表示する。
- **show server** - 現在稼働中のサーバを表示する。
- **disconnect** - サーバへのコネクションを終了する。

ルートコマンド

*ルート*は、送信元 (プライマリ) Replication Server から送信先 (ターゲット) Replication Server への一方向のメッセージストリームです。

Replication Server は、ルートを介して別の Replication Server とメッセージをやり取りします。複写トランザクションのデータも、このようなメッセージの 1 つです。ルートは、ローカルエリアネットワークまたは広域ネットワークにまたがる Replication Server 同士を接続します。

ルートを管理するには、次のコマンドを使用します。

- **create route** - 現在の Replication Server から別の Replication Server へのルートを作成し設定する。
- **alter route** - 現在の Replication Server から別の Replication Server へのルートを変更または再設定する。
- **drop route** - 別の Replication Server へのルートを削除する。
- **suspend route** - 別の Replication Server へのルートをサスペンドする。
- **resume route** - サスペンドされているルートをレジュームする。

システム情報コマンド

システム情報コマンドは Replication Server に関する情報を提供します。

Replication Server に関連する情報を取得するには、次のコマンドを使用します。

- **admin auto_part_path** - 動的にサイズ変更可能な Replication Server パーティションに関する情報を表示します。
- **admin disk_space** - Replication Server がアクセスする各ディスクパーティションの使用状況を表示する。
- **admin echo** - Replication Server が稼働していることを確認するため、ユーザが入力した文字列を返す。
- **admin get_generation** - プライマリデータベースの世代番号を取得する。
- **admin health** - Replication Server の全体的なステータスを表示する。
- **admin log_name** - 現在のログファイルのパス名を表示する。
- **admin logical_status** - ウォームスタンバイアプリケーションでの論理コネクションのステータスを表示する。
- **admin pid** - Replication Server のプロセス ID を表示する。
- **admin quiesce_check** - Replication Server のキューがクワイスされているかどうかを調べる。
- **admin quiesce_force_rsi** - Replication Server がクワイスされているかどうかを確認し、Replication Server にアウトバウンドメッセージを送信するように指示する。
- **admin rssd_name** - Replication Server システムデータベース (RSSD) のデータサーバとデータベースの名前を表示する。
- **admin security_property** - Replication Server によってサポートされる、ネットワークベースのセキュリティメカニズムとセキュリティ機能を表示する。
- **admin security_setting** - Replication Server によってサポートされる、ネットワークベースのセキュリティ機能のステータスを表示する。

- **admin set_log_name** - 既存の Replication Server ログファイルをクローズし、新しいログファイルをオープンする。
- **admin show_connections** - Replication Server からのすべてのコネクションについて情報を表示する。
- **admin show_function_classes** - 既存のファンクション文字列クラスとその親クラスの名前を表示して、継承のレベル数を示す。
- **admin show_route_versions** - Replication Server で開始および終了するルートのバージョン番号を表示する。
- **admin show_site_version** - Replication Server のサイトバージョンを表示する。
- **admin sqm_readers** - インバウンドキューを読み込んでいる各 Replication Server スレッドの読み込みポイントと削除ポイントを表示する。
- **admin stats** - Replication Server のカウンタに関する情報と統計を表示する。
- **admin stats, backlog** - ステーブルキューの現在のトランザクションバックログをレポートする。
- **admin stats, {md | mem | mem_in_use}** - メモリの使用状況に関する情報をレポートする。
- **admin stats, status** - すべてのカウンタのフラッシュステータスを表示する。
- **admin stats, reset** - リセット可能なカウンタをすべてリセットする。
- **admin stats, {tps | cps | bps}** - スループットの 1 秒あたりのトランザクション数、コマンド数、またはバイト数をレポートする。
- **admin time** - Replication Server の現在の時刻を表示する。
- **admin translate** - 特定のデータのデータ型変換を実行し、デリミタを使用したリテラルフォーマットで結果を表示する。
- **admin version** - Replication Server のソフトウェアバージョンを表示する。
- **admin who** - Replication Server で実行されているスレッドについての情報を表示する。
- **admin who_is_down** - 停止している Replication Server のスレッドについての情報を表示する。
- **admin who_is_up** - 実行中の Replication Server のスレッドに関する情報のサブセットを表示する。

パーティションコマンド

Replication Server では、ディスクパーティション上に格納されているステーブルキューにメッセージが格納されます。Replication Agent から受信したメッセージはインバウンドキューに格納され、データサーバまたは他の Replication Server に送信するメッセージはアウトバウンドキューに格納されます。

Replication Server の初期パーティションを作成するには、**rs_init** を使用します。**rs_init** によるパーティションの扱いについては、『Replication Server インストールガイド』と『Replication Server 設定ガイド』を参照してください。

パーティションの追加、削除、または監視には以下を使用します。

- **create partition** - Replication Server でパーティションを使用できるように設定します。 追加する前に、パーティションを作成してください。

  注意： **create partition** は既存の **add partition** コマンドに代わるものです。下位互換性を保つために、**add partition** は **create partition** のエイリアスとして現在もサポートされていますが、今後は推奨されません。
- **create auto partition path** - Replication Server で自動でサイズ変更可能なパーティションを使用できるように設定します。
- **alter partition** - パーティションのサイズを変更します。
- **alter auto partition path** - パーティションファイルのサイズを変更し、動的にサイズ変更可能なパーティションの最大サイズを変更します。
- **drop partition** - Replication Server からパーティションを削除します。
- **drop auto partition path** - Replication Server から自動でサイズ変更可能なパーティションを削除します。
- **rs_helppartition** - すべての Replication Server パーティションに関する情報を表示します。

ステーブルキューとパーティションの詳細については、『Replication Server 管理ガイド 第 1 巻』の「ステーブルキューのパーティション」を参照してください。

設定コマンド

Replication Server が起動すると、システムテーブルまたは設定ファイルから設定パラメータが読み込まれます。設定パラメータには、静的なものと動的なものがあります。動的パラメータは Replication Server の実行中に変更できますが、静的パラメータを変更した場合は Replication Server を再起動する必要があります。

Replication Server を設定するには、次のコマンドを使用します。

- **alter connection** および **configure connection** - データベースへの Replication Server コネクションの特性を変更する。
- **configure logical connection** - ウォームスタンバイアプリケーションの論理コネクション用に Replication Server 設定を変更する。
- **configure replication server** - ルートおよびコネクションに対する Replication Server のパラメータとデフォルトパラメータを変更する。

- **alter route** および **configure route** - ルートの特性を変更する。ルートは、ある Replication Server を別の Replication Server に接続する。

create route および **create connection** を使用してルートおよびコネクションを作成した場合にも、設定パラメータが設定されます。

詳細については、『Replication Server 管理ガイド 第 1 巻』を参照してください。

システム管理コマンド

システム管理タスクの実行、およびシステム障害を引き起こすような問題のトラブルシューティングを行う場合には、次のコマンドを使用します。これらのコマンドを実行するには “sa” パーミッションが必要です。

警告！ これらのコマンドの多くは、非常に制限された環境の下で注意して使用する必要があります。使用する前に、該当するマニュアルをよく調べてください。

- **alter queue** - 16 キロバイトを超える長いメッセージを受け取ったときのステーブルキューの動作を指定します。Replication Server のバージョンが 12.5 以降で、サイトのバージョンが 12.1 以降の場合にのみ使用してください。
- **resume distributor** - データベースへのコネクションの、サスペンドしていたディストリビュータスレッドをレジュームする。
- **shutdown** - Replication Server を停止する。
- **suspend distributor** - データベースへのコネクションのディストリビュータスレッドをサスペンドする。
- **sysadmin apply_truncate_table** - 特定のテーブルの既存のサブスクリプションについて “subscribe to truncate table” オプションをオンまたはオフにして、**truncate table** の複写を有効または無効にする。
- **sysadmin dropdb** - ID サーバからデータベースへの参照を削除する。
- **sysadmin dropldb** - ID サーバから論理データベースへの参照を削除する。
- **sysadmin drop_queue** - ステーブルキューを削除する。
- **sysadmin droprs** - ID サーバから Replication Server への参照を削除する。
- **sysadmin dump_file** - ステーブルキューをダンプするときに使用する代替ログファイルを指定する。
- **sysadmin dump_queue** - ステーブルキューの内容をダンプする。
- **sysadmin erssd** - ERSSD ファイルの場所のチェックと設定のバックアップ、ERSSD ファイルのデフラグ、ERSSD ファイルの移動、ERSSD のスケジュールされていないバックアップを実行できる。

- **sysadmin fast_route_upgrade** - ルートバージョンを、プライマリ Replication Server とレプリケート Replication Server のサイトバージョン番号のうち、どちらか小さい方に更新する。
- **sysadmin hibernate_off** - Replication Server のハイバネーションモードをオフにして、アクティブステータスに戻す。
- **sysadmin hibernate_on** - Replication Server のハイバネーションモードをオンにするか、Replication Server をサスペンドする。
- **sysadmin log_first_tran** - データサーバインタフェース (DSI) キュー内の最初のトランザクションを例外ログに書き込む。
- **sysadmin purge_all_open** - すべてのオープントランザクションをインバウンドキューからパージする。
- **sysadmin purge_first_open** - 最初のオープントランザクションをインバウンドキューからパージする。
- **sysadmin purge_route_at_replicate** - プライマリ Replication Server に対するすべての参照を、レプリケートサイトの Replication Server から削除する。
- **sysadmin restore_dsi_saved_segments** - データベースに再適用できるように、バックログトランザクションをリストアする。
- **sysadmin set_dsi_generation** - レプリケートデータベースをリストアした後、Replication Server がステーブルキュー内のトランザクションを再適用しないように、RSSD 内のデータベース世代番号を変更する。
- **sysadmin site_version** - サイトバージョンレベルを設定する。
- **sysadmin sqm_purge_queue** - Replication Server インタフェース (RSI) のステーブルキューからすべてのメッセージを削除する。
- **sysadmin sqm_unzap_command** - ステーブルキュー内の削除されたメッセージをリストアする。
- **sysadmin sqm_zap_command** - ステーブルキュー内のメッセージを 1 つ削除する。
- **sysadmin sqt_dump_queue** - 各インバウンドまたは DSI キューのトランザクションキャッシュをダンプする。
- **sysadmin system_version** - 複写システムの Replication Server の最小バージョンレベルを設定する。

リカバリコマンド

リカバリコマンドは、データベースを再ロードした後、または Replication Server のステーブルキューに障害が発生したときに、リカバリを行うために使用するものです。

警告！ これらのコマンドの多くは、非常に制限された環境の下で注意して使用する必要があります。使用する前に、該当するマニュアルをよく調べてください。

- **allow connections** - 指定したデータベースのリカバリモードに Replication Server を設定する。
- **ignore loss** - Replication Server でロスが検出された後、メッセージを受け入れるように指定する。
- **rebuild queues** - Replication Server のステーブルキューを再構築する。
- **resume log transfer** - RepAgent スレッドが Replication Server に接続できるようにする。
- **resume queue** - 16 キロバイトより大きいメッセージを受け取ったときに止まったステーブルキューを再開する。このコマンドは、Replication Server のバージョンが 12.5 以降で、サイトバージョンが 12.1 以前の場合にのみ適用される。
- **set log recovery** - Replication Server をデータベースのログリカバリモードに設定する。
- **suspend log transfer** - Replication Server から RepAgent を切断して、どちらも接続できないようにする。

リカバリ手順の詳細については、『Replication Server 管理ガイド 第 2 巻』を参照してください。

トピック

データ型、識別子、予約語、SAP ASE のサポート、混合バージョン環境について説明します。

データ型

SAP Replication Server がサポートしている SAP データ型について説明します。

表 1 : SAP Replication Server がサポートしているデータ型

データ型クラス	データ型
真数値 (整数)	*bigint*、*int*、*smallint*、*tinyint*、*unsigned bigint*、*unsigned int*、*unsigned smallint*、*unsigned tinyint*、*rs_address*
真数値 (10 進数)	*decimal*、*numeric*、*identity*
概数値 (浮動小数)	*float*、*real*
文字	*char(n)*、*varchar(n)*、*text*、*opaque*
通貨	*money*、*smallmoney*
日付および時間	*datetime*、*smalldatetime*、*date*、*time*、*timestamp*、*bigdatetime*、*bigtime*
バイナリ	*binary(n)*、*varbinary(n)*、*image*、*rawobject*、*rawobject in row*
ビット	*bit*
Unicode	*unichar(n)*、*univarchar(n)*、*unitext*
Java	*rawobject*、*rawobject in row*
データ型定義	「データ型定義」を参照してください。

RCL は次の SAP のデータ型を間接的にサポートしています。

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

次のデータ型はサポートされていません。

- *float* データ型の精度の引数 (オプション)
- 真数値 (10 進数) データ型の精度と位取りの引数 (オプション)

サポートされていないデータ型がカラムに含まれている場合は、表 1 のサポートされているデータ型のうち、サポートされているデータ型の 1 つを使って複写定

義を作成すれば複写できます。たとえば、*double precision* のカラムを複写するときは、複写定義に *float* としてカラムを定義します。ユーザ定義のデータ型のカラムを複写するには、基本となるデータ型を複写定義で使用してください。

SAP ASE で *nchar* と *nvarchar* 型のカラムに格納されているデータを複写するには、*char* と *varchar* の SAP Replication Server データ型を、それぞれ使用してください。これらの異なる点は、長さの単位が *nchar* と *nvarchar* では SAP ASE のネイティブ文字セットの文字数であるのに対し、*char* と *varchar* では常にバイトであるという点だけです。

対応する SAP Replication Server の *char* データ型と *varchar* データ型の長さを調べるには、*nchar* データ型または *nvarchar* データ型の宣言された長さを、SAP ASE のグローバル変数 *@@ncharsize* の値で乗算します。

たとえば、*@@ncharsize* が 1 (iso_1、cp850、cp437、roman8、mac のように、すべてがシングルバイト文字セット) の場合は、1 つ 1 つが一致し、宣言された長さも同じになります。*@@ncharsize* が 2 (シフト JIS や EUC-JIS など一部のマルチバイト文字セットの場合) であれば、*nchar* データ型と *nvarchar* データ型の宣言された長さに 2 をかけたものを、複写定義で *char* と *varchar* として宣言します。

以降の項で、サポートされているデータ型について説明します。 SAP ASE のデータ型の詳細については、『Adaptive Server Enterprise リファレンスマニュアル』を参照してください。

SAP Replication Server は、SAP 以外のデータサーバに対する一連のデータ型定義をサポートしており、あるデータ型のカラム値を、レプリケートデータベースの異なるデータ型のカラムに複写できます。 異機種データ型サポート (HDS) の詳細については、『管理ガイド 第 1 巻』を参照してください。

真数値 (整数) データ型

真数値 (整数) データ型について説明します。

Replication Server は次の真数値 (整数) データ型をサポートしています。

- *bigint* - -2^{63} ～ $+2^{63} - 1$ (-9,233,372,036,854,775,808 ～ +9,233,372,036,854,775,807) の整数値
- *int* - -2^{31} ～ $+2^{31} - 1$ (-2,147,483,648 ～ +2,147,483,647) の整数値
- *smallint* - -2^{15} ～ $+2^{15} - 1$ (-32,768 ～ +32,767) の整数値
- *tinyint* - 0 ～ 255 の正の整数値
- *unsigned bigint* - 0 ～ 18,446,744, 073, 709,551,615 の整数値
- *unsigned int* - 0 ～ 4,294,967,295 の整数値
- *unsigned smallint* - 0 ～ 65535 の整数値
- *unsigned tinyint* - 0 ～ 255 の整数値

基本となるデータ型に *int* を使用する *rs_address* データ型は、特別なサブスクリプション解析メソッドで使用されます。*rs_address* データ型の詳細については、『Replication Server 管理ガイド 第 1 巻』を参照してください。

参照：

- create subscription (376 ページ)

真数値 (10 進数) データ型

真数値 (10 進数) データ型について説明します。

Replication Server は、次の真数値 (10 進数) データ型をサポートしています。

- *decimal* - -10^{38} ～ 10^{38} -1 の真数値 (10 進数)。
- *numeric* - -10^{38} ～ 10^{38} -1 の真数値 (10 進数)。

複写定義を作成する場合、*numeric* データ型の宣言から長さと精度を省略してください。 Replication Server は、精度に影響を与えずに *numeric* の値を処理します。

注意： 複写定義の **where** 句で numeric データ型を使用している場合は、値に精度の情報を含める必要があります。

identity カラム (ID カラム) は基本となるデータ型として *numeric* を使用しており、1 ～ 10^{38} -1 までの、位取り 0 の真数値 (10 進数) です。

identity カラムを含むテーブルの複写定義を作成する場合は、カラムのデータ型として identity を指定してください。

このコマンドは、**insert** コマンドの前に複写済みテーブルに適用されます。

```
set identity_insert table_name on
```

このコマンドは、**insert** コマンドの後で複写済みテーブルに適用されます。

```
set identity_insert table_name off
```

identity カラムは、**update** コマンドでは更新できません。

レプリケートデータサーバが Adaptive Server であり、テーブルに *identity* カラムが含まれている場合、Transact-SQL® **identity_insert** オプションを使用するには、メンテナンスユーザがレプリケートデータベースのテーブルの所有者 (または dbo ユーザか dbo ログイン名のエイリアス) である必要があります。

概数値 (浮動小数点) データ型

概数値 (浮動小数点) データ型について説明します。

概数値 (浮動小数点) データ型には、次の 2 種類があります。

- *float* - 正と負の浮動小数点数。精度と有効桁数は、機種によって異なります。記憶サイズは 8 バイト。
- *real* - *float* に似ているが、記憶サイズは 4 バイト。

複写定義のプライマリキーに概数値 (浮動小数点) データ型を使用したカラムを含めないでください。

文字データ型

文字データ型について説明します。

注意： Unicode のデータ型 *unichar*、*univarchar*、*unitext* は、対応する *char*、*varchar*、*text* と同じ属性を備えています。

- *char(n)* - 32,768 までのシングルバイト文字、記号、数値の組み合わせ。*n* には文字列の最大数を指定します。*char* には 0 文字の値を格納できますが、*n* には 1 ～ 32,768 の間の値を指定します。マルチバイト文字列は 32,768 バイト以下で指定してください。
- *varchar(n)* - 32,768 までのシングルバイト文字、記号、数値の組み合わせ。null 値を入力できるよう定義されている場合、*varchar* には 0 文字の値を格納できますが、*n* には 1 ～ 32,768 の間の値を指定します。

  char と *varchar* のデータの違いは、Adaptive Server データベースでの値の格納方法です。Replication Server では、これらは同じ型として処理されますが区別して管理されます。そのため、記憶方法はプライマリデータベースとレプリケートデータベースで同じになります。
- *text* - 長さが 2,147,483,647 バイトまでの可変長文字カラム。

  Replication Server 15.1 は、テキストポインタがある *text*、*unitext*、*image* データ型と、テキストポインタなしの *text*、*unitext*、*image* データ型など、ラージオブジェクト (LOB) データ型間のデータ型変換をサポートしています。

文字データの入力フォーマット

リテラルの *char*、*varchar*、 *text* 値またはそれに相当する値は、一重引用符で囲む必要があります。

char および *varchar* リテラルに一重引用符を埋め込む方法は 2 つあります。埋め込みの一重引用符を表すには、次の例のように引用符を 2 つ続けて入力します。

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

最初と最後の引用符は、文字列を区切るためのものです。内側の 2 組の引用符が、埋め込みの一重引用符として解釈されます。

Replication Server は、文字値をファンクション文字列テンプレートの変数に置き換えるときに、一重引用符を生成します。

参照：

- create function string (318 ページ)

通貨データ型

通貨データ型は、通貨や金銭の値を固定精度で格納します。

- *money* - -922,337,203,685,477.5808 ～ 922,337,203,685,477.5807 の金銭の値。精度は通貨単位の 1/10000。記憶サイズは 8 バイト。
- *smallmoney* - -214,748.3648 ～ 214,748.3647 の金銭の値。精度は通貨単位の 1/10000。記憶サイズは 4 バイト。

金銭データの入力フォーマット

money および *smallmoney* リテラル値には、値の前に US ドル記号 ($) を付けて浮動小数のデータ型と区別します。値が負の場合は、ドル記号の後にマイナス記号を付けます。

Replication Server は、*money* と *smallmoney* の値をファンクション文字列の出力テンプレートに置き換えるときに、ドル記号を出力します。

日時および日付と時刻のデータ型

日付と時刻のデータ型について説明します。

SAP Replication Server は、日付と時刻のデータに次のデータ型をサポートしています。

- *datetime* - 1753 年 1 月 1 日から 9999 年 12 月 31 日までの日付と時刻。記憶サイズは 8 バイト。そのうちの 4 バイトは基本日付の 1900 年 1 月 1 日からの日数、残りの 4 バイトは 1/300 秒までの時間に使用されます。基本日付より前の日付は、負の値として格納されます。
- *smalldatetime* - 1900 年 1 月 1 日から 2079 年 6 月 6 日までの日付と時刻。精度は 1 分。記憶サイズは 4 バイト。1900 年 1 月 1 日からの日数および午前 0 時からの分数に、それぞれ 1 つの small integer が使用されます。
- *date* - 0001 年 1 月 1 日から 9999 年 12 月 31 日までの日付。記憶サイズは 4 バイト。基本日付より前の日付は、負の値として格納されます。
- *time* - 12:00:00 AM から 11:59:59.999 PM までの時刻。記憶サイズは 4 バイト。
- *bigtime* - SAP IQ の *TIME* データ型に対応する時、分、秒、秒以下で構成される時刻。 秒の小数点以下は 6 桁まで格納されます。 *bigtime* の値には 8 バイトの格納領域が必要です。 ODBC 規格では、*bigtime* データ型の精度を秒の単位までに制限しています。 そのため、**WHERE** 句の比較に、秒の単位より高い精度に依存する *bigtime* データ型を使用しないでください。

  bigtime の有効範囲は 12:00:00.000000AM から 11:59:59.999999PM までです。

- *bigdatetime* - SAP IQ の *TIMESTAMP* データ型に対応する年、月、日、時、分、秒、秒以下で構成される時点。秒の小数点以下は 6 桁まで格納されます。日にはゼロでない値を格納してください。*bigdatetime* 値には 8 バイトの格納領域が必要です。
  bigdatetime の有効範囲は、0001 年 1 月 1 日 から 9999 年 12 月 31 日の 12:00:00.000000AM から 11:59:59.999999PM までです。1600-02-28 23:59:59 から 7911-01-01 00:00:00 の範囲を超える *bigdatetime* データは表示が不完全になる可能性がありますが、データベースには *bigdatetime* の完全な値が格納されます。
- *timestamp* - 基本のデータ型として *varbinary*(8) を使用します。ステータスビットがあることが、*timestamp* が *varbinary* と異なる点である。
  timestamp は、SAP Replication Server 15.1 には *timestamp* として、SAP Replication Server 15.0.1 以前には *varbinary* として送信されます。

  注意： timestamp カラムへの複写は、SAP ASE 15.0.2 以降でのみサポートされています。

日時の値の入力フォーマット

datetime および *smalldatetime* の値は、一重引用符で囲んで文字列として入力します。

Replication Server は、*datetime* の値をファンクション文字列の出力テンプレートに置き換えるときに、*datetime* を一重引用符で囲みます。*datetime* 型の変数を含むファンクション文字列を作成するときは、この点に注意してください。

データの日付と時間の部分はそれぞれ区別して認識されるので、時間は日付の前後どちらにも配置できます。時間を省略した場合は、真夜中 (12:00:00:000AM) とみなされます。日付を省略した場合は、1900 年 1 月 1 日とみなされます。

時刻は次の一般的な規則に従って入力します。

- 時間の範囲は 0 ～ 23、分と秒の範囲は 0 ～ 59、ミリ秒の範囲は 0 ～ 999 です。
- 時刻として認識されるためには、値にコロンまたは “AM” か “PM” のインジケータが必要です。
- “AM” または “PM” と値の間にはスペースを入れても入れなくてもかまいません。12AM は午前 0 時、12 PM は正午を示します。AM を指定すると、時間は 1 ～ 12 (0 を 12 の代わりに使用できる) の間になります。PM を指定すると、時間は 13 ～ 23 の間になります。
- ミリ秒の前には、コロンまたはピリオドが使用できます。前にコロンを使用すると、数値は 1000 分の 1 秒を示します。前にピリオドを使用すると 1 桁は 10 分の 1 秒、2 桁は 100 分の 1 秒、3 桁は 1000 分の 1 秒を示します。たとえば、“12:30:20:1” は 12 時 30 分 20.001 秒、“12:30:20.1” は 12 時 30 分 20.1 秒を表します。

- 時間値はどの部分も省略できます。ただし、秒を省略する場合は、ミリ秒も省略します。分を省略する場合は、秒とミリ秒も省略します。省略した部分はすべて 0 とみなされます。

次に、時間リテラルの例を示します。

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

日付は次の規則に従って、年、月、日を任意の順序で入力します。

- 月は 1 ～ 12 の数字、または英語表記の月名かその省略形 (3 文字) で入力できます。
- 月を数字で表記する場合、日付の各部分をスラッシュ (/)、ハイフン (-)、またはピリオド (.) で区切り、月、日、年の順序で指定します。
- 以下の例で、1998 年 3 月 15 日という日付を入力する方法をいくつか示します。

  ```
  3-15-1998
  March-15-1998
  March 15 1998
  15/March/1998
  March.15.1998
  ```

- 月には、英語名を 3 文字に省略した形を使用できます。大文字と小文字は区別されません。

  ```
  JAN 9 1998
  31 oct 1997
  ```

- 月に英語名を使用する場合は、月名と日付の後にカンマを使用できます。次に、有効な日付を示します。

  ```
  Nov 17, 1997
  1997 Nov, 17,
  17 Nov, 1997
  ```

- 年は 1、2、または 4 桁で入力できます。50 未満の 1 桁または 2 桁の年は、今世紀 (21 世紀) を示します。50 以上の 2 桁の年は、前世紀 (20 世紀) を示します。
- 4 桁の年は日付値のどこに置いても認識されます。2 桁の年は、月の日付の後に置いてください。
- 英語の月名と 4 桁の年を使用する場合は、月の日付を省略できます。その場合、日付のデフォルトは月の最初の日 (1 日) になります。月名の後に使用できるセパレータはカンマだけです。

  Replication Server はこの日付を 1998 年 5 月 1 日と解釈します。

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

以下の例では、複写定義、ファンクション複写定義、サブスクリプションで *bigdatetime* と *bigtime* を使用する方法を示します。この例では、以下の単語は次のことを示します。

- PDS - プライマリデータサーバ
- pdb1 - プライマリデータベース
- RDS - レプリケートデータサーバ
- rdb1 - レプリケートデータベース
- tb1 - テーブル
- col1, col2, col3 - カラム
- rep1 - 複写定義
- func1 - ファンクション複写定義
- sub1 - サブスクリプション

例 1
複写定義でデータ型を使用します。

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

例 2
ファンクション複写定義でデータ型を使用します。

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

例 3
サブスクリプションでデータ型を使用します。

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

バイナリデータ型

バイナリデータ型について説明します。

バイナリデータ型には、以下の種類があります。

- *binary(n)* - 32,768 バイトまでの固定長バイナリデータ。*binary* データ型は、数値ではなく、プログラミングコードや図の格納に使用します。*n* には値の最大バ

イト長を指定します。*binary* には 0 バイトの値を格納できますが、*n* には 1 ～ 32,768 の間の値を指定します。

- *varbinary(n)* - 32,768 バイトまでの可変長バイナリデータ。*varbinary* データ型は、数値ではなく、プログラミングコードや図の格納に使用します。*n* には値の最大バイト長を指定します。*varbinary* には 0 バイトの値を格納できますが、*n* には 1 ～ 32,768 の間の値を指定します。

  binary と *varbinary* のデータの違いは、Adaptive Server データベースでの値の格納方法です。Replication Server では、これらは同じ型として処理されますが区別して管理されます。そのため、記憶方法はプライマリデータベースとレプリケートデータベースで同じになります。
- *rawobject in row* - 255 バイトの可変長バイナリデータ。*rawobject in row* データ型は、直列化された Java 値を、テーブルに割り付けられたデータページ内に格納するために使用します。

  Replication Server は、*rawobject in row* データを *varbinary* データとまったく同様に扱います。*rawobject in row* の基本データ型は *varbinary(255)* です。
- *rawobject large in row* - 32,768 バイトの可変長バイナリデータ。*rawobject large in row* データ型は、直列化された Java 値を、テーブルに割り付けられたデータページ内に格納するために使用します。

  Replication Server は、*varbinary* データと同じ方法で *rawobject large in row* データを扱います。*rawobject large in row* の基本データ型は *varbinary(32768)* です。
- *image* - 長さが 2,147,483,647 バイトまでの可変長バイナリカラム。

  Replication Server 15.1 は、テキストポインタがある *text*、*unitext*、*image* データ型と、テキストポインタなしの *text*、*unitext*、*image* データ型など、LOB データ型間のデータ型変換をサポートしています。
- *rawobject* - 長さが 2,147,483,647 バイトまでの可変長バイナリカラム。*rawobject* データ型は、直列化された Java の値の格納に使用します。Replication Server では、*rawobject* データのデータ型変換はサポートされていません。つまり、複写定義でカラムを *rawobject* として宣言している場合、プライマリテーブルのカラムも *rawobject* とします。

  Replication Server は、*rawobject* データを *image* データとまったく同様に扱います。*rawobject* の基本データ型は *image* です。

参照：

- Java データ型 (36 ページ)

バイナリデータの入力フォーマット

Enter *binary*、*varbinary*、 *image*、*rawobject*、*rawobject in row*、*rawobject large in row* の各リテラル値は、16 進数の 0 ～ 9 および A ～ F (または a ～ f) を使用して入力します。

各バイトは 2 桁の 16 進数で表され、値全体の前に “0x” が付きます。次に、10 バイトの *binary* 文字列の例を示します。

```
0x010305070B0D1113171D
```

Replication Server は、*binary* 値をファンクション文字列の出力テンプレートに置き換えるときに、プレフィクス “0x” を出力します。

Bit データ型

bit データ型はブール値に使用します。

- *bit* - 1 または 0。1 または 0 以外の整数値は 1 と解釈されます。

Unicode データ型

Replication Server では、*unichar(n)*、*univarchar(n)*、*unitext* の 3 種類の Unicode データ型がサポートされています。Unicode を使用すると、1 つのデータサーバ内で、さまざまな言語グループに属する言語を混合して使用できます。

Unicode データ型の動作は、Replication Server の対応するデータ型とまったく同じです。

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

Unicode データ型の構文とセマンティックは対応するデータ型と同じですが、Unicode 値は Replication Server のデフォルト文字セットに関係なく、常に UTF-16 で格納されます。*unichar(n)* は、固定幅の NULL 入力不可能なデータ型です。*univarchar(n)* は、可変幅の NULL 入力可能なデータ型です。*unichar(n)* と *univarchar(n)* の場合は、*n* を使用して Unicode の文字数を指定します。*unitext* は可変幅の NULL 入力可能なデータ型です。

次のことができます。

- *unichar(n)* カラム、*univarchar(n)* カラム、*unitext* カラムをレプリケートデータベースとスタンバイデータベースに複写する。
- *unichar(n)* カラムと *univarchar(n)* カラムを複写定義のプライマリキーで使用する。

- 複写定義、および関連するサブスクリプションとアーティクルの **where** 句で、*unichar(n)* カラムと *univarchar(n)* カラムをサーチャブルカラムとして使用する。
- ファンクション複写定義、および関連するサブスクリプションとアーティクルの **where** 句で、*unichar(n)* カラムと *univarchar(n)* カラムをサーチャブルカラムとして使用する。
- 機種の異なるデータサーバ間で複写するときに、*unichar(n)* カラム、*univarchar(n)* カラム、*unitext* カラムを使用する。

text と同様に、次の制限があります。

- *unitext* カラムは、複写定義においてプライマリキーの一部になれない。
- *unitext* カラムは、複写定義でサーチャブルカラムとして指定できない。
- *unitext* カラムは、ファンクション複写定義でサーチャブルカラムとして指定できない。
- *unitext* データ型は、基本データ型やデータ型定義として、またはカラムレベル変換やクラスレベル変換の変換元や変換対象として使用できない。

unichar カラムと *univarchar* カラムを正しく複写するには、utf8 文字セットを使用する必要なしに、**unicode_format** 設定を使用する方法もあります。

Replication Server の文字セットが UTF-8 以外の場合、Replication Server は、ASCII-7 コード範囲内の *unichar* 文字および *univarchar* 文字しか複写できません。

アップグレードに関する問題

unichar データ型および *univarchar* データ型を完全にサポートするには、プライマリ Replication Server とレプリケート Replication Server の両方がバージョン 12.5 以降を実行している必要があります。

unitext データ型を完全にサポートするには、プライマリ Replication Server とレプリケート Replication Server の両方がバージョン 15.0.1 以降を実行しており、ルートバージョンが 15.0.1 以降、LTL バージョンが 700 以上であることが必要です。connect source で LTL バージョンが 700 より小さい場合、RepAgent は *unitext* カラムを *image* に変換します。

sysadmin upgrade, "route" により *unichar*、*univarchar*、*unitext* データ型を参照している複写定義がアップストリーム Replication Server からコピーされます。

混合バージョンに関する問題

混合バージョン環境では、プライマリ Replication Server とレプリケート Replication Server 間のルートバージョンによって、サポートされる機能が決まります。

- *bigdatetime* と *bigtime* は Adaptive Server バージョン 15.5 以降でのみサポートされています。少なくともプライマリデータサーバが Adaptive Server 15.5 以降であれば、次のように対処できます。

  - プライマリおよびレプリケート Replication Server がバージョン 15.5 以降で、レプリケート Adaptive Server がこれらのデータ型をサポートしていない場合は、その 2 つのデータ型をそれぞれ *varchar* データ型にマッピングする定義を複写定義に含めます。または、複写定義でその 2 つのデータ型の代わりに *varchar* データ型を使用します。
  - プライマリ Replication Server がバージョン 15.5 以降で、レプリケート Replication Server と Adaptive Server がこれらのデータ型をサポートしていない場合は、複写定義でその 2 つのデータ型の代わりに *varchar* データ型を使用します。
  - プライマリ Replication Server、レプリケート Replication Server、レプリケート Adaptive Server がこれらのデータ型をサポートしていない場合は、RepAgent が自動的に *varchar* データ型を Replication Server に送信します。
- 引用符付き識別子の複写を成功させるには、プライマリ Replication Server とレプリケートデータサーバに接続する Replication Server のバージョンを 15.2 にします。ただし、ルート上の中間 Replication Server は、以前のバージョンでもかまいません。
- *unitext* カラムで作成された複写定義は、バージョン 12.6 以前の Replication Server に送信されない。
- バージョン 12.6 以前の Replication Server によってサブスクリプションが作成された複写定義は、*unitext* カラムを追加するように変更できない。
- *unitext* カラムで作成された複写定義は、*unitext* カラムを削除した場合、バージョン 12.6 以前の Replication Server に送信される。

Java データ型

Java データ型について説明します。

Java カラムは、次の 3 種類の Replication Server データ型のいずれかとして、扱われます。

- *rawobject - image* データと同様に、データベース内の独立したロケーションに情報が格納されます。*rawobject* の基本データ型は *image* です。*rawobject* は Replication Server の Java カラムのデフォルトデータ型です。
- *rawobject in row -* 情報は、*char* データと同じ方法で、テーブルに割り付けられた連続するデータページ上のデータベースに格納される。*rawobject in row* の基本データ型は *varbinary(255)* です。
- *rawobject large in row* のように、情報は *char* データと同じ方法で、テーブルに割り付けられた連続するデータページ上のデータベースに格納される。*rawobject large in row* の基本データ型は *varbinary(32768)* です。

rawobject データ型、*rawobject in row* データ型、*rawobject large in row* データ型は、それぞれの基本データ型とのみ互換性があります。3 つのデータ型の間に互換性

はありません。一方の Java データ型と他方の Java データ型の間で複写を実行することはできません。

rs_subcmp 調整ユーティリティでは、Java データ型はそれぞれの基本データ型として扱われます。

Opaque データ型

opaque データ型は、Replication Server が現在サポートしていないデータ型を処理します。RepAgent は、ターゲットデータサーバに直接適用するフォーマットデータを Replication Server に提供します。このようなデータ型の例としては、Oracle の *anydata* データ型や Microsoft SQL Server の *sql_variant* データ型などがあります。

制限事項

opaque データ型の制限事項は次のとおりです。

- 複写定義、サブスクリプション、アーティクルのサーチャブルカラムと **where** 句で opaque データ型を使用できない。
- **map to** 句を *opaque* データ型で使用できない。
- 複写定義に *opaque* データ型のカラムまたはパラメータが存在する場合は、動的 SQL 機能を使用できない。
- ファンクション文字列にリモートプロシージャコール (RPC) が含まれる場合は、*opaque* データ型を使用できない。
- 文字変換やバイト順序変換を *opaque* データに適用できない。

混合バージョンのサポート

opaque データ型をサポートするには、プライマリ Replication Server とレプリケート Replication Server のサイトバージョンが 15.1 以降、LTL のバージョンが 710 以降であることが必要です。

データ型定義

SAP では、ユーザ定義データ型とデータ型クラスのセットを提供しています。 この 2 つを使用して、次のデータサーバ間で複写を行うときに、カラム値のデータ型を変更できます。

- SAP データサーバの間
- SAP データサーバと SAP 以外のデータサーバの間
- SAP 以外の同機種データサーバの間
- SAP 以外の異機種データサーバの間

データ型定義は、SAP 以外のデータ型を、基本となる SAP Replication Server のネイティブデータ型に置き換えて記述します。 基本データ型は、データ型定義に関

連する最大長と最小長を決定し、他のデータ型属性にデフォルト値を提供します。 また、基本データ型は、データ型定義に関連するデリミタも定義します。

各データ型クラスには、特定のデータサーバのデータ型定義が含まれます。 データ型クラスには、次のものがあります。

- SAP ASE - **rs_sqlserver_dt_class**
- SAP SQL Anywhere - **rs_asa_dt_class**
- DB2 - **rs_db2_dt_class**
- Microsoft SQL Server - **rs_msss_dt_class**
- Oracle - **rs_oracle_dt_class**
- SAP HANA データベース - **rs_hanadb_dt_class**

各データ型クラスでサポートされるデータ型定義のリストと詳細については、『異機種間複写ガイド』を参照してください。

識別子

識別子は、データベース、テーブル、複写定義、パブリケーション、サブスクリプション、ファンクション、パラメータ、ファンクション文字列変数などのオブジェクトの名前です。

次のオブジェクトの識別子の長さは 1 ～ 255 バイトです。

- テーブル
- カラム
- プロシージャ
- パラメータ
- ファンクション - ファンクション複写定義または内部機能の一部

  注意： **create function**、**alter function**、**drop function** の各コマンドは、長い識別子をサポートしていません。ファンクション名およびこれらのコマンドのパラメータは、最大で 30 バイトです。
- ファンクション文字列
- 複写定義 - テーブル複写定義、ファンクション複写定義、データベース複写定義を含む。
- アーティクル
- パブリケーション
- サブスクリプション

これ以外のすべての識別子の長さは 1 ～ 30 バイトです。

識別子が引用符で囲まれていない場合、最初の文字は ASCII 文字でなければなりません。2 文字目以降の文字には、ASCII 文字、数字、ドル記号 ($) またはアンダースコア (_) を使用できます。スペースは使用できません。

文字 “rs_” で始まる識別子は、Replication Server で予約されています。予約語のリストについては、「予約語」を参照してください。

Replication Server ファンクションおよび Adaptive Server ストアドプロシージャのパラメータ名のみが、拡張子を @ 文字で始めることができます。

- Replication Server ファンクションのパラメータ名は、@ 文字を含め最大で 256 バイトまで指定できる。
- Adaptive Server ストアドプロシージャのパラメータ名は、@ 文字を含め最大で 255 バイトまで指定できる。

予約語を二重引用符で囲むことで、識別子として使用できます。引用符で囲んだ場合、スペースや !@#$%^&*() などの本来は使用できない文字、8 ビット文字やマルチバイト文字も使用できるようになります。ただし、引用符で囲んだ場合でも、識別子の最後の (連続する) 空白はすべて削除されます。例：

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

警告！ Adaptive Server では、**quoted_identifier** をオンに設定すると、識別子を引用符で囲むことができます。これにより、Adaptive Server のオブジェクト名に予約語を使用できるようになります。ただし Replication Server では、Adaptive Server に送信するコマンド内の引用符で囲まれた識別子が認識されません。このため、Adaptive Server 複写オブジェクトの名前に Transact-SQL キーワードは使用できません。必要であれば、ファンクション文字列を変更して、複写オブジェクトの識別子を引用符で囲むことができます。

ファンクション文字列テンプレート内の変数名は、疑問符 (?) で囲みます。たとえば、プライマリデータベースを参照するファンクション文字列に、次の変数名を使用できます。

```
?rs_origin_db!sys?
```

引用符で囲んだ識別子を使用する場合は、次のようになります。

```
?”rs_origin_db”!sys?
```

識別子のネームスペース

識別子のネームスペースとは、その識別子が Replication Server に認識される範囲 (スコープ) のことです。

たとえば、データサーバ名は複写データシステム全体 (Replication Server でデータサーバのデータを複写する 1 個のデータシステム全体) でユニークでなければならないため、グローバルネームスペースを持つと言えます。一方、カラム名はテー

ブルのスコープを持ちます。複数のテーブルで同じ名前のカラムを使用できるため、カラム名はテーブルの名前で修飾する必要があります。

Replication Server 識別子のネームスペーステーブルは、各識別子の Replication Server でのネームスペースを示します。

表 2 : Replication Server 識別子のネームスペース

識別子の種類	ネームスペース
アーティクル	パブリケーション
カラム	テーブル
データサーバ	グローバル
データベース	データサーバ
エラークラス	グローバル
ファンクション文字列クラス	グローバル
ファンクション	複写定義。Adaptive Server データベースで実行される非同期プロシージャで使用するユーザ定義ファンクションには、グローバルな範囲でユニークな名前が必要。ただし、プロシージャにテーブル複写定義が指定されている場合は除く。
ファンクション複写定義	グローバル
パラメータ	機能
パブリケーション	プライマリデータサーバとデータベース
複写定義	グローバル
Replication Server	グローバル
サブスクリプション	複写定義、レプリケートデータサーバ、データベース。サブスクリプションには、グローバルな範囲でユニークな名前が必要。
ユーザ	Replication Server
変数	ファンクションまたはテーブル

複写定義やその他のグローバルなスコープを持つ Replication Server オブジェクトには、グローバルネームスペースでユニークな名前が確保されるよう、何らかの命名規則を適用してください。

警告！ グローバルネームスペースを持つ識別子の管理は慎重に行ってください。グローバルネームスペースで重複が発生しても、すべてが即座に検出されるわけではなく、後でエラーが発生します。

グローバルでない範囲のネームスペースを持つ識別子に、修飾が必要な場合があります。たとえば、Replication Server の多くのコマンドでは、次のように、テーブ

ルが格納されているデータサーバとデータベースを指定する **at** 句が構文に含まれています。

```
at data_server.database
```

正しく設定されたシステムでは、すべてのサーバで同じソート順が使用されます。サーバ間で同じソート順が使用されていない場合、異なるサーバで識別子が正しく比較されないため、ネットワーク上で異常が発生する可能性があります。

予約語

Replication Server の予約語について説明します。

Replication Server の予約語テーブル内の語は、Replication Server で予約されているキーワードです。ここでは小文字で記載していますが、Replication Server では大文字と小文字は区別されません。したがって、これらの単語の大文字と小文字の組み合わせもすべて予約語とみなします。また、"rs_" で始まるキーワードや識別子もすべて Replication Server で予約されています。

表 3 : Replication Server の予約語

	予約語
A	**abort**、**_aco**、**action**、**activate**、**active**、**add**、**_add_recov_pending**、**admin**、**_af**、**after**、**all**、**allow**、**alter**、**always_rep**、**always_replicate**、**_alt_attr2**、**_alter_attributes2**、**_alter_col_objid**、**and**、**_ap**、**_apd**、**article**、**articles**、**_apd**、**append**、**applied**、**_ar**、**_arp**、**article**、**articles**、**as**、**assign**、**at**
B	**before**、**begin**、**_bf**、**_bg**
C	**changed**、**_ch**、**check**、**ci**、**class**、**_cm**、**columns**、**commit**、**configure**、**connect**、**connection**、**connections**、**connector**、**controller**、**create**
D	**database**、**datarow**、**dataserver**、**ddl**、**debug**、**define**、**definition**、**deletelen**、**deliver**、**description**、**disconnect**、**display_only**、**distribute**、**distribution**、**distributor**、**_dln**、**_dr**、**drop**、**drop_repdef**、**_ds**、**dsi_suspended**、**dump**、**dynamic**
E	**enable**、**error**、**exec**、**execute**、**expand**
F	**_fi**、**first**、**for**、**from**、**function**、**functions**
G	**get**、**grant**
H	**_ha**、**hastext**、**holdlock**
I	**ignore**、**in**、**incrementally**、**init**、**installjava**、**internal_use_only**、**into**、**_instj**、**_isb**、**_isbinary**
J	**_jar**

	予約語
K	**key**
L	**language**、**large**、**last**、**load**、**log**、**logical**、**loss**
M	**maintenance**、**map**、**marker**、**materialization**、**message**、**_mbf**、**min_before**、**min_row**、**minimal**、**move**、**_mr**
N	**name**、**named**、**_ne**、**never_rep**、**new**、**next**、**no**、**no_password**、**none**、**not**、**notrep**、**nowait**、**npw**、**_nr**、**_nu**、**null**、**nullable**
O	**of**、**off**、**offset**、**on**、**only**、**open_xact**、**or**、**_os**、**osid**、**output**、**overwrite**、**owner**
P	**parameters**、**parent**、**partialupd**、**partition**、**passthru**、**password**、**primary**、**procedure**、**procedures**、**profile**、**_pu**、**public**、**publication**、**purge**
Q	**queue**、**queues**、**quoted**
R	**_rar**、**rebuild**、**reconfigure**、**recover**、**recovery**、**references**、**reject**、**remove**、**_rename_phystable_name**、**_reorder_columns**、**repfunc**、**replay**、**rep_if_changed**、**replicate**、**replicate_if_changed**、**replication**、**request**、**_resetq**、**_resetqueue**、**resetqueue**、**resume**、**resync**、**retry**、**revoke**、**_rc**、**_rf**、**_rl**、**_roc**、**rollback**、**route**、**row**、**rpc**、**_rpn**、**_rs_alterrepdef**、**rs_rcl**、**rs_ticket**、**_rsc**、**rsrpc**
S	**scan**、**schedule**、**searchable**、**segment**、**select**、**send**、**sendallxacts**、**seq**、**server**、**set**、**shutdown**、**site**、**size**、**skip**、**source**、**sql**、**sqlddl**、**sqldml**、**_st**、**standby**、**starting**、**status**、**stdb**、**string**、**subscribe**、**subscription**、**suspend**、**suspension**、**switch**、**sys_sp**、**sysadmin**、**system**
T	**table**、**tables**、**template**、**textcol**、**textlen**、**_tl**、**_tn**、**to**、**_tp**、**tpinit**、**tpnull**、**_tr**、**trace**、**tran**、**transaction**、**transactions**、**transfer**、**truncate**、**truncation**、**twosave**
U	**_up**、**unsigned**、**update**、**use**、**user**、**username**、**using**
V	**validate**、**verify**、**verify_repserver_cmd**、**vers**
W	**wait**、**warmstdb**、**_wh**、**where**、**with**、**without**、**withouttp**、**_wo**、**writetext**
Y	**_yd**、**yielding**
Z	**_zl**、**zerolen**

SAP ASE のサポート

SAP ASE に対する SAP Replication Server の特殊なサポートについて説明します。

SAP Replication Server は以下の機能を提供して、海外のお客様をサポートしています。

- 8 ビット文字セット、マルチバイト文字セット、Unicode 文字セットを含む、SAP がサポートしているすべての文字セットをサポートします。
- バイナリソート順以外のソート順や Unicode ソート順を含む、SAP がサポートしているすべてのソート順をサポートします。
- 英語、フランス語、ドイツ語、日本語で SAP Replication Server メッセージを表示します。
- SAP Replication Server 論理ページサイズ、カラム数とカラムサイズ、ストアドプロシージャの引数の数をサポートします。

次の項では、これらの機能について説明します。 国際的な環境における複写システムの設計については、『デザインガイド』の「国際的な複写システムの設計」を参照してください。

文字セットのサポート

SAP Replication Server では、SAP がサポートするすべての文字セットがサポートされ、必要に応じてデータと識別子の文字セット変換が行われます。

文字セットの変換には、以下のガイドラインが適用されます。

- すべての SAP ソフトウェアと同様、シングルバイト文字のデータとマルチバイト文字のデータ間での変換は行われません。
- テーブル名やカラム名などの識別子にマルチバイト文字や上位ビットセットのシングルバイト文字が含まれる場合は、識別子を二重引用符で囲む必要があります。
- XML テキストデータは、シングルバイトの文字セットでエンコードするか、SAP ASE と同じ文字セットでエンコードする必要があります。

文字セットの指定

文字セットは、Replication Server 設定ファイルの *rs_charset* パラメータで指定します。Replication Server 設定ファイルの記述に使用する文字セットも指定できます。使用するパラメータは *CONFIG_charset* です。

複写が正しく行われるためには、Replication Server の文字セットと、制御下にあるデータサーバの文字セットを同じにしてください。また、システムに含まれる他のすべての Replication Server の文字セットとの互換性も必要です。

文字セットの変換

Replication Server では、プライマリデータベースとレプリケートデータベースの間で、データと識別子の文字セットが変換されます。ただし、互換性のない文字セット間では文字セット変換は行われません。文字セットに互換性があっても、両方の文字セットに共通しない文字が含まれている場合、認識できない文字は疑問符 (?) に置き換えられます。

rs_config システムテーブル内の設定パラメータ **dsi_charset_convert** を使用すると、Replication Server での文字セット変換の方法をオプションとして指定できます。このパラメータは **alter connection** コマンドで設定します。

rs_get_charset システムファンクション
Replication Server では、データサーバに接続するたびに **rs_get_charset** が実行され、データサーバで使用されている文字セットが取得されます。予期した文字セットでない場合は、エラーログファイルに警告メッセージが出力されます。

参照：

- rs_get_charset (554 ページ)
- alter connection (134 ページ)

ソート順のサポート

SAP Replication Server では、ソート順 (照合順) によって、文字データと識別子の比較および並べ替えの方法が決まります。 SAP Replication Server では、バイナリソート以外のソート順も含めて、SAP がサポートするすべてのソート順をサポートしています。 バイナリソート以外のソート順は、ヨーロッパ言語での文字データと識別子を正しくソートするために必要です。

ソート順を指定するには、SAP Replication Server 設定ファイルの *RS_sortorder* パラメータを使用します。 使用している文字セットと互換性があり、SAP がサポートしているソート順であれば、どれでも指定できます。

複写を正しく行うには、使用する複写システムのすべてのソート順を同じにしてください。

rs_get_sortorder システムファンクション
SAP Replication Server では、データサーバに接続するたびに **rs_get_sortorder** が実行され、データサーバで使用されているソート順が取得されます。 予期した文字セットでない場合は、エラーログファイルに警告メッセージが出力されます。

参照：

- rs_get_sortorder (557 ページ)

メッセージ言語のサポート

SAP Replication Server では、エラーログやクライアントへのメッセージをフランス語、ドイツ語、日本語で出力できます。 言語を指定するには、SAP Replication Server 設定ファイルの *RS_language* パラメータを使用します。

SAP Replication Server がローカライズされている上記の言語の中から、使用している文字セットと互換性のある言語を指定できます。 デフォルトの言語は英語で、これは SAP がサポートするすべての文字セットと互換性があります。

ストアドプロシージャのメッセージ
rs_msgs システムテーブルには、インストール中、および RSSD を管理する SAP Replication Server ストアドプロシージャで使用される、ローカライズ済みのエラーメッセージが格納されています。 *rs_msgs* システムテーブルの詳細については、「*rs_queuemsg*」を参照してください。

ページサイズとカラムサイズのサポート強化

拡張された制限値のサポートについて説明します。

Replication Server バージョン 12.5 以降では、Adaptive Server バージョン 12.5 以降でサポートされている、制限値の拡張がサポートされます。サポートされる内容は次のとおりです。

- 論理ページサイズの選択肢：2K、4K、8K、16K
- ローサイズの拡張 (選択したページサイズの範囲内で可能)
- カラムサイズの拡張 (選択したページサイズの範囲内で可能)
- インデックスキー長の拡張
- テーブル単位のカラム数の増加
- 16KB を超えるメッセージ

Replication Server における制限値の拡張の詳細については、『Replication Server 管理ガイド 第 1 巻』を参照してください。

混合バージョンの複写システム

複写システムには、さまざまなバージョンの Replication Server または Adaptive Server を含めることができます。各システムには、それぞれ異なる問題があります。

- 複写システムドメインに Replication Server 15.5 以降がある場合は、複写システムドメインのシステムバージョンとすべてのサイトおよびルートバージョンが 12.6 以降でなければなりません。

バージョン 15.5 以降にアップグレードするには、その前に Replication Server をバージョン 12.6 以降にアップグレードし、サイトバージョンを 12.6 以降に設定して、ルートを 12.6 以降にアップグレードする必要があります。

詳細については、『Replication Server 設定ガイド』の「Replication Server のアップグレードまたはダウングレード」を参照してください。

- すべての Replication Server がバージョン 12.6 以降で、システムバージョンが 12.6 に設定されている場合、各 Replication Server はその「*サイトバージョン*」に応じた機能を使用できます。たとえば、バージョン 15.5 が動作している Replication Server では 15.2 の機能をすべて使用できますが、11.0.2 が動作している Replication Server で使用できるのは 11.0.2 の機能だけです。このようなシステムは、「*混合バージョンシステム*」と呼ばれ、各 Replication Server はそれぞれの機能をすべて使用できます。

混合バージョンシステムの制限

異なるバージョンの Replication Server 間の対話は、最も古いバージョンの機能に制限されます。

新機能に対応する情報は、古いバージョンの Replication Server では利用できないことがあります。ファンクション文字列の継承や複数の複写定義など、新しいバージョンで採用された各機能を、混合バージョン環境で使用する場合の制限事項の詳細については、マニュアルを参照してください。

混合バージョンシステム、およびサイトバージョンとシステムバージョンの設定の詳細については、使用しているプラットフォームの『Replication Server インストールガイド』、『Replication Server 設定ガイド』、『Replication Server リリースノート』を参照してください。

SAP Replication Server コマンド

複写環境を管理するには、Replication Command Language (RCL) で提供されるコマンドを使用します。

abort switch

Replication Server の処理がある程度進んでアクティブの切り替えをアボートできない場合を除いて、**switch active** コマンドをアボートする。**switch active** コマンドは、ウォームスタンバイアプリケーションでアクティブデータベースを変更します。

構文

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

パラメータ

- **logical_ds –** 論理コネクションのデータサーバの名前です。
- **logical_db –** 論理コネクションのデータベースの名前です。

例

- **例 1 –** Replication Server は、アクティブの切り替え処理を中止できない状態になっています。切り替えが終了するのを待ってから、もう一度 **switch active** コマンドを入力して、元のアクティブデータベースに戻ってください。

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

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

- **例 2 –** Replication Server は、アクティブの切り替え処理をアボートしました。アクティブデータベースは変更されません。

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

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

使用法

- **abort switch** コマンドは、**switch active** コマンドを取り消します。
- 論理コネクションを切り替え中でない場合は、エラーメッセージが返されます。

- アクティブの切り替えが正常に中止された場合、アクティブデータベースの RepAgent を再起動することが必要になる場合があります。
- **switch active** コマンドは、ある時点まで到達すると中止することはできません。その場合は、**switch active** が終了するのを待たなければなりません。その後、もう一度 **switch active** を使用すると、元のアクティブデータベースに戻ります。

パーミッション

abort switch には、"sa" パーミッションが必要です。

参照：

- switch active (456 ページ)
- admin logical_status (64 ページ)
- wait for switch (529 ページ)

activate subscription

複写定義またはパブリケーションのサブスクリプションを使用する場合に、プライマリデータベースからレプリケートデータベースへの更新の分配を開始し、サブスクリプションステータスを ACTIVE に設定します。**activate subscription** コマンドは、バルクマテリアライゼーション処理の一部、またはパブリケーションサブスクリプションのリフレッシュ処理の一部です。

構文

```
activate subscription sub_name
for {table_rep_def | function_rep_def |
publication pub_name
with primary at data_server.database}
with replicate at data_server.database
[with suspension [at active replicate only] | with catchup_queue]
```

パラメータ

- **sub_name –** アクティブにするサブスクリプションの名前です。
- **for table_rep_def –** サブスクリプションの対象となるテーブル複写定義の名前を指定します。
- **for function_rep_def –** サブスクリプションの対象となるファンクション複写定義の名前を指定します。
- **for publication pub_name –** サブスクリプションの対象となるパブリケーションの名前を指定します。
- **with primary at data_server.database –** プライマリデータのロケーションを指定します。プライマリデータベースがウォームスタンバイアプリケーションの一

部である場合、*data_server.database* は論理データサーバと論理データベースの名前になります。この句は、パブリケーション用のサブスクリプションにだけ使用します。

- **with replicate at data_server.database –** レプリケートデータのロケーションを指定します。レプリケートデータベースが論理コネクションを使用するウォームスタンバイアプリケーションの一部である場合、*data_server.database* は論理データサーバと論理データベースの名前になります。
- **with suspension –** サブスクリプションのステータスを変更した後、レプリケートデータベース用のデータサーバインタフェース (DSI) をサスペンドします。DSI がサスペンドされている間、レプリケートデータベースへの更新はステーブルキュー内に保持されます。初期データをロードして DSI をレジュームすると、更新が適用されます。ウォームスタンバイアプリケーションでこの句を使用すると、アクティブデータベースの DSI とスタンバイデータベースの DSI がサスペンドされます。
- **with suspension at active replicate only –** ウォームスタンバイアプリケーションで、アクティブデータベースの DSI はサスペンドされますが、スタンバイデータベースの DSI はサスペンドされません。
- **with catchup_queue –** キャッチアップキューを開始してプライマリテーブルの DML を格納するよう Replication Server に指示します。 サブスクリプションが VALID になる前に **validate subscription** を発行すると、キャッチアップキュー内の DML オペレーションがレプリケートテーブルに適用されます。

例

- **例 1 –** テーブル複写定義 *titles_rep* のサブスクリプション *titles_sub* をアクティブにします。ここでは、レプリケートデータベースは SYDNEY_DS.*pubs2* です。このコマンドによって、DSI がサスペンドされます。

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

- **例 2 –** ファンクション複写定義 *myproc_rep* のサブスクリプション *myproc_sub* をアクティブにします。ここでは、レプリケートデータベースは SYDNEY_DS.*pubs2* です。

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

- **例 3 –** パブリケーション *pubs2_pub* のサブスクリプション *pubs2_sub* をアクティブにします。ここでは、プライマリデータベースは TOKYO_DS.*pubs2*、レプリケートデータベースは SYDNEY_DS.*pubs2* です。

```
activate subscription pubs2_sub
 for publication pubs2_pub
```

```
 with primary at TOKYO_DS.pubs2
 with replicate at SYDNEY_DS.pubs2
```

- **例 4 –** テーブル複写定義 *titles_rep* のサブスクリプション *titles_sub* を **catchup_queue** 句を指定してアクティブにします。ここでは、レプリケートデータベースは SYDNEY_DS.*pubs2* です。 アクティブにされるプライマリテーブルの DML は、キャッチアップキューに格納されます。 サブスクリプションが VALID になるのは、キャッチアップキュー内のすべての DML オペレーションがレプリケートテーブルに適用されてからです。

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

使用法

- **activate subscription** プライマリおよびレプリケート Replication Server に対して、サブスクリプションをアクティブにする場合に使用します。サブスクリプションは、テーブル複写定義、ファンクション複写定義、データベース複写定義、またはパブリケーションに対して作成できます。
- このコマンドを使用すると、バルクマテリアライゼーション処理の 2 番めの手順が開始されます。最初の手順は、**define subscription** を使用したサブスクリプションの作成です。
- バルクマテリアライゼーションを完了するには、メディアからデータをロードし、レプリケートデータベースへのコネクションがサスペンドされている場合はレジュームして、**validate subscription** を実行します。
- **activate subscription** は、サブスクリプションを作成した Replication Server で実行します。
- **activate subscription** は、サブスクリプションのステータスを DEFINED から ACTIVE へ変更します。以降のプライマリデータサーバで行われる更新は、プライマリ Replication Server から分配されます。
- 既存のサブスクリプションを持つパブリケーションに新しいアーティクルを追加した場合は、新しいアーティクルのサブスクリプションを作成するために、新しいデータのマテリアライズを行って、パブリケーションサブスクリプションをリフレッシュしてください。

  define subscription を使用してこの処理を開始してから、**activate subscription** を使用して新しいアーティクルサブスクリプションをアクティブにします。次に、新しいアーティクルサブスクリプションのサブスクリプションデータを手動でロードし、**validate subscription** を使用して、パブリケーションサブスクリプションを確定化します。
- パブリケーションサブスクリプションをアクティブにすると、そのアーティクルサブスクリプションは、1 回に 1 つずつアクティブになるのではなく、すべて同時にアクティブになります。

- このコマンドを使用すると、複数のサイトで RSSD テーブルが修正されます。プライマリおよびレプリケート Replication Server で **check subscription** を使用して、各サイト上の影響を調べてください。
- サブスクリプションマテリアライゼーションの詳細については、『Replication Server 管理ガイド 第 1 巻』を参照してください。

with suspension 句

- **with suspension** 句を使用すると、サブスクリプションのステータスが変更された後、**activate subscription** によって DSI がサスペンドされます。これによって、サブスクリプションデータがロードされる前に、レプリケート Replication Server が複写テーブルへの更新を送信するのを防ぎます。

  データがレプリケートサイトにロードされたら、**resume connection** を実行して更新を適用してください。**with suspension** を指定しない場合は、サブスクリプションがマテリアライズされるまで、プライマリバージョンに対する更新を行わないようにする必要があります。
- データベースがウォームスタンバイアプリケーションの一部である場合、**with suspension** 句はサブスクリプションのステータスを変更してから、アクティブデータベースの DSI とスタンバイデータベースの DSI をサスペンドします。これにより、アクティブデータベースに対する更新の継続を許可する前に、両方のデータベースにデータをロードできるようになります。

  ロギングされた (たとえば、ログを指定した **bcp** を使用するか、またはアクティブデータベースでトランザクションを実行することによって) アクティブデータベースにデータをロードする場合は、スタンバイ DSI がサスペンドされないように **with suspension at active replicate only**, 句を使用してください。この場合、サブスクリプションデータはアクティブデータベースから複写されるため、スタンバイデータベースにロードする必要はありません。

with catchup_queue 句

- **catchup_queue** 句を使用するとき、マテリアライズされるプライマリテーブルでの DML オペレーションはキャッチアップキューに格納されます。 バルクマテリアライゼーションの完了後に **validate subscription** コマンドを発行すると、サブスクリプションが VALID になる前に、キャッチアップキューにあるすべての DML オペレーションがレプリケートテーブルに適用されます。
- **catchup_queue** 句を使用する場合、**with suspension** を指定したり、バルクマテリアライゼーションの実行中 (つまり、**activate subscription** コマンドと **validate subscription** コマンド実行の間の時間) に DML オペレーションを制限したりする必要はなくなりました。
- キャッチアップキュー内の DML オペレーションがレプリケートテーブルに適用されるときに、各 **insert** オペレーションが **delete** と **insert** の連続オペレー

ションに変換されます。 更新でプライマリキーが変更されると、マテリアライゼーションは失敗します。
- (プライマリ Microsoft SQL Server、DB2 UDB または Oracle データベースのみ) **direct_load** オプションでサブスクリプションが作成され、マテリアライゼーション中にプライマリテーブルに対する更新が予定される場合は、**ra_set_autocorrection** Replication Agent コマンドを有効にして、Replication Agent からダウンストリーム Replication Server にすべてのカラムの値を送信できるようにします。

パーミッション

activate subscription は、レプリケート Replication Server では “create object” パーミッション、プライマリ Replication Server では “primary subscribe” パーミッションを持つユーザが実行できます。

参照：
- check subscription (233 ページ)
- create subscription (376 ページ)
- define subscription (395 ページ)
- drop subscription (424 ページ)
- resume connection (436 ページ)
- validate subscription (525 ページ)

add partition

Replication Server でパーティションを使用できるようにします。パーティションには、ディスクパーティションまたはオペレーティングシステムファイルを使用できます。

注意： コマンド名が違っているだけで、**add partition** と **create partition** は同じです。下位互換性を保つために、**add partition** は **create partition** のエイリアスとして現在もサポートされていますが、今後は推奨されません。

構文

構文情報については、「**create partition**」を参照してください。

使用法

使用方法の情報については、「**create partition**」を参照してください。

参照：

- create partition (339 ページ)

admin auto_part_path

自動でサイズ変更可能な Replication Server パーティションに関する情報を表示します。

構文

```
admin auto_part_path
```

例

- **例 1 –** コマンドを実行する Replication Server 上で利用可能な、自動でサイズ変更可能なパーティションに関する情報を表示します。

  ```
  admin auto_part_path
  ```

  次のように表示されます。

  ```
  Part Path           Logical  Auto   Max       State
                               Expand Size
                               Size
  ------------------  -------  ------ --------  -------
  /dev/autoprt/auto1  auto_1   50     1048576   ON-LINE
  /dev/autoprt/auto2  auto_2   70     1048576   DROP-PENDING
  /dev/autoprt/auto3  auto_3   40     1048576   HAS-PARTITION
  /dev/autoprt/auto4  auto_4   80     1048576   FULL
  ```

使用法

表 4 : admin auto_part_path で出力されるカラムの説明

カラム	説明
Part Path	動的にサイズ変更可能なパーティションに対するオペレーティングシステム上の物理ロケーション。
Logical	動的にサイズ変更可能なパーティションに割り当てる論理名。
Auto Expand Size	Replication Server が自動で作成する各パーティションファイルに対して設定したメガバイト単位のサイズ。
Max Size	論理パーティションパスに割り当てられている動的にサイズ変更可能なパーティションのすべてのパーティションファイルに対して設定した、メガバイト単位の合計サイズ。

カラム	説明
State	動的にサイズ変更可能なパーティションのステータス。 • ON-LINE - パーティションは使用可能 • DROP-PENDING - Replication Server によって削除されるパーティション • HAS-PARTITION - パーティションパスの下に存在するパーティション • FULL - パーティションパスがいっぱいで、Replication Server はこのパーティションに新しいパーティションファイルを追加できない。 使用可能なディスク領域がある場合は、**alter auto partition path** を使用して *max_size* の値を増やすことができる。 あるいは、**create auto partition path** を使用して、十分なディスク容量のある別の場所に、自動でサイズ変更可能なパーティションを新しく作成することができる。

パーミッション

このコマンドは、すべてのユーザが実行できます。

参照：

- alter auto partition path (133 ページ)
- create auto partition path (284 ページ)
- drop auto partition path (405 ページ)
- rs_helppartition (714 ページ)
- admin disk_space (57 ページ)

admin config

Replication Server のすべての設定パラメータを表示します。

構文

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

注意： 設定値が 255 バイトよりも長い場合、**admin config** コマンドには最初の 251 バイトと省略記号 (...) が表示されます。

パラメータ

- **"connection" –** コネクションの設定パラメータを表示します。

- **logical_connection –** 論理コネクションの設定パラメータを表示します。
- **"table" –** 問い合わせるテーブルの名前を指定します。最大 200 文字までの文字列である *table_name* と一緒に使用します。*table_owner* は、テーブル名のオプション修飾子であり、テーブルの所有者を表します。

  テーブル名を指定しない場合、**admin config** はすべてのテーブルの設定パラメータを表示します。
- **data_server, database –** 問い合わせ対象のデータサーバとデータベースです。

  コネクションに関連する設定パラメータを表示する場合、サーバはデータサーバである必要があり、*database* を指定する必要があります。ルートに関連する設定パラメータを表示する場合、サーバは Replication Server である必要があります。*database* は指定できません。
- **“route” –** ルートの設定パラメータを表示します。
- **repserver –** ルートのターゲット Replication Server を指定します。
- **configuration_name –** 値とステータスを表示する設定パラメータです。

例

- **例 1 –** Replication Server のグローバル設定パラメータをすべて表示します。

```
admin config
go
```

```
ConfigurationConfigRunDefault
ValueValueValue
-------------------------------------
cm_max_connections656564
dsi_cmd_batch_size819381938192

LegalValuesDatatypeStatus
-------------------------------------------
range: 1,2147483647integerRestart required
range: 1,2147483647integerRestart required
(2 rows affected)
```

- **例 2 –** Replication Server TOKYO_RS へのルートの設定パラメータをすべて表示します。

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

- **例 3 –** pdb1 へのコネクションの設定パラメータをすべて表示します。

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

```
ConfigurationConfigRunDefault
ValueValueValue
-------------------------------------
dsi_cmd_batch_sizeNULLNULL8192

LegalValuesDatatypeStatus
```

```
-----------------------------------------
range: 1,2147483647integerConnection/Route
restart required
(1 row affected)
```

- **例 4 – dsi_command_convert** を使用して **d2none** を SYDNEY_DS データサーバの pubs2 データベースの *tb1* テーブルに設定した後で、すべての設定パラメータを表示します。

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

admin config を使用すると、次のように表示されます。

```
ConfigurationConfigRunDefault
ValueValueValue
-------------------------------------
dsi_compile_enable<server default><server default>on
dsi_command_convertd2noned2nonenone

Legal ValuesDatatype
-----------------------------------------------------------------
list: on,ofstring
list: none, i2none, d2none, u2none, i2di, u2di, t2none string

StatusTable
-------------------------
Restart not requireddbo.tb1
Restart not requireddbo.tb1

(2 rows affected)
```

- **例 5 – dsi_command_convert** に対する設定パラメータのみを表示します。表示されるのは、**dsi_command_convert** を SYDNEY_DS データサーバの pubs2 データベースの *tb1* テーブルで使用した後です。

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

admin config を使用すると、次のように表示されます。

```
ConfigurationConfigRunDefault
ValueValueValue
-------------------------------------
dsi_command_convertd2noned2nonenone

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

StatusTable
-------------------------
Restart not requireddbo.tb1

(1 row affected)
```

使用法

- **admin config** は、Replication Server のカスタマイズとチューニングに使用するさまざまなタイプの設定パラメータ (サーバ、コネクション、論理コネクション、ルートなど) を取得するときに使用します。
- Data Server Interface (DSI) および Replication Agent の設定パラメータを表示するには、**admin config, "connection"** を使用します。

注意： **admin config, "connection"** コマンドは、ディストリビュータおよびスステーブルキューマネージャ (SQM) のスレッド設定パラメータを表示しません。

Replication Server の各パラメータの設定とチューニングの詳細については、『Replication Server 管理ガイド 第 1 巻』と『Replication Server 管理ガイド 第 2 巻』を参照してください。

admin disk_space

Replication Server がアクセスする各ディスクパーティションの使用状況を表示します。

構文

```
admin disk_space
```

例

- **例 1 –** ディスクパーティションに関する情報を表示します。

```
admin disk_space
```

```
PartitionLogicalPart.Id
---------------------------
/dev/hdb2partition_1101

TotalSegsUsedSegsState
---------------------------
203ON-LINE
```

使用法

表 5 : admin disk_space で出力されるカラムの説明

カラム	説明
Partition	Replication Server が使用しているデバイス名。
Logical	パーティションに割り当てられた論理名。

カラム	説明
Part.Id	パーティションの ID。
Total Segs	パーティションの 1MB の合計セグメント数。
Used Segs	Replication Server が現在使用しているセグメントの合計数。
State	このデバイスのステータス。 • ON-LINE - デバイスは正常。 • OFF-LINE - デバイスが見つからない。 • DROPPED - デバイスは削除されているが、まだ残っている (あるキューがそれを使用している)。 • AUTO - デバイスは自動でサイズ変更可能。『Replication Server 管理ガイド第 1 巻』の「自動でサイズ変更可能なパーティション」を参照。

パーミッション

このコマンドは、すべてのユーザが実行できます。

参照：

- admin who (107 ページ)
- alter partition (196 ページ)
- create partition (339 ページ)
- drop partition (417 ページ)
- admin auto_part_path (53 ページ)
- alter auto partition path (133 ページ)
- create auto partition path (284 ページ)
- drop auto partition path (405 ページ)

admin echo

ユーザが入力した文字列を返します。

構文

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

パラメータ

- **character_string –** ユーザが入力する文字列です。

- **with_log –** ユーザが入力した文字列を Replication Server ログに書き込みます。

例

- **例 1 –** Replication Server は、ユーザが入力した文字列 “hello” を返します。

```
admin echo, hello
```

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

- **例 2 –** Replication Server は “Hello world!” を返し、“Hello world!” を Replication Server ログに書き込みます。

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

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

使用法

- ローカル Replication Server が稼働しているかどうかを調べるには、**admin echo** を使用します。
- このコマンドは、ネットワークのエコーとして機能するわけではありません。引数を指定しない場合は、何も返されません。

パーミッション

このコマンドは、すべてのユーザが実行できます。

admin get_generation

プライマリデータベースの世代番号を取得する。

構文

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

パラメータ

- **data_server –** プライマリデータベースのあるデータサーバです。
- **database –** 世帯番号を取得するデータベースです。

例

- **例 1 –**

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

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

使用法

- データベースの世代番号は、ログレコード用に RepAgent が生成するオリジンキュー ID の最初の 2 バイトで表されます。世代番号は、ログ転送言語 (LTL: Log Transfer Language) の **distribute** コマンドで使用するパラメータの 1 つです。
- プライマリデータベースをロードした後は、世代番号の数をインクリメントする (増やしていく) 必要があります。そうすることで、ロード後に適用されたトランザクションが、Replication Server で (重複として) 無視されるのを防ぎます。
- 世代番号の数を増やすには、Adaptive Server データベースで Adaptive Server の **dbcc settrunc** を実行します。

パーミッション

このコマンドは、すべてのユーザが実行できます。

参照：

- dbcc settrunc (602 ページ)

admin health

Replication Server のステータスを表示します。

構文

```
admin health
```

例

- **例 1 –** Replication Server のステータスを表示します。

```
admin health
```

```

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

使用法

表 6 : admin health で出力されるカラムの説明

カラム	説明
Mode	Replication Server のリカバリに関するステータス。 値は次のいずれかになる。 • NORMAL - Replication Server は正常に稼働している。 • REBUILDING - Replication Server が **rebuild queues** コマンドを実行中であることを示す一時的なステータス。 • RECOVERY - Replication Server がスタンドアロンモードにあり、**rebuild queues** コマンドが実行されている。 • STANDALONE - Replication Server はいかなるコネクションも開始または受け付けない。**-M** オプションを付けて Replication Server を起動したときだけこのステータスとなる。スタンドアロンモードを終了するには、Replication Server を停止し、**-M** オプションを付けずに再起動する。
Quiesce (静止)	Replication Server がクワイスされているかどうかを示す。 次のいずれかになる。 • TRUE - Replication Server はクワイスされている。つまり、すべてのメッセージがフラッシュされている。 • FALSE - Replication Server はクワイスされていない。
Status	Replication Server 全体のステータス。 値は次のいずれかになる。 • HEALTHY - すべてのスレッドが正常に実行されている。 • SUSPECT - あるスレッドが停止し、起動が必要である。 または、スレッドが "接続中 (connecting)" の状態である。 "接続中" の状態とは、Replication Server が接続しているサーバが使用不可能で問題が発生しているか、または Replication Server はすぐに正常に接続するが、それまで一時的にサスペクト (suspect) 状態であることを示す。 実行されていないスレッドを確認するには、**admin who_is_down** を実行する。

カラム	説明

Loss Status	データ消失ステータス:
	• SUSPECT - キューでデータが失われた可能性が疑われる。
	• DETECTING - キューのデータ消失を確認中。
	• IGNORING - **ignore loss** コマンドが実行されているため、キューのデータ消失は無視される。
	• NO LOSS - キューではデータ消失は検出されていない。
	Replication Server のログにはこれらのステータスに対応した項目がある。 例は次のとおり。
	• DETECTING
	`I. 2012/11/05 21:46:08. Checking loss for NY_RS.ERSSD2 from BEJ_RS.ERSSD1`
	`Date: Nov 5 2012 2:46:08:576PM`
	`qid= 0000000000018ddb0000142f00290000142f00140000a10000f362ed0000000000000005`
	`I. 2012/11/05 21:46:08. Checking loss for SYDNEY_DS.rdb1 from LONDON_DS.pdb1`
	`Date: Nov 5 2012 2:46:56:583PM`
	`qid= 000000000000baf100000b58008800000b5800860000a10000f39b2f0000000000000003`
	NY_RS および BEJ_RS がプライマリであり、それぞれが Replication Servers を複写する場合、SYDNEY_DS.rdb1 がレプリケートデータサーバであり、LONDON_DS.pdb1 はプライマリデータサーバおよびデータベースである。
	• SUSPECT
	`Replication Server has identified the possibility of data loss for NY_RS.ERSSD2 from BEJ_RS.ERSSD1. Confirmation of data consistency is needed for validation.`
	SUSPECT ステータスが示されている場合は、プライマリとレプリケートデータベースの間の整合性を確認し、データ消失がないかどうかを確認する必要がある。 データ消失がある場合、データベースの再同期を行うなどの方法で手動で消失を修正することが必要な場合がある。 状況に応じて次のようになる。
	• Adaptive Server - 『Replication Server 管理ガイド第 2 巻』>「複写システムリカバリ」>「Adaptive Server のレプリケートデータベースの再同期」を参照。
	• Oracle - 『Replication Server 異機種間複写ガイド』>「Oracle レプリケートデータベースの再同期」を参照。
	データ消失がない場合は、データ消失を無視するように指示できる。
	`ignore loss from LONDON_DS.pdb1 to SYDNEY_DS.rdb1`

カラム	説明
	• IGNORING `I. 2012/11/05 21:54:46. Ignoring loss for NY_RS.ERSSD2 from BEJ_RS.ERSSD1`

パーミッション

このコマンドは、すべてのユーザが実行できます。

参照：

- admin quiesce_check (66 ページ)
- admin quiesce_force_rsi (68 ページ)
- admin who (107 ページ)
- admin who_is_down (126 ページ)
- admin who_is_up (127 ページ)
- rebuild queues (434 ページ)

admin log_name

現在のログファイルのパスを表示します。

構文

```
admin log_name
```

例

- **例 1 –** 現在の Replication Server のログファイルへのパス名を表示します。

```
admin log_name
```

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

使用法

-e オプションを指定して Replication Server を起動し、エラーログにフルパス名を指定した場合、**admin log_name** はフルパスを返します。相対パス名を指定した場合、**admin log_name** は Replication Server の現在の作業ディレクトリの相対パス名を返します。

パーミッション

このコマンドは、すべてのユーザが実行できます。

参照：

- admin set_log_name (73 ページ)

admin logical_status

論理コネクションのステータス情報を表示します。

構文

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

パラメータ

- **logical_ds –** 論理コネクションのデータサーバの名前です。
- **logical_db –** 論理コネクションのデータベースの名前です。

例

- **例 1 –** この例は、正常なアクティブ状態の LDS.*pubs2* 論理コネクションの出力を示しています。現在のアクティブデータベースは、TOKYO_DS データベースサーバの *pubs2* データベースです。スタンバイデータベースは、SYDNEY_DS データベースサーバの *pubs2* データベースです。TOKYO_RS Replication Server は、論理コネクションを管理しています。両方の物理コネクションがアクティブになっています。特別なオペレーションは何も実行されていません。

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

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

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

使用法

- **admin logical_status** は、ウォームスタンバイアプリケーションでアクティブデータベースとスタンバイデータベースの論理コネクションのステータスを調べるときに使用します。
- *logical_ds* と *logical_db* を指定しない場合、**admin logical_status** は、この Replication Server によって制御されるすべての論理コネクションについての情報を表示します。
- 表 7 に、出力カラムを示します。

表 7 : admin logical_status で出力されるカラムの説明

カラム	説明
Logical Connection Name	論理コネクションの DBID (データベース ID)、論理データサーバ、データベース名。
Active Connection Name	現在のアクティブデータベースの DBID、データサーバ、データベース名。
Active Connection State	アクティブコネクションのステータスの説明。active、suspended、suspended by error のいずれか。
Standby Connection Name	現在のスタンバイデータベースの DBID、データサーバ、データベース名。
Standby Connection State	スタンバイコネクションのステータスの説明。active、suspended、suspended by error、waiting for marker のいずれか。
Controller RS	論理データベース、アクティブデータベース、スタンバイデータベースを管理する Replication Server の RSID (Replication Server ID) と名前。
Operation in Progress	実行中のオペレーションの説明。None、Switch Active、Create Standby のいずれか。
State of Operation in Progress	オペレーションの現在の段階。
Spid	オペレーションを実行しているサーバスレッドのプロセス ID。

パーミッション

このコマンドは、すべてのユーザが実行できます。

参照：

- abort switch (47 ページ)
- admin sqm_readers (87 ページ)
- admin who (107 ページ)
- create connection (287 ページ)
- create logical connection (338 ページ)
- switch active (456 ページ)
- wait for create standby (528 ページ)
- wait for switch (529 ページ)

admin pid

Replication Server のプロセス ID を表示します。

構文

```
admin pid
```

例

- **例 1 –** 現在の Replication Server のプロセス ID は 12032 です。

```
admin pid
```

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

使用法

Replication Server のプロセス ID を表示します。

パーミッション

このコマンドは、すべてのユーザが実行できます。

admin quiesce_check

Replication Server のキューがクワイスされているかどうかを調べます。

構文

```
admin quiesce_check
```

例

- **例 1 –** TOKYO_RS という Replication Server が、クワイスされています。

```
admin quiesce_check
```

```
Replication Server TOKYO_RS is quiesced
```

- **例 2 –** 次のメッセージは、キュー 103:1 に読み込まれていないメッセージがあるため、システムがクワイス状態ではないことを示します。レポートされている読み込みロケーション (30.2) と書き込みロケーション (32.1) は、キューに書き込まれているブロックが読み込まれたブロックよりも多いことを示します。これ以上ブロックが書き込まれない場合、システムがクワイスされる前に、読み込みロケーションはセグメント 32、ブロック 2 になっていなければなりません。

```
admin quiesce_check
```

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

使用法

- **admin quiesce_check** は、Replication Server がクワイスされているかどうかを調べます。
- Replication Server は、次の場合にクワイスされます。
  - サブスクリプションマテリアライゼーションキューがない。
  - Replication Server ですべてのキューにあるすべてのメッセージが読み込まれ、処理されている。
  - 未配信のコミットされたトランザクションを含むインバウンド (RepAgent) キューがない。
  - RSI キューのすべてのメッセージが送信先 Replication Server に送信され、受信確認を受け取っている。
  - DSI キューのすべてのメッセージが適用され、データサーバから受信確認を受け取っている。

パーミッション

このコマンドは、すべてのユーザが実行できます。

参照：

- admin quiesce_force_rsi (68 ページ)
- suspend connection (452 ページ)
- suspend log transfer (454 ページ)

admin quiesce_force_rsi

Replication Server がクワイス状態かどうかを確認し、Replication Server に RSI キューのメッセージを配信し、受信確認を取得することを強制します。

構文

```
admin quiesce_force_rsi
```

例

- **例 1 –** TOKYO_RS という Replication Server が、クワイスされています。

  ```
  admin quiesce_force_rsi
  ```

  ```
  Replication Server TOKYO_RS is quiesced
  ```

- **例 2 –** 次のメッセージは、キュー 103:1 に読み込まれていないメッセージがあるため、システムがクワイス状態ではないことを示します。レポートされている書き込みロケーション (32.1) と読み込みロケーション (30.2) は、キューに書き込まれているブロックが読み込まれたブロックよりも多いことを示します。

  ```
  admin quiesce_force_rsi
  ```

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

使用法

- **suspend log transfer from all** は、**admin quiesce_force_rsi** を実行する前に実行します。これにより、RepAgent が Replication Server に接続できないようにします。
- このコマンドは、すべてのインバウンドキューがクワイス状態になってから実行します。
- Replication Server は、次の場合にクワイスされます。
  - サブスクリプションマテリアライゼーションキューがない。
  - Replication Server ですべてのキューにあるすべてのメッセージが読み込まれている。
  - 未配信のコミットされたトランザクションを含むインバウンド (RepAgent) キューがない。
  - RSI キューのすべてのメッセージが送信先 Replication Server に送信され、受信確認を受け取っている。
  - DSI キューのすべてのメッセージが適用され、データサーバから受信確認を受け取っている。
- 通常、RSI はそのキューを 30 秒ごとに空にします。

パーミッション

このコマンドは、すべてのユーザが実行できます。

参照：

- admin quiesce_check (66 ページ)
- suspend connection (452 ページ)
- suspend log transfer (454 ページ)

admin rssd_name

RSSD のデータサーバとデータベースの名前を表示します。

構文

```
admin rssd_name
```

例

- **例 1 –** この例では、データサーバの名前は TOKYO_DS で、RSSD の名前は TOKYO_RSSD です。

```
admin rssd_name
```

```
RSSDDataserverRSSDDatabase
--------------------------
TOKYO_DSTOKYO_RSSD
```

使用法

RSSD のデータサーバとデータベースの名前を表示します。

パーミッション

このコマンドは、すべてのユーザが実行できます。

admin schedule

Replication Server 内のタスクスケジュールに関する情報を表示します。

構文

admin "schedule"[, '*sched_name*']

パラメータ

- **'sched_name'** – 表示するスケジュールの名前です。

例

- **例 1 – schedule1** というスケジュールを表示するには、次のように入力します。

```
admin "schedule", 'schedule1'
```

出力は次のようになります。

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

使用法

"schedule" 句は二重引用符で囲む必要があります。これは **schedule** が Replication Server のキーワードであるためです。

スケジュール名を指定せずに **admin "schedule"** のみを実行すると、Replication Server の既存のすべてのスケジュールに関する情報が表示されます。

パーミッション

admin "schedule" には、"sa" パーミッションが必要です。

参照：

- alter schedule (221 ページ)
- drop schedule (423 ページ)
- create schedule (373 ページ)

admin security_property

サポートされているネットワークベースセキュリティメカニズムとネットワークベースセキュリティサービスに関する情報を表示します。

構文

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

パラメータ

- **mechanism_name –** サポートされているネットワークベースセキュリティメカニズムです。

例

- **例 1 –**

```
admin security_property
```

```
Mechanism  Feature         Supported
---------  ---------      -------------------------
DCE        Unified Login   yes
DCE        Confidentiality yes
DCE        Integrity       no
...
```

使用法

- オプションを指定しないで実行した場合は、デフォルトのセキュリティメカニズムの名前、そのメカニズムに対して使用可能なセキュリティサービス、その使用可能なサービスがユーザのサイトでサポートされているかどうかが表示されます。
- **admin security_property** を実行するには、現在の Replication Server でネットワークベースセキュリティを有効にします。これを行うには、**configure replication server** を使用して **use_security_services** パラメータを on に設定してください。
- このコマンドは、ExpressConnect for SAP HANA データベースなどの ASE 以外および IQ 以外のコネクタのセキュリティ設定の管理には使用できません。 これらのコネクタのセキュリティ設定の管理の詳細は、『Replication Server 異機種間複写ガイド』を参照してください。

パーミッション

このコマンドは、すべてのユーザが実行できます。

参照：

- admin security_setting (72 ページ)
- alter connection (134 ページ)
- alter route (213 ページ)
- configure replication server (238 ページ)
- create connection (287 ページ)
- create route (368 ページ)
- set proxy (448 ページ)

admin security_setting

Replication Server のネットワークベースセキュリティのパラメータとその値を表示します。

構文

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

パラメータ

- **rs_idserver –** 現在の Replication Server が接続する ID サーバです。
- **rs_server –** 現在の Replication Server が接続する Replication Server です。
- **data_server –** 現在の Replication Server が接続するターゲットデータベースのデータサーバです。
- **database –** 現在の Replication Server が接続するターゲットデータベースです。

例

- **例 1 –**

```
admin security_setting
```

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

使用法

- **admin security_setting** を実行するには、現在の Replication Server でネットワークベースセキュリティを有効にします。これを行うには、**configure replication server** を使用して **use_security_services** パラメータを on に設定してください。
- オプションを使用しないで **admin security_setting** を実行した場合は、**configure replication server** を使用して設定されたデフォルト値が表示されます。
- このコマンドは、ExpressConnect for HANA データベースなどの ASE 以外および IQ 以外のコネクタのセキュリティ設定の管理には使用できません。 これらのコネクタのセキュリティ設定の管理の詳細は、『Replication Server 異機種間複写ガイド』を参照してください。

パーミッション

このコマンドは、すべてのユーザが実行できます。

参照：

- admin security_property (70 ページ)
- alter connection (134 ページ)
- alter route (213 ページ)
- configure replication server (238 ページ)
- create connection (287 ページ)
- create route (368 ページ)
- set proxy (448 ページ)

admin set_log_name

Replication Server の既存のログファイルをクローズし、新しいログファイルをオープンします。

構文

```
admin set_log_name, log_file
```

パラメータ

- **log_file –** 新しいログファイルの名前です。

例

- **例 1 –** SYDNEY_RS.log という新しいログファイルをオープンします。**admin log_name** コマンドを使用すると、パスとログファイル名を確認できます。

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

使用法

- このコマンドが失敗した場合は、元のログファイルがオープンされたままになります。
- Replication Server が再起動される場合は、このコマンドラインで指定したログファイルが使用されます。コマンドラインに名前の指定がないときは、デフォルトのログファイル名が使用されます。

- ログファイル名に文字と数字以外を入力する場合は、そのファイル名を引用符で囲んでください。たとえば、ログファイル名にピリオド (.) が含まれている場合は、前述の例のように入力します。
- **admin set_log_name** は、ユーザが入力した名前を表示します。利便性を高めるために、絶対パス名を入力してください。

パーミッション

このコマンドは、すべてのユーザが実行できます。

参照：

- admin log_name (63 ページ)

admin show_connection_profiles

SAP Replication Server で定義されている各プロファイルのプロファイル名、バージョン、コメントをリストします。

構文

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

パラメータ

- **match_string** – 表示される接続プロファイルをフィルタします。名前にオプションで指定した文字列が含まれている接続プロファイルのみが表示されます。

例

- **例 1** – Replication Server で現在定義されているすべての接続プロファイルの名前をリスト表示します。

```
admin show_connection_profiles
go
```

```
Profile Name     Version   Comments
------------     -------  ----------------------------
rs_ase_to_db2    standard  Standard ASE to DB2 replication
                           connection profile.
rs_ase_to_udb    standard  Standard ASE to UDB replication
                           connection profile.
rs_ase_to_oracle standard  Standard ASE to Oracle replication
                           connection profile.
rs_ase_to_msss   standard Standard ASE to Microsoft SQL Server
                          replication connection profile.
rs_ase_to_ase    standard Standard ASE to ASE replication
connection profile.
```

```
rs_ase_to_iq     standard  Standard ASE to Sybase IQ replication
                           connection profile.
rs_ase_to_hanadb     ech              ASE to HanaDB replication
connection
                                      using Express Connector for
HanaDB.
rs_db2_to_msss   standard  Standard DB2 to Microsoft SQL Server
                           replication connection profile.
rs_db2_to_oracle standard  Standard DB2 to Oracle replication
                           connection profile.
rs_db2_to_udb    standard  Standard  DB2 to UDB replication
                           connection profile.
rs_db2_to_ase    standard  Standard DB2 to ASE replication
                           connection profile.
rs_oracle_to_db2 standard  Standard Oracle to DB2 replication
                           connection profile.
rs_oracle_to_udb standard  Standard Oracle to UDB replication
                           connection profile.
rs_oracle_to_msss standard Standard Oracle to Microsoft SQL
                          Server replication connection profile.
rs_oracle_to_ase standard  Standard Oracle to ASE replication
                           connection profile.

rs_oracle_to_iq   standard Standard Oracle to IQ replication

                                      connection profile. Oracle to
rs_oracle_to_hanadb          ech
                                     HanaDB replication using
connection
                                     Express Connector for
HanaDB.
rs_msss_to_db2  standard Standard Microsoft SQLServer to DB2
                         replication connection profile.
rs_msss_to_oracle standard Standard Microsoft SQLServer to
                           Oracle replication connection profile.
rs_msss_to_udb  standard  Standard Microsoft SQL Server to UDB
                          replication connection profile.
rs_msss_to_ase  standard  Standard MicrosoftSQL Server to ASE
                          replication connection profile.
rs_msss_to_hanadb          ech     Microsoft SQLServer to HanaDB
replication
                                      connection using Express
                                    Connector for HanaDB.
rs_udb_to_db2  standard Standard udb to db2 replication
                          connection profile.
rs_udb_to_msss standard Standard UDB to Microsoft SQL Server
                          replication connection profile.
rs_udb_to_oracle standard Standard UDB to Oracle
                          replication connection profile.
rs_udb_to_ase  standard Standard UDB to ASE replication
                          connection profile.
rs_udb_to_hanadb            ech      UDB to HanaDB replication
connection
                                      using Express Connector for
HanaDB.
rs_db2_to_db2 standard  Standard  DB2 to DB2 replication
                        connection profile.
```

```
rs_oracle_to_oraclestandard Standard Oracle to Oracle
replication connection profile.
rs_udb_to_udb standard Standard UDB to UDB replication
connection profile.
rs_msss_to_msss  standard Standard Microsoft SQLServer to
Microsoft SQLServer replication
connection profile.
rs_ase_to_oracle             eco      ASE to Oracle replication
connection
                                 profile using Express Connect for
Oracle.
rs_db2_to_oracle             eco      DB2 to Oracle replication
connection
                                 profile using Express Connect for
Oracle.
rs_msss_to_oracle           eco     Microsoft SQLServer to Oracle
replication
                                  connection profile using Express
Connect for Oracle
rs_oracle_to_oracle         eco     Oracle to Oracle replication
connection
                                 profile using Express Connect for
Oracle.
rs_udb_to_oracle             eco      UDB to Oracle replication
connection profile
                                  using Express Connect for Oracle.
rs_rs_to_oracle_ra          standard Standard RS to RA direct load
                                          connection profile.
rs_rs_to_udb_ra              standard Connection profile for
replication from
                                          IBM DB2 LUW (UDB) using
Replication Agent
                                   for direct load materialization.
rs_rs_to_msss_ra             standard Connection profile for
replication from
                                          Microsoft SQL Server using
Replication Agent
                                 for direct load materialization.

(39 rows affected)
```

- **例 2 –** 接続プロファイル名に文字列 "oracle" を含み、SAP Replication Server で現在定義されている、すべての接続プロファイルの名前をリスト表示します。

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

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

                                  replication connection profile.
rs_oracle_to_ases tandard Standard Oracle to ASE replication
```

```
                                        connection profile.
rs_oracle_to_iq   standard Standard Oracle to IQ
replication connectionprofile.
rs_oracle_to_hanadb   ech      Oracle to HanaDB replication
connection
                            using Express Connector for HanaDB.

rs_ase_to_oracle eco      ASE to Oracle replication connection
                               profile using Express Connect for
Oracle.
rs_db2_to_oracle eco      DB2 to Oracle replication connection
profile
                                  using Express Connect for Oracle.
rs_msss_to_oracle    eco      Microsoft SQLServer to Oracle
replication
                                  connection profile using Express
Connect
                                       for Oracle.
rs_oracle_to_oracle         eco     Oracle to Oracle replication
connection
                                  profile using Express Connect for
Oracle.
rs_udb_to_oracle             eco      UDB to Oracle replication
connection profile
                                  using Express Connect for Oracle.
rs_rs_to_oracle_ra         standard Standard RS to RA direct load
                                        connection profile.
```

使用法

using profile オプションを使用してコネクションを作成するには、**admin show_connection_profiles** を使用して使用可能なプロファイルの名前とバージョンを指定します。

参照：

- create connection using profile (294 ページ)

admin show_connections

Replication Server からデータサーバへのすべてのコネクション、または Replication Server から他の Replication Server へのすべてのコネクションに関する情報を表示します。

構文

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

パラメータ

- **primary –** すべてのプライマリコネクションに関する情報を表示します。
- **replicate –** すべての複写接続に関する情報を表示します。
- **logical –** すべての論理コネクションに関する情報を表示します。

例

- **例 1 –** この Replication Server のコネクションデータを表示します。

```
admin show_connections
```

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

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

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

- **例 2 –** プライマリデータベースへのすべてのコネクションを表示します。たとえば、SALES_DS データサーバのプライマリデータベースを制御する Replication Server で、次のように入力します。

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

次のように表示されます。

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

SALES_DS.pdb は、コネクション名がデータサーバとデータベース名の組み合わせに一致するため、Replication Server と SALES_DS データサーバの `pdb` データベースの間のデフォルトのコネクションになります。

SALES_DS.pdb は、コネクション名がデータサーバとデータベース名の組み合わせに一致するため、Replication Server と SALES_DS データサーバの pdb データベースの間のデフォルトのコネクションになります。

- **例 3 –** レプリケートデータベースへのすべてのコネクションを表示します。たとえば、FINANCE_DS データサーバと NY_DS データサーバでレプリケートデータベースを制御する Replication Server で、次のように入力します。

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

次のように表示されます。

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

FINANCE_DS.fin_rdb は、コネクション名がデータサーバとデータベース名の組み合わせに一致するため、Replication Server と FINANCE_DS データサーバの fin_rdb データベースの間のデフォルトのコネクションになります。

NY_DS.ny_db_conn2 は、コネクション名がデータサーバとデータベース名の組み合わせに一致しないため、Replication Server と NY_DS データサーバの ny_rdb データベースの間の代替コネクションになります。

- **例 4 –** 論理データベースへのすべてのコネクションを表示します。

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

次のように表示されます。

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

ここで WS_DS.ws_db はデフォルトの論理コネクションとなり、WS_DS.ws_db1 は代替論理コネクションとなります。

使用法

- このコマンドは、デフォルトおよび代替データベースコネクションと、現在の Replication Server からのルートについての情報を表示します。
- 表 8 に、このコマンドの出力を示します。

表 8 : admin show_connections で出力されるカラムの説明

カラム	説明
Connection Name	この Replication Server から始まるデフォルトの、また代替のプライマリまたは複写接続の名前です。

カラム	説明
Server	この Replication Server の接続先となる、データサーバまたは Replication Server の名前。
User	このクライアントのログイン名。
Database	この Replication Server が接続しているデータベースの名前 (ルートには null)。
State	このコネクションのステータス。
Owner	スレッドの所有者を示す。値は次のいずれかになる。 DSI - データサーバインタフェース (データベースへの)。 RSI - Replication Server インタフェース (Replication Server への)。
Spid	このスレッドのユニークな識別子。
connection state	値は次のいずれかになる。 • *active* - コネクションは使用中である。 • *already_faded_out* - コネクションは所有されており、すでにクローズされている。 • *being_faded_out* - コネクションは所有されており、現在クローズ中である。 • *closed* - クローズされているコネクションは、どのスレッドにも所有されていない。 • *connecting* - サーバに接続中である。 • *free* - コネクションはオープンされているが、誰にも所有されていない。 • *idle* - コネクションは所有されているが、使用されていない。 • *limbo* - コネクションハンドルは状態推移中である。 • *total* - コネクションの合計数。
number	このタイプのコネクションの数。
comments	*connection state* フィールドの説明。

パーミッション

このコマンドは、すべてのユーザが実行できます。

参照：

- alter connection (134 ページ)
- alter logical connection (191 ページ)
- alter route (213 ページ)
- create connection (287 ページ)
- create logical connection (338 ページ)

- create route (368 ページ)
- drop connection (407 ページ)
- drop logical connection (416 ページ)
- drop route (421 ページ)
- resume connection (436 ページ)
- suspend connection (452 ページ)

admin show_function_classes

既存のファンクション文字列クラスとその親クラスの名前を表示して、継承のレベル番号を示します。

構文

```
admin show_function_classes
```

例

- **例 1 –**

```
admin show_function_classes
```

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

使用法

レベル 0 は *rs_default_function_class* などの基本クラス、レベル 1 は基本クラスから継承する派生クラスなどとなっています。

パーミッション

このコマンドは、すべてのユーザが実行できます。

参照：

- alter connection (134 ページ)
- alter function string class (190 ページ)
- create connection (287 ページ)
- create function (310 ページ)
- create function string (318 ページ)

- create function string class (334 ページ)
- drop function string class (414 ページ)
- move primary (432 ページ)

admin show_principal_name

Replication Server プリンシパル名を表示します。

構文

```
admin show_principal_name
```

例

- **例 1 –** Replication Server プリンシパル名を表示します。

  ```
  admin show_principal_name
  go
  ```

  結果は次のようになります。

  ```

  Principal Name
  ----------------
  PRS1_princ
  ```

パーミッション

このコマンドは、すべてのユーザが実行できます。

admin show_route_versions

Replication Server で開始するルートと、Replication Server で終了するルートのバージョン番号を表示します。

構文

```
admin show_route_versions
```

例

- **例 1 –** この例では、repserver_1510.repserver_1500 のルートバージョンは 15.0.0 です。

  ```
  admin show_route_versions
  ```

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

使用法

- ルートのバージョンとは、送信元 Replication Server と送信先 Replication Server の最も古いサイトバージョンのことです。ルートバージョンが最も古いサイトバージョンよりも低い場合は、ルートアップグレードを実行する必要があります。
- バージョン番号によって、そのルートで使用できる混合バージョン環境の機能セットが決まります。
- **admin show_route_versions** は、送信元 Replication Server の名前、送信先 Replication Server の名前、ルートのバージョンをルートごとに表示します。

パーミッション

このコマンドは、すべてのユーザが実行できます。

参照：

- admin show_site_version (83 ページ)
- sysadmin fast_route_upgrade (482 ページ)

admin show_site_version

Replication Server のサイトバージョンを表示します。

構文

```
admin show_site_version
```

例

- **例 1 –** Replication Server のサイトバージョンが 15.1.0 であることを示します。

  ```
  admin show_site_version
  ```

  次のように表示されます。

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

- **例 2 – sysadmin site_version** を使用してサイトバージョンを 1571100 から 1571200 にアップグレードしようとして失敗した場合は、**admin show_site_version** を使用してステータス情報を表示します。

```
admin show_site_version
```

次のように表示されます。

```
Site Version Status
------------
-------------------------------------------------------------------
-------
1571200 The current site_version is '1571200', but eRSSD upgrade
experienced failure.
Please set site_version to '1571200' again to complete the
process.
```

使用法

Replication Server のサイトバージョンを表示します。サイトバージョンによって、使用できる Replication Server の機能が決まります。サイトバージョンを設定した後は、以前のリリースにダウングレードすることはできません。

パーミッション

このコマンドは、すべてのユーザが実行できます。

参照：

- sysadmin site_version (502 ページ)

admin sqm_process_time

Replication Server でキュー内の残りのデータの処理に要する推定時間を表示します。

構文

```
admin sqm_process_time,
      {data_server, database, q_type |
      q_number, q_type |
      replication_server |
      primary_data_server, primary_database, replicate_data_server,
replicate_database |
      primary_data_server, replicate_data_server}
```

パラメータ

- ***data_server*** – 処理時間を推定する特定のキューのあるパスに接続するデータベースが格納されているデータサーバ
- ***database*** – データベース名
- ***q_type*** – キューのタイプ:

  - 0 - アウトバウンドキュー
  - 1 - インバウンドキュー
- ***q_number* –**

  Replication Server がキューに割り当てた ID 番号です。 この番号は、**admin who, sqm** コマンドの出力に表示されます。
- ***replication_server*** – 処理時間を推定するキューをホストしている Replication Server の名前
- ***primary_data_server***、***primary_database***、***replicate_data_server***、***replicate_database*** – インバウンドおよびアウトバウンドキューの合計処理時間を推定する場合の、プライマリデータサーバおよびレプリケートデータサーバおよびデータベース

例

- **例 1 –** Replication Server が、LDS プライマリデータサーバ内の `pubs1` データベースとプライマリ Replication Server の間の複写パスのインバウンドキューの残りのデータを処理するのにかかる時間を推定します。

```
admin sqm_process_time, LDS, pubs1, 1
```

  指定したキューのすべての SQM リーダモジュール、および各リーダモジュール内のトランザクションのバックログの処理にかかる推定時間を最も近い秒に丸めたものが表示されます。

```
Reader Estimated Time to Process
------------------------------ ------------------------
104:1 DSI 107 SYDNEY_DS.pubs1                          0
104:1 DIST LDS.pubs1                                   1
```

  この例では、LDS.pubs1 論理接続の `pubs1` データベース (*q_number*: 104) からのインバウンドキュー内のトランザクションのバックログの処理に要する時間は 1 秒と推定されます。 非ウォームスタンバイ環境では、インバウンドキューには 1 つのリーダのみ表示されます。 この例の 2 つめのリーダは、ウォームスタンバイ論理接続のスタンバイ DSI 用のリーダです。 スタンバイ DSI からのバックログの処理には、0 秒が必要であると推定されます。

- **例 2 –** *q_number* が 104 のデータベースと Replication Server の間の複写パスのアウトバウンドキュー内の残りのデータの処理に、Replication Server が要する時間を推定します。

```
admin sqm_process_time, 104, 0
```

  指定したキューの SQM リーダモジュールと、各リーダモジュール内のトランザクションのバックログの処理に要する推定時間を最も近い秒に丸めたものが表示されます。

```
Reader Estimated Time to Process
------------------------------ ------------------------
104:0 DSI 104 LON_DS.rdb1                             1
```

- **例 3 –** TOKYO_RRS レプリケート Replication Server へのルートにあるアウトバウンドキュー内の残りのデータの処理に、Replication Server が要する時間を推定します。

  ```
  admin sqm_process_time, TOKYO_RRS
  ```

  アウトバウンドキューのすべての SQM リーダモジュールと、各リーダモジュール内のトランザクションのバックログの処理にかかる推定時間を最も近い秒に丸めたものが表示されます。

  ```
  Reader Estimated Time to Process
  ------------------------------ ------------------------
  TOKYO_RRS                                             1
  ```

- **例 4 –** NY_DS プライマリデータサーバ内の `pdb1` プライマリデータベースから LON_DS レプリケートデータサーバ内の `rdb1` レプリケートデータベースへのルートにあるインバウンドキューおよびアウトバウンドキュー内の残りのデータの処理に、Replication Server が要する時間を推定します。

  ```
  admin sqm_process_time, NY_DS, pdb1, LON_DS, rdb1
  ```

  インバウンドキューおよびアウトバウンドキュー内のトランザクションのバックログの処理に要する合計推定時間を、最も近い秒に丸めた値が表示されます。

  ```
  Estimated Time to Process
  -------------------------
                          3
  ```

使用法

- **admin sqm_process_time** を実行後に、Replication Server のカウンタでデータを収集するための、1 分のサンプリング期間があります。 コマンドで出力が表示されるまで待ちます。サンプリング期間中、コマンドへの応答は表示されません。 1 分のサンプリング期間は、特定のキューからのデータの初期収集でのみ発生します。 キューがトランザクションの処理に追いついた場合、**admin sqm_process_time** からの応答が即時に戻されます。
- **admin sqm_process_time** の出力には、キューのすべての SQM リーダモジュールと、各 SQM リーダ内のトランザクションのバックログの処理に要する推定時間が表示されます。
- 推定時間は最も近い秒に丸められます。
- Replication Server のみが、ルートのアウトバウンドキュー内の残りのデータの処理に要する推定時間を取得するように指定します。
- 予定されたフェールオーバを実行していて、インバウンドおよびアウトバウンドキュー内のトランザクションの処理に要する合計推定時間が必要な場合、プ

ライマリデータサーバとレプリケートデータサーバの両方を指定するオプションとともに **admin sqm_process_time** を使用します。

注意： このオプションを使用する場合は、プライマリ Replication Server とレプリケート Replication Server の両方が実行されている必要があります。

- **admin sqm_process_time** は、最大 2 つの Replication Server があり、両者の間に直接ルートがある複写環境でのみ使用できます。 このコマンドは、間接ルートまたは中間ルートを持つ環境では使用できません。 2 つのデータサーバ間の推定時間を要求すると、**admin sqm_process_time** は間接ルートを使用するパスを無視します。
- フルパスのキュー処理時間を推定するには、プライマリデータベースからレプリケートデータベースに少なくとも 1 つのサブスクリプションが存在する必要があります。 このため、**admin sqm_process_time** で単一の論理接続の推定時間を示すことはできますが、**admin sqm_process_time** を使用してアクティブデータベースとスタンバイデータベース間のウォームスタンバイパスの推定時間を示すことはできません。 同様に、**admin sqm_process_time** を使用して Multi-Path Replication™ 環境内の 1 つの接続の推定時間を示すことはできますが、**admin sqm_process_time** ですべての接続の推定時間またはエンドツーエンドのプライマリからレプリケートパスへの推定時間を示すことはできません。
- 2 つのデータサーバ間のバックログのキュー処理推定時間を要求する際、データサーバ間にパスが存在しない場合、次のエラーメッセージが表示されます。

  ```
  No path was found from 'primary_data server' to 'replicate data server'.
  ```

- 次の場合、出力には秒単位の推定処理時間ではなく、最大 INT 値である 2,147,483,647 が表示されます。
  - Replication Server が起動されたばかりである場合。
  - 1 分のサンプリング時間内にカウンタでデータが収集されなかった場合。または
  - DSI などのリーダモジュールがアクティブでない場合。

パーミッション

このコマンドは、すべてのユーザが実行できます。

admin sqm_readers

ステーブルキューを読み込み中のスレッドの読み込みポイントと削除ポイントを表示します。

構文

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

パラメータ

- **q_number –** Replication Server がキューに割り当てた ID 番号です。この番号は、**admin who, sqm** コマンドの出力に表示されます。
- **q_type –** キューのタイプです。インバウンドキューのタイプは 1、アウトバウンドキューのタイプは 0 です。

例

- **例 1 –**

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

```
RdrSpid RdrType ReaderIndex
------- ------- -----------
46SQT 103:1 DIST LDS.pubs0
57SQT 103:1 DSI 107 SYDNEY_DS.pubs21

First Seg.BlockNext ReadLast Seg.BlockDeleteWriteWait
-----------------------------------------------------
14.4314.4414.4311
14.4314.4414.4310
```

使用法

- **admin sqm_readers** は、インバウンドキューを読み取っている各 Replication Server スレッドの読み取りポイントと削除ポイントを表示します。この情報は、Replication Server がキューからメッセージを削除するのに失敗したときに、原因を調べるために使用できます。
- Replication Server では、キューを読み取っているすべてのスレッドの最小の削除ポイントを超えてポイントを消すことはできません。削除ポイントは最初のセグメントブロックにあります。
- *q_number* を調べるには、**admin who, sqm** コマンドを使用してください。
- 表 9 に、 **admin sqm_readers** コマンドの出力カラムを示します。

表 9 : admin sqm_readers で出力されるカラムの説明

カラム	説明
RdrSpid	このリーダのユニークな識別子。
RdrType	キューを読み取っているスレッドのタイプ。
Reader	リーダの情報。この情報に関する詳細については、admin who で出力される Name カラムと Info カラムを参照してください。
Index	このリーダのインデックス。

カラム	説明
First Seg.Block	キューにある、削除されていない最初のセグメントとブロックの番号。この情報は、キューをダンプする場合に便利である。
Next Read	次にキューから読み取られるセグメント、ブロック、ロー。
Last Seg.Block	キューに最後に書き込まれたセグメントとブロック。この情報は、キューをダンプする場合に便利である。
Delete	リーダが削除を許可されているかどうかを示す。“1” はリーダが削除を許可されていることを示す。
WriteWait	リーダが書き込みを待っているかどうかを示す。値 “1” は、リーダが書き込みを待機していることを示す。

パーミッション

このコマンドは、すべてのユーザが実行できます。

参照：

- admin who (107 ページ)
- admin stats (89 ページ)

admin stats

Replication Server のオペレーションに関する情報と統計を表示します。

構文

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

パラメータ

- **sysmon –** パフォーマンスとチューニングのために特に重要と見なされているカウンタの統計だけを表示します。カウンタは、ほぼすべてのモジュールから選択されます。デフォルト値。
- **"all" –** すべてのカウンタから収集した統計を表示します。
- **module_name –** 指定したモジュールのカウンタから収集した統計を表示します。*module_name* は、*cm, dsi*、*dist*、*dsiexec*、*repagent*、*rsi*、*rsiuser*、*serv*、*sqm*、*sqt*、*sts*、*rsh*、*sync* などです。有効なモジュール名を確認するには、**rs_helpcounter** を使用します。

- **inbound | outbound –** *sqt* または *sqm* のタイプです。*sqt* モジュールまたは *sqm* モジュールに対して、**inbound** も **outbound** も指定されていない場合は、両方のタイプのキューの統計がレポートされます。
- **display_name –** カウンタの名前です。有効な表示名を確認するには、**rs_helpcounter** を使用します。*display_name* は必ず *module_name* と組み合わせて使用します。
- **server[, database] –** コネクションに関連する統計を収集する場合、*server* はデータサーバである必要があり、*database* を指定する必要があります。ルートに関連する統計を収集する場合、server は Replication Server である必要があります。*database* は指定できません。
- **instance_id –** SQT や SQM などのモジュールの特定のインスタンスを識別します。インスタンス ID を確認するには、**admin who** を実行し、*Info* カラムを表示します。

  注意： *rsh* モジュールの場合、*SPID* を使用してください。*SPID* を確認するには、**admin who** を実行し、*Spid* カラムを表示します。

  インスタンス ID 0 は、Replication Server 全体の統計を示します。
- **display –** コンピュータ画面に統計を表示します。デフォルト値。
- **save –** RSSD に統計を保存します。**stats_reset_rssd** の現在の設定に応じて、古いサンプリングデータがトランケートまたは保持されます。
- **obs_interval –** サンプリング期間中の各監視間隔を指定します。間隔を指定しない場合は、サンプリング期間と同じ長さの間隔が 1 つだけ存在することになります。各監視間隔は、15 秒以上であることが必要です。秒単位の数値、または “hh:mm[:ss]” のフォーマットで指定できます。
- **sample_period –** サンプリング期間の合計を示します。デフォルト値は 0 です。この場合、現在のカウンタ値がレポートされます。0 以外の値を指定すると、現在のカウンタ値がリセットされ、指定したサンプリング期間で収集されます。秒単位の数値、または “hh:mm[:ss]” のフォーマットで指定できます。

例

- **例 1 –** コネクション 108 のアウトバウンド SQT の統計を 2 時間収集し、データを RSSD に送信します。

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

- **例 2 –** コネクション 108 のアウトバウンド SQT の統計を 2 時間収集し、データを RSSD に送信します。また、サンプリング期間は 30 秒ごとの監視間隔に分割されます。

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

- **例 3 –** コネクション 102 のインバウンド キューについて、SQM モジュールと SQMR モジュールの統計を表示します。

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

```
Report Time:10/31/05 02:14:17 PM
InstanceInstance IDModType/InstVal
-------------------------------------------------
SQM, 102:1 pds01.tpcc1021

MonitorObsLastMaxAvg ttl/obs
-------------------------------------------------
#*SegsActive1111

================================================================
=========
InstanceInstance IDModType/InstVal
-------------------------------------------------------------
SQMR, 102:1 pds01.tpcc, 0 SQT10211

ObserverObsRate x/sec
---------------------------------------
SleepsWriteQ40
```

注意： 出力のカウンタ名の前にあるプレフィックスは、そのカウンタの情報を示しています。たとえば、プレフィックス # は **admin stats, reset** が実行された場合でもリセットされないカウンタを示し、プレフィックス * は **stats_sampling** の設定に関係なく必ずサンプリングされるカウンタを示します。この例では、SegsActive カウンタは常にサンプリングされ、リセットされることはありません。

- **例 4 –** SQM モジュールで SleepsWriteQ カウンタのすべてのインスタンスの統計を収集します。

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

```
Report Time10/31/05 02:17:03 PM

InstanceObserverObsRate x/sec
-----------------------------------------------------------------
SQMR, 101:0 edsprs01.edbprs01, 0,DSISleepsWriteQ00
SQMR, 102:0 pds01.tpcc, 0,DSISleepsWriteQ00
SQMR, 102:1 pds01.tpcc, 0,DSISleepsWriteQ20 0
SQMR, 103:0 rds01.tpcc, 0,DSISleepsWriteQ0 0
```

- **例 5 –** 1 時間 30 分の間、20 秒間隔でサンプリングと RSSD への統計の保存を開始します。

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

使用法

- 次の 3 タイプの統計コレクタがあります。

    - オブザーバ - イベントの発生回数をカウントする。たとえば、Replication Server は、オブザーバを使用して RepAgent からのコマンドが監視された回数をカウントする。
    - モニタ - 値を定期的にサンプリングする。たとえば、Replication Server は、モニタを使用して送信されたコマンドのサイズをサンプリングする。
    - カウンタ - モニタとオブザーバが監視していない統計を収集する。通常、カウンタは特定の値を累計する (特定のタスクを完了するために必要なミリ秒単位の総時間数など)。たとえば、Replication Server は、カウンタを使用して RepAgent から 2 つのコマンドを受信する間の経過時間を累計する。

    オブザーバ、モニタ、カウンタは、監視数、監視された値の合計、最後に監視された値、監視された最大値の 4 種類の統計を監視します。
- **admin stats** は、次の情報を含むレポートを出力します。
    - Instance - モジュールの特定のオカレンス。
    - Instance ID - 特定のモジュールインスタンスの数値識別子。たとえば、2 つの異なる SQM インスタンスのインスタンス ID は、102 と 103 になる。
    - ModType/InstVal - 場合によっては、1 つのインスタンスが複数のバージョンまたはモジュールタイプを持つことがある。たとえば、ある SQM インスタンスがインバウンドタイプとアウトバウンドタイプを持つことがある。SQM インスタンスの場合、インバウンドバージョンのモジュールタイプは 1、アウトバウンドバージョンのモジュールタイプは 0 になる。
    - Monitor、Observer、または Counter - 監視対象の統計コレクタの名前を表示する。たとえば、SleepsWriteQ。
    - Obs - 監視期間中の統計コレクタの監視数。
    - Last - 監視期間中に最後に監視された値。
    - Max - 監視期間中に監視された最大値。
    - Total - 監視期間中に監視された値の合計。
    - Avg ttl/obs - 監視された値の監視期間内での平均値。これは、Total/Obs として計算される。
    - Rate x/sec - 特定の監視期間中に監視された 1 秒あたりの変更。オブザーバは、監視期間内の Obs/seconds としてこれを計算する。モニタとカウンタは、監視期間内の Total/second としてこれを計算する。
- デフォルトでは、**admin stats** は sysmon カウンタの値をレポートします。
- デフォルトでは、**admin stats** は、監視数が 0 (ゼロ) のカウンタをレポートしません。この動作を変更するには、**stats_show_zero_counters** 設定パラメータを on に設定します。
- 統計がコンピュータ画面に表示される場合は、RSSD には格納されません。同様に、統計が RSSD に格納される場合は、画面には表示されません。

- **admin stats...display_name** を使用して特定のカウンタの統計を表示する場合は、**stats_sampling** が off で、監視数が 0 であっても、Replication Server はそのカウンタの統計を常に表示します。
- 依存するモジュールの統計を収集するには、独立したモジュール名で **admin stats** を使用します。**admin stats** コマンドで依存するモジュール名を使用して統計を収集することはできません。

独立したモジュール	依存するモジュール
データサーバインタフェース (DSI)	DSI エグゼキュータ (DSIEXEC)
データサーバインタフェース (DSI)	Active Object Statistics (AOBJ)
ステーブルキューマネージャ (SQM)	SQM リーダ (SQMR)
スレッド同期 (SYNC)	SYNC 要素 (SYNCELE)

Replication Server モジュールの詳細については、『Replication Server 管理ガイド第 2 巻』を参照してください。

パーミッション

このコマンドは、すべてのユーザが実行できます。

参照：

- configure replication server (238 ページ)

admin stats, backlog

インバウンドキューとアウトバンドキューで分配を待機している複写トランザクションの量をセグメントとブロックでレポートします。

構文

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

例

- **例 1 –** インバウンドキューとアウトバウンドキューのトランザクションバックログをレポートします。

```
admin stats, backlog
```

```
Report Time: 10/31/05 02:17:01 PM
Instance                Monitor          Obs  Last  Max  Avg
ttl/obs
---------------------   ---------------  ---  ----  ---  ---
SQMR 101:0
edsprs01.edbprs01,     *SQMRBacklogSeg     0   0     0    0
```

```
   0, DSI
SQMR 102:0
pds01.tpcc,          *SQMRBacklogSeg      0  0   0    0
   0, DSI
SQMR 102:1
pds01.tpcc,          *SQMRBacklogSeg     695   3  3  1
    0, SQT
SQMR 103:0
rds01.tpcc,          *SQMRBacklogSeg     0     0  0  0
0, DSI
=============================================================================
Report Time:10/31/05 02:56:11 PM
Instance                 Monitor            Obs  Last  Max  Avg
ttl/obs
---------------------  ---------------      ---  ----  ---  ---
SQMR101:0
        edsprs01.edbprs01,*SQMRBacklogBlock 0   0     0   0
0, DSI
SQMR 102:0
          pds01.tpcc,*SQMRBacklogBlock      0   0     0   0
  0, DSI
SQMR 102:1 pds01.tpcc,*SQMRBacklogBlock    692  50    64  29
  0, SQT
SQMR 103:0
rds01.tpcc,*            SQMRBacklogBlock    251   0     2   0
  0, DSI
=============================================================================
```

使用法

- **admin stats, backlog** は、次の情報を出力します。
  - Instance - モジュールの特定のオカレンス。
  - Monitor - モニタまたはカウンタの名前。
  - Obs - 監視期間中に監視されたセグメントまたはブロックの数。
  - Last - 最後の監視期間中に作成されたセグメントまたはブロックの数。
  - Max - 監視期間内の 1 回の監視で監視されたセグメントまたはブロックの最大数。
  - Total - 監視期間中に監視されたすべてのセグメントまたはブロックの合計。
- **admin stats, backlog** は、SQMRBacklogSeg カウンタと SQMRBacklogBlock カウンタからデータを収集します。
- セグメントは 1MB、ブロックは 16K です。

パーミッション

このコマンドは、すべてのユーザが実行できます。

admin stats, cancel

現在実行中の非同期コマンドをキャンセルします。複数の監視間隔がある場合、キャンセルした時点ですでに保存されているデータは削除されません。

構文

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

使用法

admin stats, cancel を使用すると、現在実行中の非同期コマンドを明示的に終了できます。Replication Server では、サンプリングがバックグラウンドですでに実行中であるときに、他のサンプリングコマンドを実行することはできません。

パーミッション

admin stats, cancel は、すべてのユーザが実行できます。

admin stats, {md | mem | mem_in_use | max_mem_use}

メモリの使用状況に関する情報をレポートします。

構文

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

パラメータ

- **md –** DIST と RSI ユーザに関連するメッセージ配信の統計をレポートします。
- **mem –** メモリセグメントの現在の使用状況をセグメントのサイズに従ってレポートします。
- **mem_in_use –** 現在使用されているメモリをバイト単位でレポートします。
- **max_mem_use –** Replication Server で使用された最大メモリをバイト単位でレポートします。

例

- **例 1 –** 使用された総メモリ量をバイト単位でレポートします。

```
admin stats, mem_in_use
```

```
Memory_in_Use
```

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

- **例 2 –** 使用された最大メモリ量をバイト単位でレポートします。

```
admin stats, max_mem_use
go
```

```
Max_Memory_Use

-------------
26584414
```

注意： 結果に示された数字は、Oracle Solaris **prstat** コマンドでレポートされた数字と互換性がなければなりません。 **prstat** コマンドの情報については、Oracle のマニュアルを参照してください。

使用法

- メッセージ配信の統計は、DIST スレッドと RSI ユーザに関連しています。
- セグメントのサイズに基づいたメモリセグメントの現在の使用状況
- メモリの現在の使用状況 (バイト数)
- Replication Server による最大メモリ使用量 (バイト単位)

パーミッション

このコマンドは、すべてのユーザが実行できます。

admin stats, reset

リセットできるすべてのカウンタをリセットします。

構文

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

例

- **例 1 –** すべてのカウンタを 0 にリセットします。このコマンドは出力を生成しません。

```
admin stats, reset
```

使用法

rs_statcounter.counter_status カラムのビット 0x10 は、カウンタをリセットできるかどうかを示します。カウンタにこのビットが設定されている場合、**admin stats, reset** やその他のコマンドを使用してカウンタをリセットすることはできません。

パーミッション

このコマンドは、すべてのユーザが実行できます。

参照：

- admin stats (89 ページ)
- admin stats, status (97 ページ)

admin stats, status

モニタとカウンタの設定を表示します。

構文

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

例

- **例 1 –**

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

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

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

使用法

- 次の設定パラメータのデフォルト値と現在の値を表示します。
  - **stats_sampling** - サンプリングがオンかオフかを示す。
  - **stats_show_zero_counters** - 前回のリセット以降の監視数が 0 のカウンタを表示するかどうかを指定する。

パーミッション

admin stats, status は、すべてのユーザが実行できます。

admin stats, {tps | cps | bps}

現在のスループットを1秒あたりのトランザクション数、コマンド数、またはバイト数でレポートします。

構文

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

パラメータ

- **tps** – 現在のスループットを1秒あたりのトランザクション数でレポートするように指定します。
- **cps** – 現在のスループットを1秒あたりのコマンド数でレポートするように指定します。
- **bps** – 現在のスループットを1秒あたりのバイト数でレポートするように指定します。

例

- **例 1** – 1秒あたりのコマンド数でスループットを計算するカウンタを表示します。出力が長いため、ここでは一部だけを示します。

```
admin stats, cps
```

```
Report Time:            10/31/05 02:58:54 PM
Instance
ObserverObsRate x/sec
------------------------------------------------
REP AGENT, pds01.tpcc *CmdsRecv698760

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

Report Time:              10/31/05 02:58:54 PM
Instance
ObserverObsRate x/sec
------------------------------------------------------------
SQM, 101:0 edsprs01.edbprs01 *CmdsWritten 00
SQM, 102:0 pds01.tpcc *CmdsWritten 00
SQM, 102:1 pds01.tpcc *CmdsWritten 6988625
SQM, 103:0 rds01.tpcc *CmdsWritten 4817417

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

Report Time:            10/31/05 02:58:54 PM
Instance
```

```
Observer Obs         Rate x/sec
---------------------------------------------- -----------
----------
SQMR, 101:0 edsprs01.edbprs01, 0, DSI *CmdsRead 0 0
SQMR, 102:0 pds01.tpcc, 0, DSI *CmdsRead 00
SQMR, 102:1 pds01.tpcc, 0, SQT *CmdsRead 5049918
SQMR, 103:0 rds01.tpcc, 0, DSI *CmdsRead4814417

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

...
```

使用法

- Replication Server は、1 秒あたりのスループットを計算するときに、**admin stats, reset** を使用してカウンタを最後にリセットした後に処理されたトランザクション数と経過秒数に基づいて計算します。
- スループットをレポートするモジュールは、計算のタイプごとに異なります。
  - 1 秒あたりのトランザクション数 - SQT、DIST、DSI、その他のモジュールによってレポートされる。
  - 1 秒あたりのコマンド数 - RepAgent、RSIUSER、SQM、DIST、DSI、RSI の各モジュールによってレポートされる。
  - 1 秒あたりのバイト数 - RepAgent、RSIUSER、SQM、DSI、RSI の各モジュールによってレポートされる。SQM は、1 秒あたりのバイト数とブロック数の両方でトランザクションをレポートする。

パーミッション

このコマンドは、すべてのユーザが実行できます。

admin time

Replication Server の現在の時間を表示します。

構文

```
admin time
```

パラメータ

- **なし –**

例

- **例 1 –**

```
admin time
```

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

使用法

- 遅延問題のデバッグや調査をするときに、**admin time** を使用して、マシン時間や時間帯の差を算出できます。
- このコマンドは、Replication Server でタスクを開始または完了した時間を算出するためのスクリプトを作成する場合にも便利です。

パーミッション

このコマンドは、すべてのユーザが実行できます。

admin translate

ある値に対してデータ型変換を実行し、区切られたリテラルフォーマットで結果を表示する。

構文

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

パラメータ

- **value –** 変換する値のリテラル表現です。
- **source_datatype –** データ型の名前 (Replication Server のネイティブデータ型、または *value* の内容とフォーマットを示すデータ型定義) です。
- **target_datatype –** データ型の名前 (Replication Server の基本データ型、または変換に必要な出力であるデータ型定義) です。

例

- **例 1 –** この例では、DB2 *TIMESTAMP* の値 '1999-06-22-14.35.23.123456' を、Oracle *DATE* の値 '22-Jan-99' に変換します。

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

- **例 2 –** この例では、Adaptive Server のバイナリ値 0x1122aabb を、Oracle のバイナリ値 '1122aabb' に変換します。

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

使用法

- 変換元データ型の基本データ型の区切り条件に従って、*value* を区切ります。
- *source_datatype* または *target_datatype* に長さの制限がある場合 (たとえば、*char(255)* など) は、データ型名を一重引用符で囲みます。
- 変換元と変換先のデータ型は、クラスレベル変換とカラムレベル変換のどちらを行うかによって異なります。次のようになります。
  - クラスレベル変換 - *source_datatype* に、パブリッシュデータ型を使用する。
  - カラムレベル変換 - *source_datatype* には宣言したデータ型、*target_datatype* にはパブリッシュデータ型を使用する。
- 変換中のエラーを追跡するには、Replication Server の診断バージョンとともに **admin translate** を使用します。
- サポートされているデータ型変換については、『Replication Server 異機種間複写ガイド』を参照してください。異機種データ型サポート (HDS: Heterogeneous Datatype Support) を使用したデータ型の変換については、『Replication Server 管理ガイド 第 1 巻』を参照してください。

パーミッション

このコマンドは、すべてのユーザが実行できます。

参照：

- alter replication definition (199 ページ)
- create replication definition (346 ページ)
- alter connection (134 ページ)
- create connection (287 ページ)

admin verify_repserver_cmd

Replication Server が複写定義の要求を実行できることを確認する。

構文

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

パラメータ

- **rs_api** – 確認する複写コマンド言語 (RCL) コマンドおよび対応するすべてのパラメータを含む文字列。

rs_api を一重引用符で囲み、文字列内の各一重引用符を二重引用符で置き換えます。

例

- **例 1 –** この例では、**admin verify_repserver_cmd** は **alter replication definition** を使用して複写定義からカラムを削除し、古い複写定義バージョンがスタンバイまたはレプリケートデータベースなどのターゲットに複製された後で、ターゲット DSI を正常にサスペンドするかどうかをテストします。

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

Replication Server で **alter replication definition** コマンドが実行できると、Replication Server では次のメッセージが返されます。

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

- **例 2 –** 次の例は、**admin verify_repserver_cmd** を使用して、存在しない複写定義からカラムを削除できるかどうかを確認するとどうなるかを示します。

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

Replication Server で、"authors_does_not_exist" という複写定義が存在しないことを示すメッセージが返されます。

- **例 3 –** 次の例は、**admin verify_repserver_cmd** がコマンドラインで "columns" キーワードを使用するなどの構文エラーを検出できることを示します。

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

Replication Server では次のようなメッセージが返されます。

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

- **例 4 –** 次の例は、**admin verify_repserver_cmd** で、'off' を囲むために二重引用符を使用するなど、引用符を正しく使用しているかどうかを検出します。

```
admin verify_repserver_cmd, 'alter replication
definition authors replicate sqldml "off"'
```

Replication Server では次のようなメッセージが返されます。

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

正しい構文は次のとおりです。

```
admin verify_repserver_cmd, 'alter replication
definition authors replicate sqldml ''off'''
```

使用法

- Replication Agent が複写定義 RCL を実行するために Replication Server に送信したのに、複写定義 RCL を実行できないと、Replication Agent は停止します。この状況を回避するには、**admin verify_repserver_cmd** を使用して、プライマリデータベースから直接 RCL を実行する前に、Replication Server が正常に複写定義要求を実行できることを確認します。要求を正常に実行できない場合、Replication Server はエラーを返します。
- Replication Server でサポートされる **admin verify_repserver_cmd** の複写定義コマンドは、**rs_send_repserver_cmd** と同じです。
  - **alter replication definition**
  - **create replication definition**
  - **drop replication definition**
  - **alter applied function replication definition**
  - **create applied function replication definition**
  - **alter request function replication definition**
  - **create request function replication definition**

パーミッション

このコマンドは、すべてのユーザが実行できます。

参照：

- admin verify_repserver_cmd (101 ページ)
- alter replication definition (199 ページ)
- rs_send_repserver_cmd (735 ページ)
- sysadmin skip_bad_repserver_cmd (505 ページ)

admin version

Replication Server ソフトウェアのバージョン番号を表示します。

構文

```
admin version
```

例

- **例 1 –**

```
admin version
```

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

使用法

- Replication Server のソフトウェアのバージョン番号とは、ソフトウェア製品のリリースレベルのことです。
- ソフトウェアバージョン番号は、単独では Replication Server で使用できる機能を決定できません。複写システムのシステムバージョン番号と Replication Server のサイトバージョン番号によっても、使用できる機能が決定されます。
- Replication Server のサイトバージョン番号は、ソフトウェアバージョン番号と同じであるか、それより前のバージョンになります。 詳細については、「**sysadmin site_version**」の項を参照してください。
- 複写システムのシステムバージョン番号は、ソフトウェアバージョン番号と同じであるか、それより前のバージョンになります。詳細については、「**sysadmin site_version**」の項を参照してください。

パーミッション

このコマンドは、すべてのユーザが実行できます。

参照：

- sysadmin site_version (502 ページ)
- sysadmin system_version (519 ページ)

admin version, "connection"

Replication Server をアップグレードした後、ユーザデータベースのアップグレードステータスを表示して、アップグレードが必要なユーザデータベースを特定します。

構文

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

例

- **例 1 –** アップグレードした Replication Server で次のように入力します。

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

ユーザデータベースとデータベースサーバのリスト、データベース ID、対応する Replication Server、およびデータベースのステータスが出力されます。例：

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

使用法

- アップグレードした Replication Server で **admin version, "connection"** と入力します。
- “Database is not accessible” ステータスは、Replication Server がデータベースへの接続に使用するメンテナンスユーザ ID に十分な権限がないか、データベースが使用不可のため、Replication Server がこのユーザデータベースに接続できないことを示します。

『設定ガイド』の「sysadmin upgrade, "database" を使用したユーザデータベースアップグレードの修正自動アップグレード」を参照してください。

パーミッション

admin version, "connection" は、すべてのユーザが実行できます。

参照：

- sysadmin upgrade, "database" (522 ページ)

admin version, route

現在の Replication Server から送信先 Replication Server まで、または送信元 Replication Server から現在の Replication Server までのアップグレードするルートをレポートし、ルートアップグレードのステータスを調べます。

構文

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

例

- **例 1 –** 現在の NY_RS Replication Server から送信先 LON_RS Replication Server までのルートのアップグレードステータスをレポートします。

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

  つまり、次のようになります。

  - ルートのアップグレードに失敗し、アップグレードからルートをリカバリする必要がある場合は、次のように表示されます。

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

  - ルートのアップグレードが進まず、まだアップグレードするルートがある場合は、次のように表示されます。

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

  - ルートをアップグレードする必要がない場合や、ルートのアップグレードに成功した場合、ルートは出力に表示されません。

使用法

- **admin version, route** を使用すると、以下が出力されます。
  - ルートの送信元 Replication Server。
  - ルートの送信先 Replication Server。
  - ルートの現在のバージョン。
  - アップグレードするルートの推奨バージョン。
  - ルートのアップグレードステータス。

『設定ガイド』の「ルートのアップグレード」を参照してください。

パーミッション

このコマンドは、すべてのユーザが実行できます。

admin who

SAP Replication Server で実行されているスレッドについての情報を表示します。

構文

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

パラメータ

- **dist –** ディストリビュータスレッドの情報を返します。 このスレッドは、インバウンドキューのトランザクションをレプリケートデータベースと SAP Replication Server に分配します。
- **dsi –** DSI スレッドの情報を返します。このスレッドは、複写トランザクションをデータベースに適用します。
- **rsi –** RSI スレッドの情報を返します。 このスレッドは、他の SAP Replication Server にメッセージを送信します。
- **sqm –** SQM スレッドの情報を返します。 このスレッドは、SAP Replication Server のステーブルキューを管理します。
- **sqt –** SQT スレッドの情報を返します。このスレッドは、キューとグループファンクションをトランザクションに読み込みます。
- **no_trunc –** Info カラムのサイズを 40 文字から 80 文字に増やします。 このパラメータは、長いデータサーバ名やデータベース名を表示するときに役立ちます。

  注意： 接続識別子を使用すると **no_trunc** は使用できません。
- **接続識別子 –** スレッドモジュールの **admin who** 出力をフィルタします。 スレッドモジュールによっては、次の 1 つまたは複数のコマンドを使用して、接続識別子を作成できることがあります。
  - *db_id* - データベース識別子 (数値)
  - *db_name* - データベース名
  - *ds_name* - データサーバ名
  - *q_number* - ステーブルキュー番号
  - *q_type* - ステーブルキューのタイプ。0 はアウトバウンドキュー、1 はインバウンドキュー
  - *rs_id* - SAP Replication Server 識別子 (数値)
  - *rs_name* - SAP Replication Server 名

表 10 : admin who スレッドと対応する接続識別子

スレッド	接続識別子	例
DIST	次のいずれかを提供することによって、1 つまたは複数の特定のコネクションのディストリビュータ (DIST) スレッド情報を表示します。 • *db_id* • *ds_name* • *ds_name* および *db_name* 接続識別子にデータサーバ名のみを指定すると、**admin who** はデータサーバに属するすべてのデータベースのディストリビュータ情報をリスト表示します。	データサーバ “ASE_01” とデータベース “DB01” の DIST 情報を表示します。 `admin who, dist, ASE_01, DB01`
DSI	次のいずれかを提供することによって、1 つまたは複数の特定のコネクションのデータサーバインタフェース (DSI) スレッド情報を表示します。 • *db_id* • *ds_name* • *ds_name* および *db_name* 接続識別子にデータサーバ名のみを指定すると、**admin who** はデータサーバに属するすべてのデータベースの DSI コネクション情報をリスト表示します。	*db_id* = 101 の DSI 情報を表示します。 `admin who, dsi, 101`
RSI	次のいずれかを提供することによって、特定のコネクションのレプリケーションサーバインタフェース (RSI) スレッド情報を表示します。 • *rs_id* • *rs_name* SAP Replication Server 名または SAP Replication Server 識別子のいずれかを使用して、コネクションを指定できます。	*rs_id* = 16777318 の RSI 情報を表示します。 `admin who, rsi, 16777318`

スレッド	接続識別子	例
SQM	次のいずれかを提供することによって、1 つまたは複数の特定のコネクションのステーブルキューマネージャ (SQM) スレッド情報を表示します。 • *ds_name* および *db_name* (*q_type* を使用する場合と使用しない場合) • *ds_name* (*q_type* を使用する場合と使用しない場合) • *q_number* (*q_type* を使用する場合と使用しない場合) • *rs_id* (*q_type* を使用する場合と使用しない場合) • *rs_name* (*q_type* を使用する場合と使用しない場合) *q_type* を指定しないと、指定したデータサーバのすべての **admin who, sqm** エントリのインバウンドキュータイプとアウトバウンドキュータイプがリスト表示されます。	*q_type*=1 (インバウンドキュー）のデータサーバ “ASE_01” の SQM 情報を表示します。 `admin who, sqm, ASE_01, 1`
SQT	次のいずれかを提供することによって、1 つまたは複数の特定のコネクションのステーブルキュートランザクション (SQT) スレッド情報を表示します。 • *ds_name* および *db_name* (*q_type* を使用する場合と使用しない場合) • *ds_name* (*q_type* を使用する場合と使用しない場合) • *q_number* (*q_type* を使用する場合と使用しない場合) • *rs_id* (*q_type* を使用する場合と使用しない場合) • *rs_name* (*q_type* を使用する場合と使用しない場合) *q_type* を指定しないと、指定したデータサーバのすべての **admin who, sqt** エントリのインバウンドキュータイプとアウトバウンドキュータイプがリスト表示されます。	インバウンドキューのデータサーバ “ASE_01”、データベース “DB01” の SQT 情報を表示します。 `admin who, sqt, ASE01, DB01, 1`

例

- **例 1 –** 次の例では、**admin who** コマンドによって SAP Replication Server のすべてのスレッドのステータスを表示しています。 DSI スケジューラスレッドは、出力では “DSI” として表示されます。 DSI エグゼキュータスレッドは “DSI EXEC” として表示されます。 SAP Replication Server の起動時に DSI がサスペンドすると、複数のスレッドが設定されていても、1 つの EXEC エグゼキュータスレッドだけが表示されます。

```
admin who

Spid  Name      State             Info
```

```
----- -----     -----
-----------------------------------------
97    DIST      Active             103 LDS.pubs2
98    SQT       Awaiting Wakeup    103:1 DISTLDS.pubs2
68    SQM       Awaiting Message   103:0 LDS.pubs2
89    DSI EXEC  Awaiting Message   106(1)SYDNEY_DS.pubs2sb
91    DSI       Awaiting Command   106 SYDNEY_DS.pubs2sb
21    DSI EXEC  Awaiting Message   101(1)TOKYO_DS.TOKYO_RSSD
10    DSI       Awaiting Command   101 TOKYO_DS.TOKYO_RSSD
16    DIST      Active             101 TOKYO_DS.TOKYO_RSSD
17    SQT       Awaiting Wakeup    101:1 DISTTOKYO_DS.TOKYO_RSSD
15    SQM       Awaiting Message   101:1 TOKYO_DS.TOKYO_RSSD
14    SQM       Awaiting Message   101:0 TOKYO_DS.TOKYO_RSSD
30    REPA GENT Awaiting Command   TOKYO_DS.TOKYO_RSS USER
4     DSI EXEC  Awaiting Message   104(1)TOKYO_DS.pubs2
0     DSI       Awaiting Command   104 TOKYO_DS.pubs2
8     REP AGENT Awaiting Command   TOKYO_DS.pubs2
      USER
53    RSI       Awaiting Wakeup    SYDNEY_RS
52    SQM       Awaiting Message   16777318:0 SYDNEY_RS
      RSI USER  Inactive TOKYO_RS
11    dSUB      Active
6     dCM       Awaiting Message
9     dAIO      Awaiting Message
12    dREC      Active             dREC
71    USER      Active             sa
47    GATEWAY   Awaiting Command   SYDNEY_RS
5     dALARM    Awaiting Wakeup
13    dSYSAM    Sleeping
```

- **例 2 –** 次の例では、**admin who, dist** コマンドによって、SAP Replication Server にある各 DIST スレッドの情報を表示しています。

```
admin who, dist
```

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

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

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

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

- **例 3 –** 次の例では、**admin who, dsi** コマンドによって、SAP Replication Server で実行中の各 DSI スケジューラスレッドの情報を表示しています。

```
admin who, dsi
```

```
 Spid                  State                          Info
 -----       ----------------          ----------------------
     8       Awaiting Message          101 TOKYO_DS.TOKYO_RSSD
    79       Awaiting Message          104 TOKYO_DS.pubs2
   145       Awaiting Message          105 SYDNEY_DS.pubs2sb

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

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

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

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

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

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

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

```
     Applying                    0              on              on
     Applying                    0             off             off

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

CacheSize  Serialization    Max_Xacts_in_group  Xacts_retried_blk
---------  ---------------  ------------------  -----------------
0          wait_for_commit                  20                  0
0          wait_for_commit                 200                  0
0          wait_for_start                   20                  0

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

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

- **例 4 –** 次の例では、**admin who, rsi** によって、RSI スレッドに関する情報を表示しています。

```
admin who, rsi
```

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

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

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

- **例 5 –** 次の例では、**admin who, sqm** によって、SQM スレッドに関する情報を表示しています。

```
admin who, sqm
```

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

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



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

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

    Readers     Truncs     Loss Status
    -------     ------     -----------
          1          1     Suspect
          1          1     Detecting
          1          1     No Loss
          1          1     Detecting
```

- **例 6 –** 次の例では、**admin who, sqt** によって、SQT スレッドに関する情報を表示しています。

```
admin who, sqt
```

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

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

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

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

- **例 7 –** 次の例では、NY_DS.pdb1 プライマリコネクションのために、RS_NY プライマリ SAP Replication Server と RS_LON レプリケート SAP Replication Server との間に専用ルートがあります。 2 つの SAP Replication Server に **admin who** と入力すると、次のように表示されます。

  - RS_LON で:

    ```
    admin who
    ```

    ```
    Spid Name       State               Info
    45 SQT          Awaiting Wakeup     103:1  DIST NY_DS.pdb1
    13 SQM          Awaiting Message    103:1 NY_DS.pdb1
    32 REP AGENT    Awaiting Command    NY_DS.pdb1
    16 RSI          Awaiting Wakeup     RS_LON
    11 SQM          Awaiting Message    16777318:0 RS_LON
    55 RSI          Awaiting Wakeup     RS_LON(103) /* Dedicated RSI
    thread */
    53 SQM          Awaiting Message    16777318:103 RS_LON(103) /
    *Dedicated RSI outbound queue */
    ```

  - RS_NY で:

    ```
    admin who
    ```

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

使用法

- オプションを指定して **admin who** を使用する場合は、オプションの前にカンマを付けます。
- 接続識別子を指定しても、Replication Server がそれに一致する情報を見つけることができない場合、出力にはレコードが 1 つも表示されません。
- Replication Server のすべてのスレッドについて情報を表示するには、オプションを指定しないで **admin who** を実行します。

admin who の出力カラムの説明

オプションを指定しないで **admin who** を実行すると、*spid*、*Name*、*State*、および *Info* カラムが表示されます。どのオプションを指定しても、*spid*、*State*、および *Info* カラムは表示されます。

spid カラム

Replication Server で実行中のスレッドのユニークな識別子が表示されます。スレッドがサスペンドまたは停止されている場合、このフィールドはブランクになります。

Name カラムと *Info* カラム

Name には、Replication Server スレッドのタイプが表示されます。*Info* の内容は、スレッドのタイプによって異なります。

表 11 : admin who で出力される Name カラムと Info カラムの説明

名前	説明	*info* の内容
dAlarm	アラームデーモン。このスレッドは、コネクションのフェードアウト時間やサブスクリプションデーモンのリトライインターバルなど、他のスレッドによって設定されたアラームを追跡する。	ブランク
dAIO	非同期 I/O デーモン。Replication Server のステーブルキューに対する非同期 I/O を管理する。	ブランク
dCM	コネクションマネージャのデーモン。データサーバに対するコネクションや他の Replication Server へのコネクションを管理する。	ブランク
dREC	リカバリデーモン。設定可能な時間 (**rec_daemon_sleep_time** 設定パラメータ) スリープし、*rs_recovery* テーブルに指定されたリカバリ処理を開始する。	ブランク
dSUB	サブスクリプションリトライデーモン。このスレッドは、設定されたタイムアウト時間 (**sub_daemon_sleep_time** 設定パラメータで設定可能) の後にウェイクアップし、失敗したサブスクリプションの再起動を試みる。	ブランク
dSYSAM	SySAM デーモン。このスレッドは、チェックアウトされたライセンスを追跡する。	ブランク
dVERSION	バージョンデーモン。このスレッドは、Replication Server をアップグレード後に初めて起動したときに、一時的にアクティブになる。Replication Server の新しいソフトウェアバージョン番号を ID サーバに知らせる。	この Replication Server のバージョン。
DIST	ディストリビュータスレッド。各プライマリデータベースには、インバウンドキューからトランザクションを読み取り、処理対象となるサブスクリプションを調べ、トランザクションを転送するディストリビュータスレッドがある。	データサーバとデータベースの名前。スレッドは、これらの更新を分配する。

名前	説明	*info* の内容
DSI	DSI スケジューラスレッド。このスレッドは、SQT を介してステーブルキューを読み取り、DSI エグゼキュータスレッドを介してトランザクションを適用する。	スレッドが書き込みを行うデータサーバの名前。
DSI EXEC	DSI エグゼキュータスレッド。このスレッドは、レプリケートデータベースでトランザクションを実行し、レプリケートデータサーバが返すエラーに対してアクションを実行する。	DSI エグゼキュータスレッドの ID とその接続先データサーバの名前。
GATEWAY	ゲートウェイサーバスレッド。このスレッドは、クライアントからサーバにコマンドを渡し、サーバの応答をクライアントに返す。	ゲートウェイサーバとして動作する Replication Server の名前。
REP AGENT USER	RepAgent スレッドの受信側。RepAgent の要求処理が有効かどうかを確認し、それをインバウンドキューに書き込む。	プライマリデータサーバの名前と RepAgent がログを転送するデータベース。
RSI	RSI 送信側。このスレッドは、ある Replication Server から別の Replication Server にメッセージを送信する。	メッセージが送信される Replication Server の名前。
RSI User	Replication Server から接続を受けるスレッド。このスレッドは、他の Replication Server またはデータベースに向けられたメッセージをアウトバウンドキューに書き込む。	クライアントとして、この Replication Server に接続している Replication Server の名前。
RS User	プライマリ Replication Server でサブスクリプションを作成または削除するために使用される Replication Server コネクション。	サブスクリプション所有者の名前。

名前	説明	*info* の内容
SQM	ステーブルキューマネージャ。このスレッドは、Replication Server のステーブルキューを管理する。	キュー番号：Replication Server またはデータベースの ID。 キュータイプ：1 はインバウンドキュー、0 はアウトバウンドキューを示す。 それ以外の数の場合は、キューのサブスクリプションの ID を示す。 キュー識別子：次のキューの識別子を示す。 • 別の Replication Server へのメッセージをスプールするために使用されているキューの場合は、その Replication Server の名前。 • データベースへのメッセージをスプールするために使用されているキューの場合は、データサーバとデータベースの名前。 • キューが作成または削除しているサブスクリプションに関連したメッセージを保管するために使用されている場合は、複写定義とサブスクリプションの名前。
SQT	ステーブルキュートランザクションインタフェース。このスレッドは、ステーブルキューから一連のメッセージを読み取り、トランザクションをコミットされた順序で再構築する。ディストリビュータと DSI がこのスレッドを使用する。	対応する SQM スレッドと同じ。
USER	RCL コマンドを実行しているクライアント用のスレッド。	クライアントのログイン名。

State カラム

The *State* カラムには、スレッドの実行ステータスが表示されます。次のテーブルに、Replication Server スレッドの有効なステータスを示します。DSI スレッドのステータスは、それらがスケジューラスレッドかエグゼキュータスレッドかにより

異なって定義されます。定義については、『Replication Server トラブルシューティングガイド』を参照してください。

表 12 : admin who で出力される State カラムの説明

状態	説明
Active	コマンドの処理を実行中 (アクティブ状態)。
Active, DSI timer	コマンドの処理を実行中 (アクティブ状態)。**dsi_timer** はオン。
Awaiting Batch Order	DSI スレッドがレプリケートデータサーバへのコマンドバッチの送信を待機中。
Awaiting Command	クライアントからのコマンドを待機中。
Awaiting Command, DSI timer	クライアントからのコマンドを待機中。**dsi_timer** はオン。
Awaiting Commit Order	完了したトランザクションをコミットする順番を待機中。
Awaiting I/O	I/O オペレーションの終了を待機中。
Awaiting Message	SAP® Open Server™ メッセージキューからのメッセージを待機中。
Awaiting Message, DSI timer	SAP Open Server メッセージキューからのメッセージを待機中。**dsi_timer** はオン。
Awaiting Upgrade	ストアドプロシージャやアップグレードするテーブルなど、ユーザデータベースオブジェクトを待機中。
Awaiting Wakeup	スリープ状態にあり、起動されるのを待機中。
Checking Condition	イベントの発生を待機中。
Connecting	接続中。
Controlling Mem	スレッドはメモリ制御を実行しています。
Disconnecting	切断中。
Down	起動していないか、終了している。
Getting Lock	相互排他ロックを待機中。
Inactive	送信元 SAP Replication Server が送信先 SAP Replication Server に接続されていないときの、ルートの送信先にある RSI User スレッドのステータス。
Initializing	初期化中。
Invalid	不定ステータス。
Locking Resource	共有リソースのロック試行中。

状態	説明
Not Running	停止準備のためにクリーンアップ中。
Reading Disk	ディスク読み込みの準備中。
SkipUntil Dump	スレッドが再同期データベースマーカを受け取った。このステータスは DSI がその後のダンプデータベースマーカを処理するまで続く。
Setting Condition	別のスレッドがウェイクアップするための状態を設定中。
SkipUntil Resync	**skip to resync** の実行後にスレッドがレジューム中。このステータスはスレッドが再同期データベースマーカを受け取るまで続く。
Sleeping	一定期間、プロセッサ時間を解放中。
Sleeping For Mem	空きメモリが確保されるまで、スレッドはスリープします。
Suspended	ユーザによってサスペンドされている。
Unlocking Resource	共有リソースを解放中。

admin who, dist の出力カラムの説明

このコマンドは、SAP Replication Server の各 DIST スレッドのローを含むテーブルを返します。

表 13 : admin who, dist で出力されるカラムの説明

カラム	説明
PrimarySite	SQT スレッドのプライマリデータベースの ID。
Type	スレッドが物理コネクションであるか論理コネクションであるかを示す。
Status	スレッドのステータスが “normal” であるか “ignoring” であるかを示す。
PendingCmds	スレッドで保留中のコマンドの数。
SqtBlocked	SQT を待機中かどうかを示す。
Duplicates	スレッドが検出し、削除した重複コマンドの数。
TransProcessed	スレッドによって処理されたトランザクションの数。
CmdsProcessed	スレッドによって処理されたコマンドの数。
MaintUserCmds	メンテナンスユーザに属するコマンドの数。

カラム	説明
NoRepdefCmds	対応するテーブル複写定義が定義されていないために削除されたコマンドの数。 ウォームスタンバイの場合、Replication Server に複写定義を作成させることができる。MSA (Multi-Site Availability) では、ユーザがデータベース複写定義を定義する。いずれの場合も、複写データがテーブル複写定義のない送信元のデータである場合は、カウンタが増やされ、複写データがターゲットに送られる。
CmdsIgnored	ステータスが “normal” になる前に削除されたコマンドの数。
CmdMarkers	処理された特別なマーカの数。
RSTicket	SAP Replication Server の **stats_sampling** パラメータが on の場合に DIST スレッドによって処理された **rs_ticket** サブコマンド数。 最小値：0Maximum:2^{63}-1Default:0
SqtMaxCache	データベースコネクションの SQT (ステーブルキュートランザクションインタフェース) キャッシュメモリの最大量 (バイト単位)。 デフォルトの 0 は、コネクションの最大キャッシュサイズとして **sqt_max_cache_size** の現在の設定を使用することを示す。

admin who, dsi の出力カラムの説明

このコマンドは、SAP Replication Server で実行中の各 DIST スケジューラスレッドに対するローを含むテーブルを返します。 データベースの DSI スケジューラスレッドが存在していても **admin who, dsi** の出力に表示されない場合は、**resume connection** を使用してデータベースのデータサーバインタフェースを再起動してください。

表 14 : admin who, dsi で出力されるカラムの説明

カラム	説明
Maintenance User	トランザクションを適用するメンテナンスユーザのログイン名。
Xact_retry_times	エラーアクションが RETRY_LOG または RETRY_STOP の場合に、失敗したトランザクションをリトライした回数。
Batch	バッチオプションがオンかどうかを示す。オンの場合、複数のコマンドを 1 つのバッチとしてデータサーバに送信できる。
Cmd_batch_size	データサーバに送信できる出力コマンドのバッチの最大サイズ (バイト単位)。

カラム	説明
Xact_group_size	送信元コマンドを構成するトランザクショングループの最大サイズ (バイト単位)。
Dump_load	ダンプ／ロードオプションがオンかどうかを示す。この設定オプションは、プライマリデータベースとレプリケートデータベース間のダンプを調整する。
Max_cmds_to_log	トランザクションの例外ログに書き込めるコマンドの最大数。-1 はコマンドの数に制限がないことを示す。
Xacts_read	アウトバウンドステーブルキューから DSI によって読み取られたトランザクションの数。この数は、DSI がトランザクションを適用するたびに増加する。この情報は、アクティビティの割合のモニタに使用できる。
Xacts_ignored	重複すると判断されたトランザクションの数。 通常、以前に適用されたトランザクションは起動時に無視される。 DSI キューからの削除が遅れるため、起動時に重複が検出され、無視される。 無視されたトランザクションの数が多い場合は、*rs_lastcommit* テーブルに障害が発生している可能性がある。 詳細については、『トラブルシューティングガイド』を参照する。
Xacts_skipped	**skip first transaction** でコネクションをレジュームしたことによってスキップされたトランザクションの数。
Xacts_succeeded	データベースに対して正常に適用されたトランザクションの数。
Xacts_failed	失敗したトランザクションの数。エラーマッピングに応じて、一部のトランザクションは例外ログに書き込まれることがある。この場合は例外ログを調べる。
Xacts_retried	リトライされたトランザクションの数。
Current Origin DB	現在のトランザクションのオリジンデータベース ID。
Current Origin QID	ステータスが Active の場合は、処理中のトランザクションの begin ログレコードのオリジンキュー ID。それ以外の場合は、最後に処理されたトランザクションの begin ログレコードのオリジンキュー ID。
Subscription Name	スレッドがサブスクリプションを処理中の場合は、サブスクリプションの名前。
Sub Command	スレッドがサブスクリプションの処理を実行中の場合は、サブスクリプションのコマンド (activate、validate、drop、または unknown)
Current Secondary QID	スレッドが、インクリメンタルに適用されたアトミックサブスクリプションを処理中の場合、このカラムには現在のトランザクションのキュー ID が表示される。
Cmds_read	DSI キューから読み取られたコマンドの数。

カラム	説明
Cmds_parsed_by_sqt	DSI キューによって読み取られる前に、SQT によって解析されたコマンドの数。
IgnoringStatus	マーカの待機中に DSI がトランザクションを “Ignoring” している場合は、DSI がデータベースでトランザクションを実行している場合は、“Applying” となる。
Xacts_Sec_ignored	ウォームスタンバイアプリケーションで、切り替え後に無視されたトランザクションの数。
GroupingStatus	DSI がグループ化されたトランザクションを実行している場合は、“on” となる。DSI が一度に 1 つのトランザクションを実行している場合は、“off” となる。
TriggerStatus	**set triggers** が **on** に設定されているときは “on”。**set triggers** が **off** に設定されているときは “off” になる。
ReplStatus	SAP Replication Server がデータベースのトランザクションを複写するかどうかを示す。 デフォルトでは、スタンバイデータベースに対しては “off”、 他のすべてのデータベースに対しては “on” となる。
NumThreads	使用中の並列 DSI スレッドの数。
NumLargeThreads	ラージトランザクションで使用するために予約されている並列 DSI スレッドの数。
LargeThreshold	並列 DSI 設定で、ラージトランザクションと見なされるまでに 1 つのトランザクション内で許可されるコマンドの数。
CacheSize	データベースコネクション用の最大 SQT キャッシュメモリ (バイト単位)。デフォルトの 0 は、**sqt_max_cache_size** パラメータの現在の設定が最大 SQT キャッシュメモリとして使用されていることを示す。
Serialization	並列 DSI スレッドが使用される場合に、順序一貫性を維持するために使用する方法。
Max_Xacts_in_group	グループ内のトランザクションの最大数。デフォルトは 20。この数は、**alter connection** コマンドを使用して設定できる。
Xacts_retried_blk	ロック競合チェックの最大回数を超えたために、DSI がトランザクションをロールバックした回数。
CommitControl	コミット制御が内部と外部のいずれであるかを示す。内部の場合は true に設定する。
CommitMaxChecks	トランザクションをロールバックし、リトライするまでのロック競合試行の最大回数を示す。
CommitLogChecks	メッセージをロギングするまでのロック競合試行の最大回数を示す。

カラム	説明
CommitCheckIntvl	ロック競合チェックを実行するまでにトランザクションが待機する時間 (ミリ秒単位)。
IsolationLevel	DSI コネクションのためのデータベースの独立性レベル。
RSTicket	SAP Replication Server の **stats_sampling** パラメータが “on” の場合に DSI キューマネージャによって処理された **rs_ticket** サブコマンド数。 デフォルトの 0 は、コネクションの最大キャッシュサイズとして **sqt_max_cache_size** の現在の設定を使用することを示す。
dsi_rs_ticket_report	**rs_ticket_report** ファンクション文字列を呼び出すかどうかを指定する。**dsi_rs_ticket_report** を on に設定すると、**rs_ticket_report** ファンクション文字列が呼び出される。 デフォルト値は off

admin who, rsi の出力カラムの説明

このコマンドは、他の SAP Replication Server にメッセージを送信する RSI スレッドの情報を表示します。

表 15 : admin who, rsi で出力されるカラムの説明

カラム	説明
Packets Sent	送信されたネットワークパケットの数。
Bytes Sent	送信されたバイトの合計数。
Blocking Reads	ステーブルキューがブロッキングリードで読み込まれた回数。
Locater Sent	最後に送信されたメッセージのロケータ (キューセグメント、ブロック、ローを含む)。
Locater Deleted	受信側が確認通知を受け取り、SAP Replication Server によって削除された最後のロケータ。

admin who, sqm の出力カラムの説明

このコマンドは、SAP Replication Server のステーブルキューを管理する SQM スレッドの情報を表示します。

表 16 : admin who, sqm で出力されるカラムの説明

カラム	説明
Duplicates	検出され、無視された重複メッセージの数。通常は、起動時にいくつかの重複メッセージが発生する。
Writes	キューに書き込まれたメッセージの数。
Read	キューから読み取られたメッセージの数。書き込みを開始する場所を調べるために、最後のセグメントが起動時に読み取られるため、この数は通常、書き込みの数より多くなる。トランザクションが長い場合は、メッセージが再度読み取られることがある。
Bytes	書き込まれたバイト数。
B Writes	書き込まれた 16K ブロックの数。書き込まれたすべての 16K ブロックが満杯というわけではないため、*Bytes*/16K より多くなる場合がある。ブロックの密度は、*Bytes* を *B Writes* で除算することによって判断できる。
B Filled	満杯になったためディスクに書き込まれた 16K ブロックの数。
B Reads	読み取られた 16K ブロックの数。
B Cache	キャッシュにある、読み取られた 16K ブロックの数。
Save_Int:Seg	*Save_Int* の間隔、および *Save_Int* リスト内の最も古いセグメント。 *Save_Int* の間隔は、セグメントにあるすべてのメッセージがターゲットに通知された後、SAP Replication Server によって SQM セグメントが保持される分数。 たとえば、値 5:88 は *Save_Int* の間隔が 5 分であり、セグメント 88 が *Save_Int* リスト内の最も古いセグメントであることを示す。 この機能は、複写システムの障害時に冗長性を提供する。 たとえば、他の SAP Replication Server からデータを受信中に、SAP Replication Server のディスクパーティションの空きがなくなる可能性がある。 *Save_Int* 機能により、送信側の SAP Replication Server は *Save_Int* の間隔中に保存されたすべてのメッセージを再作成できる。 *Save_Int* の “strict” 値は、キューが複数のリーダスレッドによって読み取られているときに使用できる。 SAP Replication Server は、キューを読み取っているすべてのスレッドが、セグメントにあるメッセージを読み取り送信先に適用するまで、SQM 上のセグメントを保持する。
First Seg.Block	キューにある、削除されていない最初のセグメントとブロックの番号。*First Seg.Block* の数と *Last Seg.Block* の数が一致しない場合、データは処理のためにキュー内に残る。 この情報は、キューをダンプする場合に便利である。 詳細については、『トラブルシューティングガイド』を参照する。

カラム	説明
Last Seg.Block	キューに最後に書き込まれたセグメントとブロック。*First Seg.Block* の数と *Last Seg.Block* の数が一致しない場合、データは処理のためにキュー内に残る。 この情報は、キューをダンプする場合に便利である。 詳細については、『トラブルシューティングガイド』を参照する。
Next Read	次にキューから読み取られるセグメント、ブロック、ロー。
Readers	キューを読み取っているスレッドの数。
Truncs	キューのトランケーションポイントの数。
Loss Status	データ消失ステータス: • Suspect - キューでデータが失われた可能性が疑われる。 • Detecting - キューのデータ消失を確認中。 • Ignoring - **ignore loss** コマンドが実行されているため、キューのデータ消失は無視される。 • No Loss - キューではデータ消失は検出されていない。

admin who, sqt の出力カラムの説明

SQT スレッドは、ステーブルキューからトランザクションを読み込み、コミットされた順序で SQT リーダに渡します。リーダは DIST スレッドまたは DSI スレッドのいずれかです。

SQT は、処理中のトランザクションをメモリキャッシュに格納します。このテーブルの *Closed*、*Read*、*Open*、*Trunc*、および *Removed* 各カラムは、SQT キャッシュにあるトランザクションに適用されます。

表 17 : admin who, sqt で出力されるカラムの説明

カラム	説明
Closed	SQT キャッシュにあるコミットされたトランザクションの数。トランザクションはステーブルキューから読み取られ、処理を待機中である。
Read	処理されたが、まだキューから削除されていないトランザクションの数。
Open	SQT キャッシュにある、コミットまたはアボートされていないトランザクションの数。
Trunc	トランザクションキャッシュにあるトランザクションの数。*Trunc* は、*Closed*、*Read*、*Open* カラムの合計である。

カラム	説明
Removed	トランザクションを構成するメッセージがメモリから削除されたトランザクションの数。これは、SQT がラージトランザクションを処理するときに発生する。メッセージは、ステーブルキューから再度読み取られる。
Full	SQT がキャッシュのメモリを使い果たしたことを示す。これは、クローズされた、または読み取られたトランザクションが処理の待機中であれば、問題はない。SQT が頻繁にフルになる場合は、設定サイズの増加を検討する。これを行う場合は、"alter connection" を参照。
SQM Blocked	SQT がメッセージを読み取るために SQM で待機中の場合は 1。このステータスは、クローズされたトランザクションがないかぎり、一時的なものでなければならない。
First Trans	このカラムには、キューにある最初のトランザクションについての情報が含まれている。この情報は、トランザクションが終了されていないかどうかを調べるために使用できる。カラムには 3 つの情報がある。 • ST：O (open)、C (closed)、R (read)、または D (deleted) が後に表示される。 • Cmds：最初のトランザクションにあるコマンドの数が後に表示される。 • qid:最初のトランザクションのセグメント、ブロック、ローが後に表示される。
Parsed	解析されたトランザクションの数。
SQM Reader	SQM リーダハンドルのインデックス。
Change Oqids	オリジンキュー ID が変更されたことを示す。
Detect Orphans	孤立 (orphan) の検出中であることを示す。

パーミッション

このコマンドは、すべてのユーザが実行できます。

admin who_is_down

停止している Replication Server スレッドに関する情報を表示します。

構文

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

パラメータ

- **no_trunc –** Info カラムのサイズを 40 文字から 80 文字に増やします。このパラメータは、長いデータサーバ名やデータベース名を表示するときに役立ちます。

例

- **例 1 –**

```
admin who_is_down
```

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

使用法

- **admin who_is_down** の出力に含まれる *Spid* カラムは常に空です。実行されていないスレッドに対するプロセスはありません。
- **admin who_is_down** は、**admin health** によって Replication Server がサスペクト状態であることが示されたときに実行してください。このコマンドの出力には、サスペクト状態の原因の可能性のある “Connecting” 状態のスレッドは表示されません。
- このコマンドの出力の詳細については、「**admin who**」を参照してください。

パーミッション

このコマンドは、すべてのユーザが実行できます。

参照：

- admin health (60 ページ)
- admin who (107 ページ)
- admin who_is_up (127 ページ)

admin who_is_up

実行されている Replication Server スレッドに関する情報を表示します。

構文

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

パラメータ

- **no_trunc –** Info カラムのサイズを 40 文字から 80 文字に増やします。このパラメータは、長いデータサーバ名やデータベース名を表示するときに役立ちます。

例

- **例 1 –**

```
admin who_is_up
```

```
Spid  Name       State              Info
----  --------   ---------------    --------------------
  97  DIST       Active             103 LDS.pubs2
  98  SQT        Awaiting Wakeup    103:1 DIST LDS.pubs2
  96  SQM        Awaiting Message   103:1 LDS.pubs2
  68  SQM        Awaiting Message   103:0 LDS.pubs2
  89  DSI EXEC   Awaiting Message   106(1) SYDNEY_DS.pubs2sb
  91  DSI        Awaiting Command   106 SYDNEY_DS.pubs2sb
  21  DSI EXEC   Awaiting Message   101(1) TOKYO_DS.TOKYO_RSSD
  10  DSI        Awaiting Command   101 TOKYO_DS.TOKYO_RSSD
  16  DIST       Active             101 TOKYO_DS.TOKYO_RSSD
  17  SQT        Active             101:1 DIST TOKYO_DS.TOKYO
  15  SQM        Awaiting Message   101:1 TOKYO_DS.TOKYO_RSSD
  14  SQM        Awaiting Message   103:1 TOKYO_DS.TOKYO_RSSD
  30  REP AGENT  Awaiting Command   TOKYO_DS.TOKYO_RSSD
      USER
   4  DSI EXEC   Awaiting Message   104(1) TOKYO_DS.pubs2
   9  dAIO       Awaiting Message
  12  dREC       Active             dREC
  61  USER       Active             sa
   5  dALARM      Awaiting Wakeup
```

使用法

出力の詳細については、「**admin who**」を参照してください。

パーミッション

このコマンドは、すべてのユーザが実行できます。

参照：

- admin who (107 ページ)
- admin who_is_down (126 ページ)

allow connections

指定したデータベースの Replication Server をリカバリモードに設定します。

構文

```
allow connections
```

使用法

- 再ロードされたダンプからログレコードをリプレイするために、**allow connections** を実行します。
- Replication Server をスタンドアロンモードで起動し、ログをリプレイする各データベースに **set log recovery** を実行します。
- **allow connections** の実行後、Replication Server は指定したデータベースのリカバリモードで起動した RepAgent からの接続要求だけを受け入れます。これにより、Replication Server は現在のトランザクションの前にリプレイされたログレコードを確実に受け取ることができます。
- Replication Server をスタンドアロンモードで再起動し、先に **set log recovery** コマンドを実行しないで **allow connections** を実行すると、Replication Server はスタンドアロンモードからノーマルモードに移行します。
- リカバリ手順の詳細については、『Replication Server 管理ガイド 第 2 巻』を参照してください。

パーミッション

allow connections には、“sa” パーミッションが必要です。

参照：

- ignore loss (431 ページ)
- rebuild queues (434 ページ)
- set log recovery (447 ページ)

alter applied function replication definition

create applied function replication definition コマンドによって作成されたファンクション複写定義を変更します。

構文

```
alter applied function replication definition repdef_name
{with replicate function named 'func_name' |
```

```
add @param_name datatype[, @param_name datatype]… |
add searchable parameters @param_name[, @param_name]… |
send standby {all | replication definition} parameters}
[with DSI_suspended]
```

パラメータ

- **repdef_name –** 変更する適用ファンクション複写定義の名前です。
- **with replicate function named 'func_name' –** レプリケートデータベースで実行するストアドプロシージャの名前を指定します。*func_name* は、最大 255 文字の文字列です。
- **add –** 適用ファンクション複写定義にパラメータとそのデータ型を追加します。
- **@param_name –** 複写パラメータまたはサーチャブルパラメータのリストに追加するパラメータの名前です。各パラメータ名は、@ 文字で始まる必要があります。
- **datatype –** パラメータリストに追加するパラメータのデータ型です。データ型とその構文のリストについては、「データ型」を参照してください。Adaptive Server のストアドプロシージャとファンクション複写定義には、*text*、*unitext*、*rawobject*、*image* の各データ型のパラメータを含めることはできません。
- **add searchable parameters – where** 句 (**create subscription** または **define subscription** コマンド内) で使用できる追加パラメータを指定します。
- **send standby –** ウォームスタンバイアプリケーションで、スタンバイデータベースにファンクションのすべてのパラメータを送信するか (**send standby all parameters**)、複写定義で指定されたパラメータだけを送信するか (**send standby replication definition parameters**) を指定します。デフォルトは、**send standby all parameters** です。
- **with DSI_suspended –** スタンバイ DSI (存在する場合) と、各複写先 DSI スレッドをサスペンドできるようにします。Replication Server は、古いバージョンの複写定義のデータをすべてスタンバイデータベースまたはレプリケートデータベースに適用した後に、スタンバイデータベースまたはレプリケートデータベースの DSI スレッドをサスペンドします。

  Replication Server が DSI スレッドをサスペンドした後、ターゲットストアドプロシージャおよび任意のカスタムファンクション文字列を変更できます。DSI スレッドをレジュームすると、Replication Server は変更された複写定義を使用してプライマリの更新を複写します。

  次の場合、**with DSI_suspended** を使用する必要はありません。

  - 複写定義へのサブスクリプションがない。
  - カスタムファンクション文字列を変更する必要がない。

- レプリケートデータベースまたはスタンバイデータベースのストアドプロシージャを変更する必要がない。

注意： サイトバージョンが 1550 より古いレプリケート Replication Server からのサブスクリプションがある場合、その Replication Server のレプリケート DSI スレッドはサスペンドされません。

例

- **例 1 –** *@notes*、*@pubdate*、*@contract* の各パラメータを **titles_frep** ファンクション複写定義に追加します。

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

- **例 2 – titles_frep** ファンクション複写定義のサーチャブルパラメータのリストに、*@type* パラメータと *@pubdate* パラメータを追加します。

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

- **例 3 – titles_frep** ファンクション複写定義を変更してレプリケートデータベースで **newtitles** ストアドプロシージャとして複写されるようにし、**alter applied replication definition** の実行前に存在するプライマリデータがレプリケートデータベースに複写された後に、ターゲット DSI をサスペンドするように Replication Server に指示します。

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

使用法

- **alter applied function replication definition** は、既存の適用ファンクション複写定義を変更するときに使用します。複写パラメータやサーチャブルパラメータを追加したり、ウォームスタンバイに送信するパラメータを選択したりできます。また、レプリケートデータベースで実行するストアドプロシージャに別の名前を指定することもできます。
- **alter applied function replication definition** によって変更できるのは、**create applied function replication definition** コマンドを使用して作成した複写定義だけです。
- ファンクション複写定義を変更する場合、ファンクション複写定義に指定した名前、パラメータ、データ型が複写するストアドプロシージャと一致する必要があります。ファンクション複写定義で指定したパラメータだけが複写されます。
- 同じストアドプロシージャの複数のファンクション複写定義には、同じパラメータリストが必要です。新しいパラメータを追加すると、そのストアドプロ

シージャ用に作成されたすべてのファンクション複写定義に自動的に追加されます。

- **alter applied function replication definition** は、プライマリ Replication Server で実行します。
- 1 つの句の中で同じパラメータ名を 2 回以上指定することはできません。
- パラメータを追加するときは、ファンクション複写定義の分配に合わせて、**alter applied function replication definition** を調整するように Replication Server に指示する必要があります。また、ストアドプロシージャと複写定義も調整するように指示する必要があります。
  『Replication Server 管理ガイド 第 1 巻』の「複写テーブルの管理」の「複写定義の変更要求プロセス」を参照して複写定義を変更してください。
- レプリケートデータベースで実行するストアドプロシージャの名前を指定するには、**with replicate function named** 句を使用します。**create applied function replication definition** を参照してください。

  alter applied function replication definition の詳細については、『Replication Server 管理ガイド 第 1 巻』を参照してください。

パーミッション

alter applied function replication definition には、“create object” パーミッションが必要です。

参照：

- alter function string (188 ページ)
- alter replication definition (199 ページ)
- alter function replication definition (185 ページ)
- alter request function replication definition (210 ページ)
- create applied function replication definition (274 ページ)
- create request function replication definition (362 ページ)
- rs_send_repserver_cmd (735 ページ)
- rs_helprepversion (728 ページ)

alter auto partition path

自動でサイズ変更可能な Replication Server パーティションに許可するパーティションファイルサイズと最大サイズを変更します。

構文

```
alter auto partition path logical_name
{[auto expand size = size]
[max size = max_size]}
```

パラメータ

- ***logical_name*** – 既存の自動でサイズ変更可能な Replication Server パーティションに対する論理パーティションパス名。 名前は識別子の規則に従う必要があります。 *logical_name* は、**create auto partition path** コマンドおよび **drop auto partition path** コマンドでも、自動でサイズ変更可能なパーティションの指定に使用されます。
- ***size*** – Replication Server が自動でサイズ変更可能なパーティションで自動作成できるパーティションファイルに対して設定できるメガバイト単位のサイズ。
  - 最小値 - 16MB
  - 最大値 - 1,048,576MB
- ***max_size*** – 自動でサイズ変更可能なパーティションで自動作成されるすべてのパーティションファイルの合計サイズに対して設定できるメガバイト単位の上限。
  - 最小値 - 16MB
  - 最大値 - 2,147,483,647MB

例

- **例 1** – `auto_uxp` 論理パーティションパスで指定された、自動でサイズ変更可能なパーティションのパーティションファイルのサイズを 200MB に増やします。

  ```
  alter auto partition path auto_uxp auto expand size = 200
  ```

- **例 2** – `auto_winp` 論理パーティションパスで指定された、自動でサイズ変更可能なパーティションのパーティションファイルのサイズを 200MB に増やし、`auto_winp` 内のすべてのパーティションファイルの合計の制限を 204,800MB に増やします。

```
alter auto partition path auto_winp auto expand size = 200 max
size = 204800
```

使用法

- **alter auto partition path** を使用して変更できるのは、**create auto partition path** で作成した、自動でサイズ変更可能なパーティションのみです。 **alter auto partition path** を使用して、他のコマンドで作成したパーティションを変更することはできません。
- **alter auto partition path** を使用すると、新しく自動で作成されるパーティションファイルのサイズを増やすことができます。 これは、Replication Server でディスク領域を増やす必要があり、既存の論理パーティションパスの同じディスクにまだ使用できる領域がある場合に役立ちます。
- **auto expand size** および **max size** を変更する場合、Replication Server ではデフォルト値が設定されていないため、これらのパラメータの値を指定する必要があります。

パーミッション

alter auto partition path を使用するには、ディスクパーティションまたはオペレーティングシステムファイルは、“sybase” ユーザが所有するようにします。このユーザには、パーティションに対する読み込み/書き込みパーミッションが必要です。 “sybase” 以外のユーザには、このパーティションに対する読み込み/書き込みパーミッションを付与しないでください。

参照：

- admin auto_part_path (53 ページ)
- create auto partition path (284 ページ)
- drop auto partition path (405 ページ)
- rs_helppartition (714 ページ)
- admin disk_space (57 ページ)
- alter partition (196 ページ)

alter connection

データベースコネクションの属性を変更します。

構文

```
alter connection to data_server.database {
[for replicate table named [table_owner.]table_name
[set table_param [to] 'value']] |
set function string class [to] function_class |
```

```
set error class [to] error_class |
set replication server error class [to] rs_error_class |
set password [to] passwd |
set dsi_connector_sec_mech [to] hdbuserstore |
set log transfer [to] {on | off} |
set database_param [to] 'value' |
set security_param [to] 'value' |
set security_services [to] 'default']
set dataserver and database name [to] new_ds.new_db |
set trace [to] 'value' |
     set schemamap [with | without decluster] from data_server.db.
{from_schema | NULL} to {to_schema | NULL} |
     set sap_trim_len to {'on' | 'off'} |
     set sapsystemname to 'sid_adm_value' |
     set reblock_ddntf to {'on' | 'off'}}
```

パラメータ

- **data_server –** コネクションを変更するデータベースを保持するデータサーバです。
- **database –** 変更するコネクションを持つデータベースです。
- **for replicate table named –** レプリケートデータベースのテーブルの名前を指定します。*table_name* は、最大 200 文字の文字列です。*table_owner* は、テーブル名のオプション修飾子であり、テーブルの所有者を表します。実際のテーブルの所有者が、複写定義内に指定されたテーブル所有者と一致しない場合には、データサーバのオペレーションが失敗する可能性があります。
- **table_param – for replicate table name** で指定したテーブルに影響を与えるテーブルレベルのパラメータ。

  有効な値 **dsi_compile_enable** および **dsi_command_convert**。詳細は、表 18 : データベースコネクションに影響を与えるパラメータを参照してください。
- **function_class –** データサーバで使用するファンクション文字列クラスです。SAP Replication Server に用意されているデータベース接続用のファンクションクラスのリストについては、「ファンクション文字列変数の変更子」を参照してください。
- **error_class –** データベースエラーを処理するエラークラスです。 SAP Replication Server に用意されているデータベース接続用のエラークラスのリストについては、「エラークラスとファンクションクラス」を参照してください。
- **rs_error_class –** データベースの SAP Replication Server エラーを処理するエラークラスです。 Replication Server のエラークラスのリストについては、「エラークラスとファンクションクラス」を参照してください。
- **passwd –** データベース接続のログイン名に使用する、新しいパスワードです。ネットワークベースのセキュリティが有効になっていない場合は、パスワードを指定します。

- **dsi_connector_sec_mech –** DSI コネクタのセキュリティメカニズムを指定します。
- **log transfer on –** コネクションが RepAgent から SAP Replication Server にトランザクションを送信できるようにします。
- **log transfer off –** コネクションがプライマリデータベースの RepAgent からトランザクションを送信できないようにします。
- **schemamap from *data_server.db.from_schema* to *to_schema* –** プライマリスキーマまたは所有者 (*from_schema*) を複写スキーマまたは所有者 (*to_schema*) にマッピングします。

  各パラメータの意味は、次のとおりです。

  - *data_server.db* はデータサーバとプライマリデータを格納するデータベースを指定します。
  - *from_schema* はプライマリデータベースのスキーマを指定します。
  - *to_schema* はレプリケートデータベースのスキーマを指定します。

  注意： NULL はすべてのユーザを指定します。 *from_schema* と *to_schema* の両方に NULL を指定すると、*data_server.db* のマッピング関係が削除されます。

  DML または 異機種間 DDL 複写では、レプリケートデータベースがテーブル所有者を含むテーブル複写定義にサブスクライブしている場合は、そのテーブル複写定義のレプリケート所有者が有効になります。 schemamap を適用できるのは、テーブルにテーブルサブスクリプションがない場合か、テーブル複写定義にレプリケートテーブル所有者が含まれていない場合のみです。
- **schemamap with decluster from *data_server.db.from_schema* to *to_schema* –** スキーマのクラスタテーブルのクラスタ化解除を有効にして、プライマリスキーマまたは所有者 (*from_schema*) を複写スキーマまたは所有者 (*to_schema*) にマップします。
- **schemamap without decluster from *data_server.db.from_schema* to *to_schema* –** スキーマのクラスタテーブルのクラスタ化解除を無効にして、プライマリスキーマまたは所有者 (*from_schema*) を複写スキーマまたは所有者 (*to_schema*) にマップします。
- **database_param –** SAP Replication Server からのデータベース接続に影響を与えるパラメータです。
- **value –** オプションの新しい値を持つ文字列です。

  trace オプションを使用する場合、*value* の構文は、“*module*, *condition*,*[on|off]*”の形式になります。構文の説明は次のとおりです。

  - *module* - モジュールタイプを指定します。有効な値は *econn* です。
  - *condition* - 設定するトレース条件を指定します。
  - *on* または *off* - 目的の条件のステータスを指定します。

注意： **alter connection** コマンドの **trace** パラメータには空の文字列を指定できます。 次に例を示します。

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

接続するか、SAP Replication Server を再起動すると、空の文字列によって ExpressConnect トレース値が無効にされます。

表 18 : データベースコネクションに影響を与えるパラメータ

database_param	説明と *値*
async_parser	SAP Replication Server が RepAgent から非同期にコマンドを解析できるようにする。 次のセットに **async_parser** を設定します。 • **exec_prs_num_threads** を 2 に • **ascii_pack_ibq** を on に • **cmd_direct_replicate** を on に • **dist_cmd_direct_replicate** を on に デフォルト値は off **注意：** 非同期パーサを設定する前に、**smp_enable** が on であり、SAP Replication Server のホストマシンが解析用の追加スレッドをサポートできることを確認してください。 **ascii_pack_ibq** を on に設定する前に、SAP Replication Server のサイトバージョンを 1571 以降に設定する必要があります。 サイトのバージョンが 1571 だと、**async_parser** を on に設定しても、**exec_prs_num_threads**、**cmd_direct_replicate**、および **dist_cmd_direct_replicate** だけが設定されます。
ascii_pack_ibq	ASCII パッキングを使用して、インバウンドキューにパックされたコマンドが消費するステーブルキューの記憶領域を低減します。 デフォルト値は off **注意：** インバウンドキューで ASCII パッキングの恩恵を受けるには、SAP Replication Server で非同期パーサを有効にする必要があります。 **ascii_pack_ibq** を on に設定する前に、SAP Replication Server のサイトバージョンを 1571 以降に設定する必要があります。

database_param	説明と*値*
batch	SAP Replication Server がデータサーバにコマンドを送信する方法を指定する。 **batch** が “on” の場合、SAP Replication Server は複数のコマンドを 1 つのコマンドバッチとしてデータサーバに送信する。 **batch** が off の場合、SAP Replication Server はコマンドを 1 つずつデータサーバに送信する。 デフォルト値は on
batch_begin	**begin transaction** を他のコマンド (**insert** や **delete** など) と同じバッチで送信できるかどうかを示す。 デフォルト値は on
cmd_direct_replicate	エクゼキュータスレッドの **cmd_direct_replicate** を on に設定して、解析データをバイナリまたは ascii データと一緒にディストリビュータスレッドに直接送信します。ディストリビュータスレッドは、必要に応じて解析済みデータからデータを直接取得して処理できるので、バイナリデータの解析に費やされる時間を節約してレプリケーションパフォーマンスを向上させることができます。 デフォルト値は off
dist_cmd_direct_replicate	**dist_cmd_direct_replicate** を on に設定すると、DIST モジュールで内部解析データをメモリ内キャッシュから DSI に送信することができます。 デフォルト値は on **dist_cmd_direct_replicate** を off に設定すると、DIST モジュールはアウトバウンドキューからデータを DSI に送信します。
command_retry	失敗したトランザクションをリトライする回数。0 以上の値を指定する。 デフォルト値は 3
db_packet_size	ネットワークパケットの最大サイズ。データベースとの通信時に、ネットワークパケットの値はデータベースの許容範囲内である必要がある。 許容範囲: 512 バイト～ 2,147,483,647 バイト (レプリケートデータサーバの限度内) ASE の上限: 16,384 バイト デフォルト値: すべての Adaptive Server データベースに対して、512 バイトのネットワークパケット

database_param	説明と*値*
deferred_name_resolution	SAP Replication Server で遅延名前解決を有効にし、SAP ASE での遅延名前解決をサポートする。 遅延名前解決は SAP ASE 15.5 以降でのみ使用できる。 SAP Replication Server で遅延名前解決のサポートを有効にする前に、レプリケート SAP ASE で遅延名前解決がサポートされていることを確認する。 **alter connection** または **alter logical connection** と共に **deferred_name_resolution** を実行した後、コネクションをサスペンドして再開する。 デフォルト値は off
disk_affinity	次のパーティションを割り当てるための割り付けヒントを指定する。現在のパーティションが満杯になった場合に、次のセグメントの割り付け先となるパーティションの論理名を入力する。 デフォルト値は off
dist_sqt_max_cache_size	インバウンドキューの最大ステーブルキュートランザクション (SQT) キャッシュサイズ。デフォルトの 0 では、**sqt_max_cache_size** パラメータの現在の設定値が、コネクションの最大キャッシュサイズとして使用される。 デフォルト値は 0 32 ビット版 SAP Replication Server の場合: • 最小値 - 0 • 最大値 - 2147483647 64 ビット版 SAP Replication Server の場合: • 最小値 - 0 • 最大値 - 2251799813685247
dist_stop_unsupported_cmd	**dist_stop_unsupported_cmd** が on の場合、ダウンストリーム SAP Replication Server がコマンドをサポートしていないときに DIST が自動的にサスペンドする。 off の場合、DIST はサポートされていないコマンドを無視する。 **dist_stop_unsupported_cmd** パラメータの設定に関係なく、SAP Replication Server はバージョンの低い Replication Server サーバには送信できないコマンドの最初のインスタンスを検出したときに、エラーメッセージをログに必ず記録する。 デフォルト値は off

database_param	**説明と*値***
dsi_alt_writetext	レプリケートデータベースにラージオブジェクトの更新を送信する方法を制御する。 値は次のとおり。 • **dcany** - プライマリキーカラムを含む **writetext** コマンドを生成する。この設定により、インタフェースとして Enterprise Connect™ Data Access (ECDA) を使用して ASE 以外のレプリケートデータベースにデータを読み込むときに、フルテーブルスキャンを防ぐことができる。 • **off** - テキストポインタを含む Adaptive Server の **writetext** コマンドを生成する。 デフォルト値は off **注意：** ExpressConnect を使用して、ASE 以外のレプリケートデータベースに接続している場合は、**dsi_alt_writetext** データベースパラメータを設定する必要はありません。
dsi_bulk_copy	コネクションのバルクコピーイン機能を on または off にする。 **dynamic_sql** および **dsi_bulk_copy** が両方とも on の場合、SAP Replication Server は必要に応じてバルクコピーインを適用し、SAP Replication Server がバルクコピーインを使用できない場合は、動的 SQL を使用する。大量のデータを挿入する必要がある場合は、パフォーマンスを向上させるために **dsi_bulk_copy** を on にする。 デフォルト値は off **注意：** SAP IQ への Real-Time Loading (RTL) 複写を有効にする前に、**dsi_bulk_copy** を **off** に設定してください。

database_param	説明と*値*
dsi_bulk_threshold	トランザクション内の連続する **insert** コマンドの数。この数に到達すると、バルクコピーインを使用するために SAP Replication Server をトリガする。ステーブルキュートランザクション (SQT) は、大量の **insert** コマンドを検出すると、バルクコピーインを適用するかどうかを決定するために、指定された数の **insert** コマンドをメモリに保持する。これらのコマンドはメモリに保持されるため、この値を **dsi_large_xact_size** の設定値よりも大きい値に設定しないことをお奨めする。 SAP Replication Server は、SAP IQ への Real-Time Loading (RTL) Replication と Adaptive Server への High Volume Adaptive Replication (HVAR) に、**dsi_bulk_threshold** を使用する。 1 つのテーブルに対する **insert**、**delete**、または **update** の各オペレーションに対するコマンドの数が、コンパイル後に指定する数よりも小さい場合、RTL と HVAR はバルクインタフェースを使用せず、代わりに言語を使用する。 最小値：1 **注意：** RTL または HVAR を有効にすると、パフォーマンスに悪影響を及ぼすため、‘1’ に設定しないでください。 デフォルト値は 20 設定レベル:サーバ、データベース 設定では、サーバレベルでは **configure replication server** を、データベースレベルでは **alter connection** を使用する。 **注意：** RTL または HVAR 用に **dsi_bulk_threshold** を使用するには、**dsi_compile_enable** を ‘on’ に設定する必要があります。

database_param	説明と*値*
dsi_cdb_max_size	SAP Replication Server が HVAR または RTL 用に生成できる最終的な変更を保管するデータベースの最大サイズ (メガバイト単位)。 • デフォルト - 1024 • 最小値 - 0 • 最大値 - 2,147,483,647 HVAR では、SAP Replication Server は **dsi_cdb_max_size** をスレッショルドに使用して次のことを行う。 • 連続レプリケーションモードを使用して複写された大規模トランザクションを検出します。 • **dsi_cdb_max_size** よりも大きく、データベースの最終的な変更を必要とするグループに、小規模なコンパイル可能なトランザクションを分けることを停止します。 RTL では、SAP Replication Server は **dsi_cdb_max_size** を使用して、フルインクリメンタルコンパイルによって大規模なトランザクショングループをフラッシュする。
dsi_charset_convert	プライマリ SAP Replication Server とレプリケート SAP Replication Server 間でのデータと識別子の文字セット変換の処理方法を指定する。 このパラメータは、該当の DSI で適用されるすべてのデータと識別子に適用される。 値は次のとおり。 • on - プライマリ SAP Replication Server の文字セットからレプリケート SAP Replication Server 文字セットに変換する。文字セットに互換性がない場合は、DSI をエラーで停止する。 • allow - 文字セットに互換性がある場合に変換する。変換されていない更新も、すべてデータベースに適用する。 • off - 変換を行わない。このオプションは、異なるが互換性のある文字セットがあるときに、変換を一切実行しない場合に役立つ。サブスクリプションマテリアライゼーションでは、“off” に設定しても、“allow” を設定した場合と同様に動作する。 デフォルト値は on

database_param	説明と*値*
dsi_check_unique_key	直接ロードマテリアライゼーション時にプライマリデータベーステーブルのユニークキーをチェックする。 このパラメータをオン設定した場合、ユニークキーのないプライマリテーブルではアクティビティをいっさい実行できない。実行すると、キャッチアップフェーズでサブスクリプションが失敗する可能性があるため、サブスクリプションがアボートされ、エラーとマークされる。 サブスクリプションがアボートされたら、それを削除して作成しなおす必要がある。 サブスクリプションを再作成する前に (テーブルのトランケートと同様に) レプリケートテーブルをクリーンアップする必要がある。 デフォルト値: オン
dsi_cmd_batch_size	SAP Replication Server が、コマンドバッチ内に配置する最大バイト数。 デフォルト値は 8,192 バイト
dsi_cmd_prefetch	データサーバからの応答の待機中に、DSI がコマンドの次のバッチを事前構築することを許す。このため、DSI の効率が改善する。データサーバのパフォーマンスを高めるように調整する場合、この機能を使用するとパフォーマンスがさらに向上する可能性がある。 デフォルト値は off **dsi_compile_enable** を 'on' に設定すると、**dsi_cmd_prefetch** に設定した値は無視される。 ライセンス: Advanced Services オプションで個別にライセンス供与される。『管理ガイド第 2 巻』の「Replication Server - Advanced Services Option」を参照
dsi_cmd_separator	コマンドバッチ内でコマンドを区切るために使用する文字。 デフォルト値は改行文字 (¥n) **注意：** このパラメータは、DDL 生成スクリプトなどのスクリプトの実行によってではなく、対話型モードで更新する必要があります。スクリプトを実行して **dsi_cmd_separator** を再設定することはできません。

database_param	説明と*値*
dsi_command_convert	replicate コマンドの変換方法を指定する。変換の種類は次のオペレーションの組み合わせによって指定されます。 • **d** - delete • **i** - insert • **u** - update • **t** - truncate • **none** - オペレーションなし **dsi_command_convert** に対するオペレーションの組み合わせには、**i2none**、**u2none**、**d2none**、**i2di**、**t2none**、および **u2di** があります。 数字の 2 を入力する必要があります。変換前のオペレーションは 2 の前に、変換後のオペレーションは "2" の後ろにあります。例: • **d2none** - **delete** コマンドを複写しない。 • **i2di**、**u2di** - **insert** と **update** の両方を **delete** とその後の **insert** に変換する。これはオートコレクションと同等のオペレーション。 • **t2none** - **truncate table** コマンドを複写しない。 デフォルト値は**なし** このパラメータはテーブルレベルで設定することもできます。 設定では、データベースレベルでは **alter connection** を使用し、テーブルレベルの設定では **for replicate table named** 句を用いて **alter connection** を使用する。 **dsi_command_convert** を **none** に設定して、コネクションまたはテーブルの現在の **dsi_command_convert** 設定を削除します。
dsi_commit_check_locks_intrvl	DSI エグゼキュータスレッドが **rs_dsi_check_thread_lock** ファンクション文字列の実行と実行の間に待機するミリ秒 (ms) 数。並列 DSI とともに使用される。 デフォルト値は 1,000 ミリ秒 (1 秒) 最小値：0 最大値：86,400,000 ミリ秒 (24 時間)
dsi_commit_check_locks_log	警告メッセージがログに記録されるまでに、DSI エグゼキュータスレッドが **rs_dsi_check_thread_lock** ファンクション文字列を実行する回数。並列 DSI とともに使用される。 デフォルト値は 200 最小値：1 最大値：1,000,000

database_param	説明と*値*
dsi_commit_check_locks_max	トランザクションをロールバックし、リトライする前に、DSI エグゼキュータスレッドがレプリケートデータベースの他のトランザクションをブロックしているどうかをチェックする最大回数。並列 DSI とともに使用される。 デフォルト値は 400 最小値：1 最大値：1,000,000
dsi_commit_control	コミット制御について、SAP Replication Server が内部テーブルを使用して内部的に処理するか (on)、*rs_threads* システムテーブルを使用して外部的に処理するか (off) を指定する。 デフォルト値は on
dsi_compile _enable	サーバレベル、データベースレベル、またはテーブルレベルで RTL または HVAR を有効にするには、‘on’ に設定する。 デフォルト値は • off - サーバレベルとデータベースレベル。 SAP Replication Server はログ順、ローごとの連続変更複写を使用する。 • on - テーブルレベル 設定では、サーバレベルでは **configure replication server** を、データベースレベルでは **alter connection** を、テーブルレベルでは **for replicate table named** 句を用いて **alter connection** を使用する。 テーブル上のすべてのオペレーションをログ順に複写する必要があるトリガがテーブルにあるためコンパイルを使用できない場合のように、新しいロー変更を複写すると問題が発生する場合、問題のテーブルで **dsi_compile_enable** を ‘off’ に設定する。 **注意：** テーブルレベルで **dsi_compile_enable** を ‘off’ に設定する前に、サーバレベルまたはデータベースレベルで、**dsi_compile _enable** を ‘on’ に設定する。 **dsi_compile_enable** を ‘on’ に設定すると、**replicate_minimal_columns** と **dsi_cmd_prefetch** に設定した値は無視される。 サーバレベル、データベースレベル、またはテーブルレベルで **dsi_compile_enable** を実行した後、コネクションをサスペンドして再開する。 ライセンス: Advanced Services オプションで個別にライセンス供与される。『管理ガイド第 2 巻』の「Advanced Services Option」を参照する。

database_param	説明と*値*
dsi_compile_max_cmds	トランザクションのグループの最大サイズを、コマンド数で指定する。HVAR または RTL がコンパイルしている現在のグループで最大グループサイズに達すると、HVAR または RTL は新しいグループを開始する。 しかし、読み込むデータがなくなると、グループが最大コマンド数に達していなくも、HVAR または RTL はすぐに現在のステータスのグループをレプリケートデータベースに適用します。HVAR または RTL は、設定した制限にグループのサイズが到達するまでデータが届くのを待つことはありません。 RTL では、SAP Replication Server は **dsi_compile_max_commands** を **dsi_cdb_max_size** と一緒に用いて、フルインクリメンタルコンパイルでグループのフラッシュをトリガする。 HVAR では、Replication Server は **dsi_compile_max_commands** を **dsi_cdb_max_size** と一緒に用いて大規模なトランザクションを検出し、連続レプリケーションモードでこれを複写します。 デフォルト値は 10,000 最小値：100 サーバレベルでもデータベースレベルでもパラメータを設定できます。 設定では、サーバレベルでは **configure replication server** を、データベースレベルでは **alter connection** を使用する。 **注意：** **dsi_compile_max_cmds** を使用するには、**dsi_compile_enable** を ‘on’ に設定する必要がある。
dsi_compile_retry_threshold	リトライフェーズにおいて、HVAR または RTL 用にコンパイルされているトランザクションのグループの多くのコマンドのスレッショルド値を指定します。 失敗したトランザクションが含まれるグループのコマンド数に応じて、次のような結果になる。 • **dsi_compile_retry_threshold** の値よりも小さい場合、Replication Server は連続複写モードでグループの処理をリトライする。 • **dsi_compile_retry_threshold** の値よりも大きい場合、Replication Server は HVAR を使用してグループの処理をリトライするが、これには失敗したトランザクションを識別するためにさらにリトライが必要になることがある。 デフォルト値は 100 最小値：0 最大値：2,147,483,647

database_param	説明と*値*
dsi_connector_sec_mech	DSI コネクタのセキュリティメカニズムを指定する。 デフォルト値は default。 有効な値: コネクタによって異なる。 ExpressConnect for HANA データベースはこのパラメータを使用して、レプリケートデータサーバへの接続に使用するセキュリティメカニズムを調整する。 Replication Server はこのパラメータを検証せずにコネクタに渡す。 有効な値については、ExpressConnect for HANA データベースのマニュアルを参照する。
dsi_connector_type	コネクタの実装に使用するデータベースドライバテクノロジを指定する。 このパラメータを **dsi_dataserver_make** と一緒に用いて、コネクションに関連付けられたコネクタを識別する。 ASE または IQ に複写する場合は、このパラメータ値を *ctlib* に設定するか、Oracle に複写する場合は、この値を *oci* に設定する。 デフォルト値は default。 有効な値: cli、ctlib、jdbc、msnative、oci、odbc。
dsi_dataserver_make	接続するレプリケートデータベースを含むデータサーバタイプを指定する。 次のように設定する。 • **ase** - SAP ASE に複写 • **ase** - z/OS の IBM DB2 に複写 • **hdb** - SAP HANA データベースに複写 • **ase** - SAP IQ に複写 • **ase** - Microsoft SQL Server に複写 • **ora** - Oracle に複写 • **udb** - IBM UDB に複写 そのコネクションに関連付けるコネクタを識別するには、**dsi_dataserver_make** と **dsi_connector_type** を使用する。 **dsi_dataserver_make** をデータベースレベルで設定できる。 このパラメータを指定しない場合、Replication Server は、データベース接続のファンクション文字列のクラス名からデータサーバタイプを推測する。 ファンクション文字列のクラスがカスタマイズされている場合、Replication Server はデータサーバタイプを推測できないため、デフォルトを 'ASE' に設定する。

database_param	説明と*値*
dsi_do_decompression	LOB データを圧縮解除するかどうかを指定する。 次のように設定する。 • **on** - LOB データを圧縮解除する • **off** - (デフォルト) LOB データを圧縮解除しない ASE 以外のレプリケートデータサーバが、ASE から送られた圧縮 LOB データを処理できるよう、`rs_ase_to_hanadb`、`rs_ase_to_iq`、`rs_ase_to_msss`、`rs_ase_to_oracle`、および `rs_ase_to_udb` 接続プロファイルでは、**dsi_do_decompression** は on に設定される。
dsi_exec_request_sproc	プライマリ Replication Server の DSI で、要求ストアドプロシージャをオンまたはオフにする。 デフォルト値は on
dsi_fadeout_time	DSI コネクションがクローズされるまでのアイドル時間 (秒単位)。この値を “-1” にすると、コネクションは永久にクローズしない。 デフォルト値は 600 秒

database_param	説明と*値*
dsi_incremental_parsing	on に設定すると、High Volume Adaptive Replication (HVAR)、Real-Time Loading (RTL)、または DSI バルクコピーインを有効にしたとき、DSI スケジューラスレッドによるインクリメンタル解析が有効になる。 **注意：** **dsi_incremental_parsing** は **dsi_compile_enable** または **dsi_bulk_copy** も on に設定されている場合にのみ反映されます。 そうでない場合は、**dsi_incremental_parsing** は無視されます。 デフォルト値: オン インクリメンタル解析をサポートするには、プライマリとレプリケートの Replication Server がバージョン 15.7.1 SP100 以降であることが必要。 **dsi_incremental_parsing** は次のコマンドとともに使用する。 • **alter connection** および **create connection** - 接続レベルで指定のデータベースに対してインクリメンタル解析を有効にする。 このパラメータ設定の変更はただちに反映される。 • **configure replication server** - サーバレベルですべての接続に対してインクリメンタル解析を有効にする。 このパラメータの変更を反映させるには、接続をサスペンドして再開する必要がある。 接続レベルの設定はサーバレベルの設定に優先する。 『Replication Server 管理ガイド 第 2 巻』で以下を参照する。 • インクリメンタル解析 • Adaptive Server への High-Volume Adaptive Replication • DSI バルクコピーイン 『Replication Server 異機種間複写ガイド』の「Real-Time Loading ソリューション」を参照する。
dsi_ignore_underscore_name	トランザクションパーティショニングルールが “name” に設定されているときに、Replication Server がアンダースコアで始まるトランザクション名を無視するかどうかを指定する。値は "on" または "off"。 デフォルト値は on

database_param	説明と*値*
dsi_isolation_level	トランザクションの独立性レベルを指定する。 ANSI 標準および Adaptive Server でサポートされている値は、次のとおり。 • 0 - 個々のトランザクションで書き込まれたデータが実際のデータであることを保証する。 • 1 - ダーティリードを防止し、個々のトランザクションで書き込まれたデータが実際のデータであることを保証する。 • 2 - 繰り返し不可能読み出しとダーティリードを防止し、個々のトランザクションで書き込まれたデータが実際のデータであることを保証する。 • 3 - 幻ロー、繰り返し不可能読み出し、ダーティリードを防止し、個々のトランザクションで書き込まれたデータが実際のデータであることを保証する。 • デフォルト値 - レプリケートデータサーバのトランザクション独立性レベルを使用する。 他の独立性レベルをサポートするデータサーバも、**rs_set_isolation_level** ファンクション文字列を使用することによってサポートされる。 サポートは ANSI 標準のみに限定されない。 Replication Server は、レプリケートデータサーバが使用できる任意の独立性レベルをサポートできる。
dsi_keep_triggers	データベース内の複写トランザクションに対してトリガを起動するかどうかを指定する。 この値を "off" に設定すると、Replication Server によって Adaptive Server データベース内のトリガがオフに設定されるため、コネクション上でトランザクションが実行されてもトリガは起動されない。 スタンバイデータベースを除くすべてのデータベースで "on" に設定する。 デフォルト値は on (スタンバイデータベースを除く)
dsi_large_xact_size	ラージトランザクションと見なされるまでに 1 つのトランザクション内で許可されるコマンドの数。 デフォルト値: 100 最小値: 4 最大値: 2,147,483,647 **dsi_compile_enable** を on にすると、**dsi_large_xact_size** は Replication Server で無視される。

database_param	説明と*値*
dsi_max_cmds_in_batch	出力コマンドのバッチ処理の対象にできるソースコマンドの最大数を定義します。 パラメータの変更を反映させるには、コネクションをサスペンドして再開する必要があります。 範囲：1 - 1000 デフォルト値は 100
dsi_max_cmds_to_log	1 つのトランザクションで例外ログに書き込むコマンドの数。 デフォルト値は-1 (すべてのコマンド) 有効な値 0 から 2147483647
dsi_max_xacts_in_group	グループ化できるトランザクションの最大数を指定する。大きい値を指定するほど、レプリケートデータベースでのデータ遅延時間が短縮される。値の範囲：1 - 1000 デフォルト値は 20 **dsi_compile_enable** が on の場合、このパラメータは無視される。
dsi_max_text_to_log	失敗したトランザクション内の各 **rs_writetext** ファンクションの例外ログに書き込むバイト数。このパラメータを変更して、*text*、*unitext*、*image* または *rawobject* のラージカラムの処理を伴うトランザクションによって RSSD またはそのログが満杯になるのを防ぐ。 デフォルト値は-1 (*text*、*unitext*、*image*、または *rawobject* のすべてのカラム)
dsi_non_blocking_commit	Replication Server がコミット後にメッセージを保存しておく時間 (分単位)。値に 0 を指定すると、非ブロッキングコミットは無効になる。 **注意：** このパラメータを **alter connection** で使用して、スタンバイ環境でアクティブデータベースコネクションを設定することはできません。 デフォルト値は 0 最大値：60
dsi_num_large_xact_threads	ラージトランザクションで使用するために予約されている並列 DSI スレッドの数。最大値は、**dsi_num_threads** の値から 1 を引いた値。 デフォルト値は 0
dsi_num_threads	使用する並列 DSI スレッドの数。最大値は 255。 デフォルト値は 1

database_param	**説明と*値***
dsi_partitioning_rule	利用可能な並列 DSI スレッド間でトランザクションを分割するために DSI が使用する、1 つ以上のパーティショニングルールを指定する。指定できる値は **origin**、**origin_sessid**、**time**、**user**、**name**、**none** のいずれか。 詳細については、『Replication Server 管理ガイド 第 2 巻』の「パフォーマンスチューニング」の「並列 DSI スレッドの使用」の「パーティショニングルール: 競合を減らして並列処理を増やす」を参照してください。 デフォルト値はなし **dsi_compile_enable** が on の場合、このパラメータは無視される。
dsi_proc_as_rpc	Replication Server がストアドプロシージャの複写を適用する方法を指定します。 • リモートプロシージャコール (RPC) の呼び出しを使用するには、on に設定します。 • 言語呼び出しを使用するには、off に設定します。 デフォルト値は off Adaptive Server がレプリケートデータベースである場合は、**dsi_proc_as_rpc** を on または off にできます。 レプリケートデータベースが Oracle である場合は、次のように設定する。 • ExpressConnect for Oracle (ECO) を使用している場合は on に設定する。 ECO では、RPC を使用したストアドプロシージャの複写のみがサポートされている。 Replication Server から Oracle データベースへのコネクションを作成するときに Oracle ECO 接続プロファイルのいずれかを使用する場合、Replication Server はデフォルトで **dsi_proc_as_rpc** を on に設定する。 『SAP® Replication Server® Options ExpressConnect for Oracle 設定ガイド』の「ExpressConnect for Oracle の設定」を参照する。

database_param	説明と*値*
dsi_quoted_identifiers	データサーバインタフェース (DSI) での引用符付き識別子のサポートを有効化または無効化する。 • **on** - 複写定義でテーブルを引用としてマークしている場合、または Adaptive Server 15.7 の RepAgent から送信された LTL がテーブルを引用としてマークしている場合に、引用符付き識別子を有効化する。 • **off** - 引用符付き識別子のサポートを無効化する。 • **always** - プライマリデータベースの設定または複写定義の設定にかかわらず、識別子を常に二重引用符で囲む。 **always** オプションは特定のテーブルまたはすべてのテーブルに対して使用できる。 デフォルト値: **off** **dsi_quoted_identifier** は、テーブルレベル、接続レベル、またはサーバレベルで設定できる。 テーブルレベルの **dsi_quoted_identifier** の設定は、既存の接続レベルの **dsi_quoted_identifier** の設定より優先される。また、接続レベルの設定はサーバレベルの **dsi_quoted_identifier** の設定より優先される。 その方法については、『SAP Replication Server 管理ガイド 第 1 巻』>「レプリケートテーブルの管理」>「複写定義の作成」>「引用符付き識別子」を参照する。 Adaptive Server データベース接続に対して **dsi_quoted_identifier** を **on** に設定する場合は、レプリケートするストアドプロシージャに二重引用符を含めないようにする必要がある。 含まれている場合、DSI スレッドが停止する。 このパラメータは、次の場合に有効にする。 • データサーバに引用符付き識別子を転送できるコネクションを作成または修正する。 **create connection** または **alter connection** を使用して **dsi_quoted_identifier** を設定する。 **admin config** コマンドを使用して、**dsi_quoted_identifier** の値を確認する。 • 複写定義で識別子の引用符付きとしてのマーク付けをサポートする。 引用符付きの識別子は、混合バージョン環境ではサポートされない。 引用符付き識別子の複写を成功させるには、プライマリ Replication Server とレプリケートデータサーバに接続する Replication Server のバージョンを 15.2 以降にする。 ただし、ルート上の中間 Replication Server は、古いバージョンでもかまわない。

database_param	**説明と*値***
dsi_replication	DSI によって適用されたトランザクションを、トランザクションログ内で複写対象としてマーク付けするかどうかを指定する。 **dsi_replication** を “off” に設定すると、DSI は Adaptive Server データベース内で **set replication off** を実行し、DSI が実行するトランザクションのログレコードに、Adaptive Server が複写情報を追加するのを防ぐ。これらのトランザクションは、メンテナンスユーザによって実行されるため、(スタンバイデータベースがある場合を除いて) 通常は、これ以上複写されることはない。そのため、このパラメータを “off” に設定すると、不必要な情報がトランザクションに書き込まれるのを防ぐことができる。 レプリケートデータベース用のウォームスタンバイアプリケーション内のアクティブなデータベースや複写統合レプリケートアプリケーションモデルを使用するアプリケーションの場合は、**dsi_replication** を “on” に設定する必要がある。 デフォルト値は on (ウォームスタンバイアプリケーション内のスタンバイデータベースの場合は “off”)

database_param	説明と*値*
dsi_replication_ddl	トランザクションを元のデータベースに複写するかどうかを指定することによって双方向複写をサポートする。 デフォルト値: **off** **dsi_replication_ddl** が on に設定されている場合、DSI はレプリケートデータベースに **set replication off** を送信し、システムログの後続の DDL トランザクションが複写されないようにマーク付けするようレプリケートデータベースに指示する。この結果、これらの DDL トランザクションが元のデータベースに複写されなくなり、双方向 MSA 複写環境での DDL トランザクションの複写が可能になる。 さらに、**dsi_replication_ddl** は、SAP Replication Server が DDL、**select into**、および request function コマンドをレプリケートデータベースで適用する方法を制御する。 **dsi_replication_ddl** は次のように設定する。 • **off** - SAP Replication Server は、切断し、プライマリでコマンドを実行するユーザとして再接続することで、コマンドを適用する。 元のユーザとして接続するとき、DSI は非同期要求ファンクションをサポートするため、**set replication off** を送信しない。 したがって、変更はレプリケートデータベースから再度複写される。 MSA システムでは、データベース repdef で DDL 複写を除外することで、送信元に DDL が再複写されるのを防ぐ。 • **on** - SAP Replication Server は、プライマリデータベースでコマンドを実行するユーザに対して **set session authorization** パーミッションを付与することで、コマンドを適用する。 DSI は通常、**set replication off** をレプリケートデータベースに送信するため、変更は最複写されない。 **注意：** **dsi_replication_ddl** を **off** に設定する場合、DSI はプライマリで呼び出し元と同じパスワードを使用して接続しようとするため、パスワードの同期が必要になる。 **dsi_replication_ddl** を **on** に設定する場合はパスワードの同期は必要ないが、メンテナンスユーザは **set session authorization** パーミッションが必要である。

database_param	説明と*値*

dsi_retry	HVAR、RTL、動的 SQL、DSI バルクコピーイン、並列 DSI、または連続ログ順言語複写モードで複写を継続できなかったら複写をサスペンドするかどうかを指定する。
	dsi_retry を以下のとおりに設定する。
	• 0 - デフォルト設定。HVAR、RTL、並列 DSI、動的 SQL または DSI バルクコピーインでトランザクションの適用に失敗すると、Replication Server がトランザクションの再適用を再試行してから自動的に連続複写モードに切り替わる。
	• 1 - 次の場合に複写を停止する。
	  • HVAR または RTL がコンパイル可能なトランザクションの適用に失敗
	  • 並列 DSI、動的 SQL、DSI バルクコピーイン、または連続ログ順複写モードがトランザクションの適用に失敗
	• 2 - 次の場合に複写を停止する。
	  • 失敗したトランザクションを含むグループのコマンド数が **dsi_compile_retry_threshold** の値より小さく、HVAR または RTL がコンパイル可能なトランザクションの適用に失敗
	  • 並列 DSI、動的 SQL、DSI バルクコピーイン、または連続ログ順複写モードがトランザクションの適用に失敗
	dsi_retry は次のコマンドとともに使用する。
	• **alter connection** および **create connection** - 指定のデータベースについて接続レベルで複写をサスペンドする。
	• **configure replication server** - すべての接続についてサーバレベルで複写をサスペンドする。
	• **create alternate connection** - マルチパス環境で指定された複写パスについて複写をサスペンドする。
	接続レベルの設定は、サーバレベルの設定よりも優先される。
	『トラブルシューティングガイド』の次の項目を参照する。
	• **dsi_retry** の使用が必要になる可能性のあるシナリオについては、「高速モードで複写を継続できない」
	• **dsi_retry** を使用した場合に表示される可能性があるエラーメッセージの例については、「dsi_retry の設定オプションとエラーメッセージの例」
	高速複写モードの詳細については、次を参照する。
	• 『管理ガイド第 2 巻』の「並列 DSI スレッド」、「DSI バルクコピーイン」、「動的 SQL で強化された Replication Server のパフォーマンス」、「Adaptive Server への High Volume Adaptive Replication」、「リトライメカニズムの強化」

database_param	**説明と*値***
	• 『異機種間複写ガイド』の「Real-Time Loading ソリューション」および「リトライメカニズムの強化」
dsi_row_count_validation	同期されていないテーブルローがあり、デフォルトのエラーアクションとメッセージをバイパスする場合、**dsi_row_count_validation** を **off** に設定してローカウントの検証を無効にできる。 デフォルト値は **on** で、ローカウントの検証を有効にする。 特定のコネクションに対して **dsi_row_count_validation** を設定した場合は、データベースコネクションをサスペンドして再開する必要はありません。パラメータはただちに有効になります。ただし、新しい設定は、このコマンドを実行した後で Replication Server が処理する複写オブジェクトのバッチに影響します。設定の変更は Replication Server が現在処理している複写オブジェクトのバッチには影響しません。
dsi_rs_ticket_report	**rs_ticket_report** ファンクション文字列を呼び出すかどうかを指定する。**dsi_rs_ticket_report** を on に設定すると、**rs_ticket_report** ファンクション文字列が呼び出される。 デフォルト値は on

database_param	説明と*値*
dsi_serialization_method	一貫性を保ちながら、トランザクションを開始できるタイミングを決定するために使用するメソッドを指定する。どの場合でもコミット順は保持される。 これらのメソッドは並列処理量の多い順になる。並列処理が多いほど、レプリケートデータベースに適用されるときに並列トランザクション間の競合が増える可能性がある。競合を減らすには、**dsi_partition_rule** オプションを使用する。 • **no_wait** - 他のトランザクションの状態に関係なく、トランザクションが準備でき次第すぐに開始できることを指定する。 • **wait_for_start** - 開始直前にコミットするようにスケジュールされているトランザクションが開始した直後に、トランザクションを開始できることを指定する。 • **wait_for_start** - 直前にコミットするようスケジュールされているトランザクションの準備が終了するまでは、トランザクションを開始できないよう指定する。 • **wait_after_commit** - 直前にコミットするようにスケジュールされているトランザクションのコミットが完了するまで、トランザクションが待機することを指定する。 **注意：** **dsi_commit_control** を “on” に設定している場合は、 **dsi_serialization_method** は **no_wait** にのみ設定できる。 以下のオプションは、Replication Server の古いバージョンとの下位互換性のためにのみ維持されている。 • **none** - **wait_for_start** と同じ。 • **single_transaction_per_origin** - **dsi_partitioning_rule** が **origin** に設定された **wait_for_start** と同じ。 **注意：** **isolation_level_3** 値は逐次化メソッドとしてサポートされなくなりましたが、**dsi_serialization_method** を **wait_for_start** に設定し、**dsi_isolation_level** を 3 に設定した場合と同じです。 デフォルト値は **wait_for_commit**

database_param	説明と*値*
dsi_sqt_max_cache_size	アウトバウンドキューの SQT (ステーブルキュートランザクションインタフェース) キャッシュサイズの最大量 (バイト単位)。 デフォルトの “0” は、コネクションの最大キャッシュサイズとして **sqt_max_cache_size** の現在の設定を使用することを示す。 デフォルト値は 0 32 ビット版 Replication Server の場合： • 最小値 - 0 • 最大 2GB (2,147,483,648 バイト) 64 ビット版 Replication Server の場合： • 最小値 - 0 • 最大 - 2 ペタバイト (2,251,799,813,685,247 バイト)
dsi_stage_all_ops	SAP Replication Server と SAP IQ InfoPrimer を設定する場合に、指定したテーブルがコンパイルされないようにする。 緩やかに変化する次元 (SCD) のテーブルなどのように、テーブル履歴を保存する必要がある場合、**dsi_stage_all_ops** を on に設定する。 『異機種間複写ガイド』の「**dsi_stage_all_ops**」を参照する。
dsi_text_convert_multiplier	レプリケートサイトでの *text* または *unitext* データ型カラムの長さを変更する。文字セット変換によって *text* または *unitext* データ型カラムが拡張または縮小される場合、**dsi_text_convert_multiplier** を使用する。Replication Server は、プライマリサイトの *text* または *unitext* データの長さに **dsi_text_convert_multiplier** の値を乗算することによって、レプリケートサイトでの *text* または *unitext* データの長さを判別する。値の型は *float*。 • 文字セット変換によって *text* または *unitext* データ型カラムのサイズが拡張される場合は、**dsi_text_convert_multiplier** に 1.0 以上の値を指定する。 • 文字セット変換によって *text* または *unitext* データ型カラムのサイズが縮小される場合は、**dsi_text_convert_multiplier** に 1.0 以下の値を指定する。 デフォルト値は 1

database_param	説明と*値*
dsi_timer	**dsi_timer** 設定パラメータを使用して、プライマリデータベースでトランザクションがコミットされる時刻と、スタンバイデータベースまたはレプリケートデータベースでトランザクションがコミットされる時刻との遅延を指定する。Replication Server は、この遅延期間が終了した後、アウトバウンドキューのトランザクションをコミット順に処理する。 **alter connection** または **alter logical connection** と共に **dsi_timer** を実行した後、コネクションをサスペンドして再開する。 遅延を hh:mm のフォーマットで指定する。 • 最大値：24 hours • デフォルト値は 00:00 で、遅延がないことを意味する。 **注意：** Replication Server は、プライマリデータベースでの RepAgent または Replication Agent と、**dsi_timer** を実行する DSI コネクションの Replication Server との時差をサポートしていません。
dsi_top1_enable	Adaptive Server データベースに対して、ユニークキーを持たないテーブルの複写を有効にするには、**dsi_top1_enable** を on に設定する。 テーブルがユニークキーを持たない場合、ローに同じ値を持つローが 2 つ以上存在することがある。 ただし、オペレーションを適用するユニークローが見つからない場合、DSI は停止する。 **Adaptive Server select 文の上の *unsigned_integer*** 句で *unsigned_integer* を 1 に設定することで、**dsi_top1_enable** パラメータは DSI に対して、複数の同様のローの最初のインスタンスのみを選択して更新するように命令する。 デフォルトは off。 すべての複写接続に対してパラメータを設定するには、**configure replication server** を使用する。 Adaptive Server 以外のレプリケートデータベースへの接続がある場合は、**dsi_top1_enable** を on に設定するために、**configure replication server** を使用しない。 代わりに **alter connection** を使用して、各 Adaptive Server レプリケートデータベース接続に個別にパラメータを設定する。 Replication Server は、次の場合、ユニークキーを持たないテーブルの複写をサポートしない。 • テーブルにカスタマイズされたファンクション文字列がある。 • テーブルにテーブル複写定義がある。 • トランザクションが、プライマリデータベースで **where** 句に **like** オプションを使用した更新および削除を使用する。 • レプリケートデータベースが Adaptive Server 以外である。

database_param	説明と*値*
dsi_xact_group_size	1 つにグループ化されたトランザクションに配置する最大バイト数 (ステーブルキューオーバヘッドを含む)。グループ化されたトランザクションとは、DSI が単一のトランザクションとして適用する複数のトランザクションのことである。1 はグループ化が行われないことを意味する。 **dsi_xact_group_size** を最大値に設定し、**dsi_max_xacts_in_group** でグループ内のトランザクション数を制御することを推奨する。 **注意：** このパラメータは、Replication Server バージョン 15.0 以降では使用されなくなっていますが、旧バージョンの Replication Server との互換性を保つために保持されています。 最大値：2,147,483,647 デフォルト値は 65,536 バイト **dsi_compile_enable** が on の場合、このパラメータは無視される。
dump_load	コーディネートダンプを有効にする目的でのみ、レプリケートサイトで “on” に設定する。詳細については、『Replication Server 管理ガイド 第 2 巻』を参照。 デフォルト値は off
dynamic_sql	コネクションの動的 SQL 機能を有効または無効にする。このパラメータを on に設定した場合にのみ、動的 SQL 関連の他の設定パラメータが有効になる。 **注意：** **dynamic_sql** と **dsi_bulk_copy** の両方を on にすると、DSI によってバルクコピーインが適用される。バルクコピーインが使用されない場合は、動的 SQL が使用される。 デフォルト値は off **注意：** SAP IQ への Real-Time Loading (RTL) 複写を有効にする前に、**dsi_bulk_copy** を **off** に設定してください。
dynamic_sql_cache_management	コネクションの動的 SQL キャッシュを管理する。 値： • mru - **dynamic_sql_cache_size** に達したら、新しい動的文に割り付けることができるように、古い動的 SQL 準備文の割り付けを解除することを指定する。 • fixed - **dynamic_sql_cache_size** に達したら、新しい動的 SQL 文の割り付けを停止することを指定する。 デフォルト値は fixed

database_param	説明と*値*
dynamic_sql_cache_size	Replication Server が、コネクションの動的 SQL 文を使用できるデータベースオブジェクトの数を見積もることができるようにする。**dynamic_sql_cache_size** を使用すると、データサーバのリソース要求を制限できる。 デフォルト値は 100 最小値：1 最大値：65,535
exec_cmds_per_timeslice	CPU を解放する前に、LTL または RepAgent エグゼキュータスレッドが処理できる LTL コマンドの数を指定する。この値を大きくすると、RepAgent エグゼキュータスレッドが CPU リソースをより長時間制御でき、RepAgent から Replication Server へのスループットが向上する。 このパラメータは、**alter connection** を使用してコネクションレベルで設定する。 『Replication Server 管理ガイド第 2 巻』の「パフォーマンスチューニング」の「RepAgent エグゼキュータが処理できるコマンド数の制御」を参照してください。 デフォルト値は 2,147,483,647 最小値：1 最大値：2,147,483,647

database_param	説明と*値*
exec_max_cache_size	エグゼキュータコマンドキャッシュに割り当てるメモリ量を指定する。 デフォルト値は 32 ビットと 64 ビットの Replication Server で 1,048,576 バイト 32 ビット版 Replication Server の場合： • 最小値 - 0 • 最大値 - 2,147,483,647 バイト 64 ビット版 Replication Server の場合： • 最小値 - 0 • 最大値 - 2,251,799,813,685,247 バイト 設定では、Replication Server へのすべてのデータベースコネクションに **configure replication server** を、特定のデータベースコネクションに **alter connection** を使用する。 『Replication Server 管理ガイド 第 2 巻』の「パフォーマンスチューニング」の「チューニングパラメータの使用についての注意事項」の「エグゼキュータコマンドキャッシュ」を参照してください。
exec_nrm_request_limit	正規化待機中のプライマリデータベースからのメッセージ用に使用可能なメモリ量を指定する。 **configure replication server** で **nrm_thread** を 'on' に設定してから、**exec_nrm_request_limit** を使用する。 最小値：16,384 バイト 最大値：2,147,483,647 バイト デフォルト値： • 32 ビット版 - 1,048,576 バイト (1MB) • 64 ビット版 - 8,388,608 バイト (8MB) **exec_nrm_request_limit** の設定を変更した後、Replication Agent をサスペンドして再開する。 ライセンス：Advanced Services オプションで個別にライセンス供与されます。『Replication Server 管理ガイド 第 2 巻』の「パフォーマンスチューニング」の「Advanced Services Option」を参照してください。

database_param	説明と*値*
exec_prs_num_threads	プライマリデータベースから特定のコネクションの複数パーサスレッドを開始して非同期解析機能を有効にし、コネクションの非同期パーサスレッドの数を指定する。 デフォルト値は 0 (非同期パーサを無効にする) 最小値：0 最大値：20 **注意：** 非同期パーサを設定する前に、**smp_enable** が on であり、Replication Server のホストマシンが解析用の追加スレッドをサポートできることを確認してください。
exec_sqm_write_request_limit	インバウンドキューへの書き込み待ちメッセージ用に使用可能なメモリ量を指定する。 デフォルト値は 1MB 最小値：16KB 最大値：2GB
md_sqm_write_request_limit	アウトバウンドキューへの書き込み待ちメッセージ用にディストリビュータが使用可能なメモリ量を指定する。 **注意：** Replication Server 12.1 の場合、**md_source_memory_pool** は **md_sqm_write_request_limit** で置換されます。**md_source_memory_pool** は旧式の Replication Server との互換性のために保持されます。 デフォルト値は 1MB 最小値：16KB 最大値：2GB
parallel_dist	ディストリビュータスレッド (DIST) での並列処理を有効にして、複写のスループットを向上させる。 デフォルトは off。 このパラメータの変更を反映させるには、影響を受けるデータベース接続をサスペンドして再開する必要がある。

database_param	説明と*値*
parallel_dsi	並列 DSI スレッドの簡易設定を行う。 **parallel_dsi** を **on** に設定すると、自動的に次のように設定される。 • **dsi_num_threads** は 5 • **dsi_num_large_xact_threads** は 2 • **dsi_serialization_method** は **wait_for_commit** • **dsi_sqt_max_cache_size** は 1 メガバイト (32 ビットプラットフォーム)、20 メガバイト (64 ビットプラットフォーム) **parallel_dsi** を **off** に設定すると、これらの並列 DSI パラメータがそれぞれのデフォルト値にリセットされる。 **parallel_dsi** を **on** に設定した後で、並列 DSI の各設定パラメータを個別に設定して複写のパフォーマンスを微調整できる。 デフォルト値: **off**
reblock_ddntf	(DB2 UDB データベース上の SAP Business Suite からサポートされている任意のデータベースへの複写の場合のみ) Replication Server でレプリケートデータベースの各 FIELDS 列の長さを、最後のロー以外は 32768 に調整するかどうかを指定する。 値は on または off。 デフォルトは on。
rep_as_standby	データベースが **sp_reptostandby** を使用してマーク付けされ、**rep_as_standby** が on の場合、データベース複写定義によって扱われないテーブル複写定義のあるテーブルは複写される。 テーブルを複写するには、次のように設定する。 • **rep_as_standby** は on • **send maint xacts to replicate** は false • **send warm standby xacts** は true デフォルト値は off

database_param	説明と*値*
replicate_minimal_columns	Replication Server がすべてのトランザクションのすべての複写定義カラムを送信するか、レプリケートデータベースで更新オペレーションまたは削除オペレーションを実行する必要があるカラムだけを送信するかを指定する。 値は "on" または "off"。 複写定義に **replicate minimal columns** 句が含まれない場合、または、複写定義がまったくない場合、Replication Server はこのコネクションレベルのパラメータを使用する。 **注意：** 複写定義に **replicate all columns** 句が含まれ、かつ **replicate minimal columns** コネクションプロパティが 'on' に設定されている場合、そのコネクションは最少数のカラムをレプリケートします。 ローのカラム値に変更が加えられていない場合でも、ターゲットデータベースにカラムをすべてレプリケートするには、DSI コネクションの **replicate minimal columns** 値を "off" に設定します。 **admin config** を使用して、**replicate_minimal_columns** 設定情報を表示できる。 **dsi_compile_enable** を on に設定すると、**replicate_minimal_columns** に設定した値は無視される。 『Replication Server 管理ガイド 第 2 巻』の「パフォーマンスチューニング」の「replicate minimal columns 句と動的 SQL」を参照してください。
sapsystemname	(SAP Business Suite からの複写の場合のみ) SAP システム名 (<*sid*>) の環境変数を指定する。
sap_trim_len	(Oracle データベース上の SAP Business Suite からサポートされている任意のデータベースへの複写の場合のみ) 次の場合に、Replication Server で複写時に最初の 2 バイトをトランケートする必要があるかどうかを指定する。 • カラムのアプリケーションデータ型 (DDIC 型) が RAW または LRAW であり、かつ • DDIC 型の長さが 255 バイトを超えている 値は on または off。 デフォルト値: off
save_interval	メッセージが送信先データサーバに正常に渡された後に、Replication Server がそのメッセージを保存しておく時間 (分単位)。詳細については、『Replication Server 管理ガイド 第 2 巻』を参照してください。 デフォルト値は 0 分

database_param	説明と*値*
sqm_cmd_cache_size	Replication Server が SQM コマンドキャッシュに保存できる解析データの最大サイズ (バイト単位)。 32 ビット版 Replication Server の場合： • デフォルト - 1,048,576 • 最小値 - 0 (SQM コマンドキャッシュは無効) • 最大値 - 2,147,483,647 64 ビット版 Replication Server の場合： • デフォルト - 20,971,520 • 最小値 - 0 • 最大値 - 2,251,799,813,685,247 Replication Server では、**cmd_direct_replicate** または **sqm_cache_enable** が off に設定されている場合、**sqm_cmd_cache_size** が無視される。 **configure replication server** を使用してサーバレベルですべてのコネクションのパラメータを設定するか、**alter connection** を使用してデータベースレベルで個々のコネクションを設定する。
sqm_max_cmd_in_block	各 SQM ブロックで、解析データが関連付けられるエントリの最大数を指定する。 デフォルト値は 320 最小値：0 最大値：4096 **sqm_max_cmd_in_block** の値を SQM ブロックのエントリ数に設定する。データのプロファイルによっては、ブロックサイズが固定されており、メッセージサイズは予測できないため、ブロックごとにエントリが異なる場合がある。設定した値が大きすぎると、メモリを無駄にすることになる。値が小さすぎると、レプリケーションのパフォーマンスが低下する。 Replication Server では、**cmd_direct_replicate** または **sqm_cache_enable** が off に設定されている場合、**sqm_max_cmd_in_block** が無視される。

database_param	説明と*値*
sqt_max_prs_size	HVAR および RTL のトランザクション。プロファイリング処理によってアンパックされたコマンドが使用する最大メモリ (バイト)。 32 ビット版 Replication Server の場合： • デフォルト - 2,147,483,647 (2GB) • 最小値 - 0 • 最大値 - 2,147,483,647 64 ビット版 Replication Server の場合： • デフォルト - 2,147,483,647 (2GB) • 最小値 - 0 • 最大値 - 2,251,799,813,685,247 **configure replication server** を使用してサーバレベルですべてのコネクションのパラメータを設定するか、**alter connection** を使用してデータベースレベルで個々のコネクションを設定する。 データベースレベルのデフォルト値は 0。 データベースレベルのデフォルトを保持するか、デフォルトにリセットすると、Replication Server はサーバレベルで設定された値を使用する。 Replication Server 15.7.1 SP100 以降にアップグレードした場合、32 ビットと 64 ビットの Replication Server の両方でデフォルトを 2GB に設定する必要がある。
stage_operations	on に設定すると、Replication Server と SAP IQ InfoPrimer の統合を設定する際に、Replication Server は指定したコネクションのステージングテーブルにオペレーションを書き込む。 『異機種間複写ガイド』の「**stage_operations**」を参照。
sub_sqm_write_request_limit	アウトバウンドキューへの書き込み待ちメッセージ用に、サブスクリプションマテリアライゼーション／マテリアライゼーション解除スレッドが使用可能なメモリ量を指定する。 デフォルト値は 1MB 最小値：16KB 最大値：2GB

database_param	説明と*値*
unicode_format	U&″ フォーマットの Unicode データの送信をサポートする。これにより、Replication Server における UTF-8 文字セットの制限がなくなる。 **unicode_format** を次の値のいずれかに設定する。 • `string` - Unicode 文字を文字列フォーマットに変換する。 たとえば、文字列 "hello" は "hello" として送信される。 この場合、Replication Server では UTF-8 が必須である。 • `ase` - Unicode 文字を `U&′′` フォーマットで送信する。 たとえば、文字列 "hello" は "U&'¥0068¥0065¥006c¥006c¥006f' " として送信される。 2 バイト Unicode 値は、Adaptive Server Enterprise が要求するネットワーク順序で送信される。 この場合、Replication Server では UTF-8 以外の文字セットを使用できる。 デフォルト値は文字列
use_batch_markers	ファンクション文字列 **rs_batch_start** と **rs_batch_end** の処理を制御する。**use_batch_markers** を on に設定すると、**rs_batch_start** ファンクション文字列がコマンドの各バッチの先頭に追加され、**rs_batch_end** ファンクション文字列がコマンドの各バッチの末尾に追加される。 レプリケートデータサーバで、**rs_begin** ファンクション文字列に含まれていないコマンドのバッチの開始時と終了時に、追加の SQL を送信する必要がある場合にのみ、**use_batch_markers** を on に設定する。 デフォルト値は off

- **security_param –** コネクションのネットワークベースのセキュリティに影響を与えるパラメータ。 パラメータのリストと値の説明については、**create route** の「ネットワークのベースセキュリティに影響を与えるパラメータ」テーブルを参照してください。 このパラメータは ASE 以外、IQ 以外のコネクタには適用されません。
- **set security_services to 'default' –** Replication Server のグローバル設定と一致させるために、コネクションのすべてのネットワークベースのセキュリティ機能をリセットします。 このパラメータは ASE 以外、IQ 以外のコネクタには適用されません。
- **new_ds および new_db –** コネクションの対象となる新しいデータサーバとデータベースの名前

  注意： *new_ds* パラメータと *new_db* パラメータの値は、*data_server* パラメータと *database* パラメータに定義したものと同じにすることができます。
- **trace –** DSI レベルでの ExpressConnect トレースを許可します。
- **value –** オプションの新しい値を持つ文字列です。

trace オプションを使用する場合、value の構文は、“module, condition,[on|off]” の形式になります。構文の説明は次のとおりです。

- *module* - モジュールタイプを指定します。有効な値は *econn* です。
- *condition* - trace オプションを *on* に設定するか *off* に設定するかを指定します。
- *on* または *off* - 目的の条件のステータスを指定します。

注意： alter connection コマンドの **trace** パラメータには空の文字列を指定できます。例：

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

接続するか、Replication Server を再起動すると、空の文字列によって ExpressConnect トレース値が無効にされます。

- ***data_server.db*** – データサーバとプライマリデータを格納するデータベースを指定します。
- ***schemamap*** – プライマリスキーマまたは所有者 (*from_schema*) を複写スキーマまたは所有者 (*to_schema*) にマッピングします。

  DML または 異機種間 DDL 複写では、レプリケートデータベースがテーブル所有者を含むテーブル複写定義にサブスクライブしている場合は、そのテーブル複写定義のレプリケート所有者が有効になります。 schemamap を適用できるのは、テーブルにテーブルサブスクリプションがない場合か、テーブル複写定義にレプリケートテーブル所有者が含まれていない場合のみです。
- ***from_schema*** – プライマリデータベースのスキーマ、つまりマップされるオリジンユーザを指定します。 *from_schema* が **NULL** の場合は、プライマリ接続名のすべてのユーザが *to_schema* にマップされます。
- ***to_schema*** – レプリケートデータベースのスキーマ、つまりマップされるターゲットユーザを指定します。 *to_schema* が **NULL** の場合は、プライマリ接続名のすべてのユーザが *from_schema* にマップされます。
- ***with decluster*** – スキーマに対してクラスタテーブルのクラスタ化解除を有効にします。
- ***without decluster*** – スキーマに対してクラスタテーブルのクラスタ化解除を無効にします。
- **NULL** – *from_schema* と *to_schema* の両方に NULL を指定すると、*data_server.db* のマッピング関係が削除されます。

例

- **例 1** – TOKYO_DS データサーバにある *pubs2* データベースのファンクション文字列クラスを *sql_derived_class* に変更します。

```
suspend connection to TOKYO_DS.pubs2

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

resume connection to TOKYO_DS.pubs2
```

- **例 2 –** LTI または RepAgent エグゼキュータスレッドが、他のスレッドに CPU を解放しなければならなくなるまでに処理できる LTL コマンドの数を変更します。

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

- **例 3 –** replicateDS.replicateDB への接続を変更し、primaryDS.primaryDB のユーザ A をユーザ B にマップします。

```
alter connection to replicateDS.replicateDB
set schemamap from primaryDS.primaryDB.A to B
```

- **例 4 –** replicateDS.replicateDB への接続を変更し、primaryDS.primaryDB のすべてのユーザをユーザ B にマップします。

```
alter connection to replicateDS.replicateDB
set schemamap from primaryDS.PrimaryDB.NULL to B
```

使用法

- **suspend connection** は、コネクションを変更する前にコネクションのアクティビティをサスペンドするときに使用します。
- **alter connection** は、コネクションが作成された Replication Server で実行します。
- **log transfer off** を使用してプライマリデータベースからのデータ転送を停止させる場合は、その前に、そのデータベースのデータに複写定義が定義されていないことを確認してください。
- Replication Server のルートを変更する場合は、**alter route** を使用します。
- SAP 以外のデータサーバのクラスレベル変換をアクティブにするには、**set function string class [to]** *function_class* を使用します。
- デフォルトコネクションまたは代替コネクションのコネクションパラメータは、**alter connection** パラメータを使用して設定できます。

  代替コネクションに対して設定した値は、デフォルトコネクションから継承された値またはデフォルト値よりも優先されます。
- **alter connection** は、コネクションが作成された Replication Server で実行します。

データベースコネクションパラメータ

- **alter connection** は、DSI またはデータベースコネクションの設定パラメータを変更するときに使用します。DSI の設定値を変更する場合は、DSI へのコネク

ションをサスペンドし、値を変更してから、DSI へのコネクションをレジュームします。これにより、新しい値が有効になります。

- Replication Server の設定パラメータは、*rs_config* システムテーブルに格納されています。パラメータの中には、テーブル内のローを更新することによって修正できるものもあります。詳細については、『Replication Server 管理ガイド 第 1 巻』を参照してください。
- 並列 DSI スレッドの設定の詳細については、『Replication Server 管理ガイド 第 2 巻』を参照してください。
- **assign action** を使用すると、データサーバの特定のエラーが原因で失敗したトランザクションをリトライできるようになります。
- ファンクション文字列クラスを変更する前に、新しいクラス用のクラスと必要なすべてのファンクション文字列が存在することを確認してください。
- エラークラスを変更する前に、新しいクラスが存在することを確認してください。
- コマンドの末尾を認識するため、コマンドセパレータが必要なデータサーバ用に文字を変更してください。
  別のセパレータ文字を指定した後に、その文字を改行文字に戻す場合は、**alter connection** コマンドを次のように入力します。

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

  ここで、一重引用符の間の Return では、他の文字は入力せずに [Return] キーを押してください。

dsi_bulk_copy パラメータ

dsi_bulk_copy を on にすると、SQT によって、トランザクションに含まれる同じテーブルでの連続する **insert** 文の数がカウントされます。 この数が **dsi_bulk_threshold** に達すると、DSI によって、以下が実行されます。

1. Bulk-copies the data to Adaptive Server until DSI が、**insert** でないコマンドまたは異なるレプリケートテーブルに属するコマンドに到達するまで、データを Adaptive Server にバルクコピーします。
2. トランザクションの残りのコマンドの実行を続行します。

Adaptive Server が、バルクオペレーションが成功した場合はその終了時点、またはオペレーションが失敗した時点でバルクコピーインの結果を送信します。

注意： DSI でのバルクコピーインの実装により、複数文のトランザクションがサポートされるため、バルクコピーに含まれないコマンドがトランザクションに含まれている場合でも、DSI でバルクコピーインを実行できます。

dsi_partitioning_rule パラメータ

一度に複数のパーティショニングルールを指定できます。値は空白ではなくカンマで区切ります。例：

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

dataserver and database name パラメータ

dataserver and database name パラメータを使用すると、コネクションを 1 つのコネクタから別のコネクタを使用するように切り替えることができます。 たとえば、ASE/CT-Lib コネクタを使用して Oracle に複写するときに、Oracle/OCI コネクタを使用するようにコネクションを切り替える場合は、新しいデータサーバとデータベース名の使用が求められることがあります。 これは、SAP インタフェースファイルで DirectConnect/Oracle に指定された名前が Oracle TNS Names ファイル内の Oracle データサーバ名と異なる場合があるためです。 次のように変更します。

1. コネクションをサスペンドします。
2. コネクション設定 **dsi_dataserver_make** を *ora* に、**dsi_connector_type** を *oci* に変更します。
3. コネクション設定 **dataserver and database name** を **new_ds** と **new_db** に変更します。
   構文の説明は次のとおりです。
   - *new_ds* - Oracle `tnsnames.ora` ファイル内のデータサーバ名
   - *new_ds* - データベース名

   注意： *new_ds* パラメータと *new_db* パラメータの値は、*data_server* パラメータと *database* パラメータに定義したものと同じにすることができます。
4. コネクションをレジュームします。

dump_load パラメータ

dump_load を “on” に設定する前に、**rs_dumpdb** ファンクションと **rs_dumptran** ファンクションのファンクション文字列を作成してください。Replication Server は、システムによって提供されるクラスやそのクラスから継承された派生クラスでは、これら 2 つのファンクションのファンクション文字列は生成しません。

save_interval 設定パラメータ

save_interval を設定すると、データベースがバックアップからリストアされた後、データベースを再同期するために使用される DSI キューにトランザクションが保存されます。セーブインターバルの設定は、レプリケートデータの保持、または複写ファンクションの受信を行うデータベースのウォームスタンバイを設定する場合にも使用できます。**sysadmin restore_dsi_saved_segments** を使用すると、バックログトランザクションをリストアできます。

ネットワークベースセキュリティのパラメータ

- これらのパラメータは ASE 以外、IQ 以外のコネクタには適用されません。
- コネクションの両端では、同じセキュリティメカニズムとセキュリティ機能を備えた互換性のある SCL (Security Control Layer) ドライバを使用してください。また、データサーバは、**set proxy** または同等のコマンドをサポートしている必要があります。
  各サーバについて、セキュリティ機能の選択と設定を行うのは複写システム管理者の仕事です。Replication Server は、リモートサーバとのコネクションを確立しようとする前に、そのサーバのセキュリティ機能の問い合わせは行いません。コネクションの両端のセキュリティ機能に互換性がないと、コネクションは失敗します。
- **alter connection** を使用すると、Replication Server からターゲットデータサーバへの送信コネクションのネットワークベースセキュリティ設定を修正できます。この修正内容によって、**configure replication server** で設定されたデフォルトのセキュリティパラメータが上書きされます。
- **unified_login** を “required” に設定すると、“sa” パーミッションを持つ複写システム管理者だけがクレデンシャルなしで Replication Server にログインできます。セキュリティメカニズムに問題が発生した場合でも、複写システム管理者はパスワードを使用して Replication Server にログインし、**unified_login** を無効にできます。
- Replication Server には、複数のセキュリティメカニズムを装備できます。サポートされるメカニズムは、それぞれ `libtcl.cfg` ファイル内の SECURITY セクションにリストされています。
- メッセージの暗号化は、深刻なパフォーマンスの低下をともなう、負荷の高い処理です。通常は、特定のコネクションに対してだけ、**msg_confidentiality** を “required” に設定してください。代わりに、**msg_integrity** などの負荷の低いセキュリティ機能を選択します。

alter connection を使用したメンテナンスパスワードの変更

- DSI コネクションのメンテナンスユーザのパスワードを変更するには、**alter connection** コマンドを使用します。

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

- Replication Server で ERSSD を使用しており、*data_server.database* が ERSSD 名と一致する場合、**alter connection** と **set password** を使用すると、*rs_maintusers* テーブルを更新し、ERSSD で **sp_password** を発行して、設定ファイルの行 *RSSD_maint_pw_enc* を更新できます。

パーミッション

alter connection には “sa” パーミッションが必要です。

参照：

- admin show_connections (77 ページ)
- admin who (107 ページ)
- create alternate connection (269 ページ)
- create connection (287 ページ)
- configure replication server (238 ページ)
- create error class (308 ページ)
- create function string class (334 ページ)
- drop connection (407 ページ)
- resume connection (436 ページ)
- set proxy (448 ページ)
- suspend connection (452 ページ)

alter connector

データベースコネクタの属性を変更します。

構文

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

パラメータ

- **dataserver_make –** データベースサーバを示します。
- **connector_type –** コネクタの実装に使用するコネクタテクノロジを示します。
- **option –** コネクタのさまざまなトレースオプションに対する選択肢を提供します。

  サポートされているオプションは次のとおりです。

  - **trace**
  - **trace_logpath**
- **value –** オプションの新しい値を持つ文字列です。

  trace オプションを使用する場合、*value* の構文は、“*module, condition,[on|off]*”の形式になります。構文の説明は次のとおりです。

  - *module* - モジュールタイプを指定します。有効な値は *econn* です。
  - *condition* - 設定するトレース条件を指定します。
  - *on* または *off* - 目的の条件のステータスを指定します。

例

- **例 1 –** general_1 トレース条件を有効にして ASE/CT-Lib コネクタを使用するようにすべての DSI インスタンスを設定します。

```
alter connector "ase"."ctlib"
set trace to "econn,general_1,on"
```

- **例 2 –** この例では、*option* パラメータが **trace_logpath** に設定されており、ASE/CT-Lib コネクタが生成するすべてのトレースメッセージは、Replication Server ログファイルに加えてコネクタ固有のトレースファイルにも書き込まれます。

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

  一般的に、ログファイル名は次の部分で構成されています。

  - `ec`
  - *dataserver_make*
  - *connector_type*
  - `.log`

  dataserver_make と *connector_type* は変数です。 値は使用しているデータベースの種類と、関連付けられているコネクタテクノロジに応じて異なります。 たとえば、ASE/CT-Lib に対して作成されるコネクタ固有のログファイルは `ecasectlib.log` です。

- **例 3 –** トレースメッセージがコネクタ固有のトレースファイルに書き込まれないようにするには、**trace_logpath** 設定を次のように変更します。

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

使用法

- **alter connecttor** は、コネクションが作成された Replication Server で実行します。
- 指定したコネクタを使用しているすべてのコネクションに対してトレースを有効にするには、**alter connector** を実行します。

参照：

- alter connection (134 ページ)

alter database replication definition

既存のデータベース複写定義を変更します。

構文

```
alter database replication definition db_repdef
with primary at data_server.database
{[{not replicate DDL} |
{replicate DDL [{with | without} {auto_update_table_list |
auto_extend_table_list}]}] |
[not] replicate setname setcont |
[not] replicate {{SQLDML | DML_options} [in table_list]} |
[alter owner from current_table_owner to new_table_owner [for
table_name]] |
[{add | remove} tables {setcont}]}
[with dsi_suspended]
[user username password pass]
setcont ::= [[in] ([owner1.]name1[, [owner2.]name2 [, ... ]])] | [in
files ('file_path')]
setname ::= {tables | functions | transactions | system procedures}
```

注意： *setname* の "functions" は、ユーザ定義ストアドプロシージャまたはユーザ定義ファンクションを指します。

パラメータ

- **db_repdef –** データベース複写定義の名前です。
- **with primary at *data_server.database* –** データサーバとプライマリデータを格納するデータベースを指定します。
- **not replicate DDL –** サブスクライブするデータベースに DLL を送信しないよう Replication Server に指示します。
- **replicate DDL [{with | without} {auto_update_table_list | auto_extend_table_list}] –** Replication Server に対し、サブスクライブするデータベースに DDL を送信するよう指示します。また、テーブルリストを更新または拡張するかどうかを指示します。 オプションなしで **replicate DDL** を指定すると、DDL はレプリケートデータベースに送信されますが、テーブルは複写パスに追加されません。
- **replicate DDL with auto_update_table_list –** DDL コマンドをレプリケートデータベースに送信します。 テーブルで **pdb_automark_tables** が **true** の場合、DDL コマンドの **drop table** または **rename table** が検出されると、そのテーブルはテーブルリストで自動的に更新されます。
- **replicate DDL without auto_update_table_list –** DDL コマンドをレプリケートデータベースに送信しますが、DDL コマンドの **drop table** または **rename table** が検出されても、テーブルリスト内のテーブルを更新しません。

- **replicate DDL with auto_extend_table_list –** DDL コマンドをレプリケートデータベースに送信します。 テーブルで **pdb_automark_tables** が **true** の場合、DDL コマンドの **create table** が検出されると、そのテーブルは自動的にテーブルリストに追加されます。
- **replicate DDL without auto_extend_table_list –** DDL コマンドをレプリケートデータベースに送信しますが、DDL コマンドの **create table** が検出されても、テーブルをテーブルリストに追加しません。

注意： replicate DDL コマンドに **auto_extend_table_list** オプションを指定する場合は、システムプロシージャまたは SQLDML オペレーションを同時に複写しないようにしてください。

- **[not] replicate *setnamesetcont* –** *setname* カテゴリのオブジェクトをレプリケートデータベースに送信するかどうかを指定します。 *setname* カテゴリには、テーブル、ファンクション、トランザクション、システムプロシージャごとに 1 つの句しか指定できません。

  システムプロシージャ *setname* を省略した場合、または **not** オプションを指定した場合、システムプロシージャは複写されません。

  テーブル、ファンクション、またはトランザクションの *setname* を省略した場合、および setname を指定し、**not** オプションを指定した場合は、*setname* カテゴリのすべてのオブジェクトが複写されます。

  setname によって指定されているフィルタカテゴリが、現在のフィルタカテゴリの代わりに使用されます。または、これが新しいカテゴリである場合は、そのフィルタカテゴリがデータベース複写フィルタに追加されます。
- **[not] replicate {SQLDML | DML_options} [in table_list] –** SQL 文を、*table_list* に定義されているテーブルに複写するかどうかを Replication Server に伝えます。
- **DML_options –** 次の DML オペレーションの任意の組み合わせです。
  - U - **update**
  - D - **delete**
  - I - **insert select**
  - S - **select into**

  データベースの複写モードを **UDIS** の任意の組み合わせに設定すると、RepAgent は、個々のログレコードと Replication Server が SQL 文を作成するために必要な情報の両方を送信します。
- **alter owner from *current_table_owner* to *new _table_owner* [for *table_name*] –** テーブルの所有者を変更する場合は、現在の所有者と新しい所有者を指定します。

所有権を譲渡する場合は、**for *tablename*** オプションを含めます。 データベース複写定義内の *setname* カテゴリのすべてのテーブルの所有者を変更する場合は、**for *tablename*** を省略します。

- **add tables in {(*table_list*) | files '*file_path*'} –** テーブルをレプリケーションパスに追加します。 テーブルのリストを指定するか、またはファイルでテーブル名を指定できます。 テーブルを複写パスに追加すると、そのテーブルのデータの複写が開始します。

  注意： 一度に 1 つのファイルのみを指定できます。*file_path* には絶対パスを指定する必要があります。
- **remove tables in {(*table_list*) | files '*file_path*'} –** テーブルを複写パスから削除します。テーブルのリストを指定するか、ファイルにテーブル名を指定できます。
- **with dsi_suspended –** レプリケート DSI をサスペンドするよう、レプリケート Replication Server に指示します。データベースの再同期が必要なことを知らせるために使用できます。
- **owner –** テーブルの所有者またはトランザクションを実行するユーザです。 Replication Server は、ファンクションまたはシステムプロシージャの所有者情報は処理しません。

  owner は、一重引用符で囲まれた 1 つのスペース、またはアスタリスクで置き換えることができます。

  - スペース (' ') - 所有者がいないことを示します。
  - アスタリスク (*) - すべての所有者を表します。 たとえば、**.publisher* は、所有者に関係なく、*publisher* という名前のすべてのテーブルを表します。
- **name –** テーブル、ファンクション、トランザクション、またはシステムプロシージャの名前です。

  name は、一重引用符で囲まれた 1 つのスペース、またはアスタリスクで置き換えることができます。

  - スペース (' ') - 名前がないことを示します。 たとえば、*maintuser.' '* はメンテナンスユーザのすべての名前のないトランザクションを表します。
  - アスタリスク (*) - すべての名前を表します。 たとえば、*robert.** は、*robert* が所有するすべてのテーブル (またはトランザクション) を表します。
- **[in files ('*file_path*')] –** 対象テーブルリストまたは除外テーブルリストが記述されているファイル。 一度に 1 つのファイルのみを指定できます。*file_path* には絶対パスを指定する必要があります。

  注意： プライマリ Replication Server を起動するユーザにこのファイルの読み取りパーミッションが必要です。

  テーブル名の書式は、ファイルでもテーブルリストでも同じです。 テーブルリストでは次の書式でテーブル名を指定できます。

- *ownername.tablename*
- *tablename* (テーブル名は dbo.tablename として格納されます)
- *.*tablename*
- *ownername*.*
- '*x*y'.'a*b' (文字列にワイルドカードが埋め込まれています)

注意： create database replication definition を発行すると、部分ワイルドカードはテーブルリストや例外リストを含むすべてのリストで展開されます。 単純ワイルドカードは、例外リストで展開されてからシステムテーブルに格納されます。 ワイルドカードの展開が必要な場合は、**user** と **password** の値を指定する必要があります。

対象テーブルリストまたは除外テーブルリストを使用するときは、次の次のガイドラインに従ってください。

- ファイル内ではテーブル名のデリミタとして改行文字を使用します。
- # で始まる行はコメントとして無視されます。
- 所有者名またはテーブル名の前の空白はトランケートされます。
- 所有者名の最大長は 30 文字です。

- **[user *username* password *pass*] –** プライマリ Adaptive Server database または Replication Agent への接続用のユーザ ID とパスワード。プライマリテーブルから選択します。

  テーブル名にワイルドカードを使用する場合は、**username** と **password** の値を指定する必要があります。

  注意： user と **password** の値は一度だけ使用され、RSSD には保存されません。

例

- **例 1 –** データベース複写定義 *rep_1C* を変更して、table2 をフィルタします。レプリケート DSI はサスペンドされます。

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

- **例 2 – update** 文と **delete** 文を *tb1* テーブルと *tb2* テーブルに適用します。

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

- **例 3**

  Adaptive Server **alter... modify owner** コマンドでテーブルの所有者を mario から angela に変更後、プライマリは Replication Server は以下を実行します。

```
alter database replication definition authors_dbrepdef
with primary at NY_DS.pdb1
alter owner from mario to angela for author_name
```

使用法

- データベース複写定義を変更すると、プライマリデータベースとレプリケートデータベースが同期しなくなる可能性があります。データベースの再同期の手順については、『Replication Server 管理ガイド 第 1 巻』を参照してください。
- SQL 文の複写:
  - 複写定義でフィルタを指定しない場合、デフォルトは **not replicate** 句です。SQLDML フィルタを変更するには、**alter database replication definition** を適用します。 **replicate** 句では、1 つまたは複数の SQLDML フィルタを指定できます。
  - SQL 文の複写の詳細については、「**データベース複写定義の作成**」を参照してください。

参照：

- create database replication definition (300 ページ)
- drop database replication definition (408 ページ)

alter encryption key

暗号化キーの再生成。

構文

```
alter encryption key key_name regenerate
```

パラメータ

- ***key_name*** – 生成する暗号化キーの名前です。

  有効な値

  `rs_password_key:password encryption key`

例

- **例 1** – パスワード暗号化キーを再生成します。

  ```
  alter encryption key rs_password_key regenerate
  ```

使用法

- Replication Server は、パスワードの暗号化に `rs_encryptionkeys` RSSD システムテーブルの `rs_password_key` ローと、Replication Server 設定ファイルの **RS_random** 属性を使用します。

  `rs_password_key` ローと **RS_random** 属性のランダム値を再生成するには、**alter encryption key** コマンドを使用します。RSSD のパスワードは、新しい暗号化キーを使用して自動的に再暗号化されます。

パーミッション

alter encryption key には sa パーミッションが必要です。

alter error class

エラーアクションを別のエラークラスからコピーすることによって、既存のエラークラスを変更します。

構文

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

パラメータ

- **replication server** – エラークラスが Replication Server エラークラスであり、データサーバのエラークラスではないことを示します。
- **error_class** – 修正するエラークラスです。
- **set template to template_error_class** – この句を使用して、別のエラークラスに基づいてエラークラスを更新します。 **alter error class** により、テンプレートのエラークラスのエラーアクションが既存のエラークラスにコピーされます。

例

- **例 1 – my_error_class** を変更します。変更は、**rs_sqlserver_error_class** に基づいて行います。

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

- **例 2 – my_rs_err_class** Replication Server エラークラスを変更します。変更は、デフォルトの Replication Server エラークラスである **rs_repserver_error_class** に基づいて行います。

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

使用法

- **alter error class** コマンドと、テンプレートとしての他のエラークラスを使用し、エラークラスを変更します。**alter error class** は、テンプレートのエラークラスから変更対象のエラークラスにエラーアクションをコピーし、同じエラーコードを持つエラーアクションを上書きします。
- *rs_sqlserver_error_class* は Adaptive Server データベースに用意されているデフォルトのエラークラスであり、*rs_repserver_error_class* は Replication Server に用意されているデフォルトのエラークラスです。最初は、この 2 つのエラークラスにはプライマリサイトがありません。デフォルトのエラーアクションを変更するには、プライマリサイトでこれらのエラークラスを作成する必要があります。
- **create connection** および **alter connection** コマンドを使用して、Adaptive Server 以外のエラークラスを Adaptive Server 以外のレプリケートデータベースの特定のコネクションに割り当てることができます。
- Replication Server は、ASE 以外のレプリケートサーバへのコネクションを確立するときに、コネクションで ASE 以外のレプリケートサーバからネイティブエラーコードが返されるオプションが有効になっているかどうかを検証します。オプションが有効になっていない場合、Replication Server は、コネクションは機能しているが、エラーアクションのマッピングが正確でない可能性があることを示す警告メッセージをログに記録します。

  Enterprise Connect Data Access Option for ODBC でレプリケートサーバ用のオプションを設定するには、Replication Server Options のマニュアルで「**ReturnNativeError**」を参照してください。
- Adaptive Server 以外のエラークラスのリストについては、「表 31 : エラークラスとファンクションクラス」を参照してください。Adaptive Server 以外の複写のエラークラスの詳細については、『Replication Server 管理ガイド 第 2 巻』を参照してください。

参照：

- assign action (226 ページ)
- create error class (308 ページ)
- drop error class (409 ページ)

alter function

ユーザ定義ファンクションにパラメータを追加します。

構文

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

パラメータ

- **table_rep_def –** ユーザ定義ファンクションが実行される複写定義の名前です。
- **function_name –** 変更するユーザ定義ファンクションの名前です。
- **@param_name –** ユーザ定義ファンクションのパラメータリストに追加するパラメータの名前です。パラメータ名は識別子の規則に従い、前に @ 記号を付けます。
- **datatype –** パラメータのデータ型です。データ型とその構文のリストについては、「データ型」を参照してください。このパラメータに、*text*、*unitext*、*raw object*、または *image* を指定することはできません。

例

- **例 1 –**

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

publishers_rep 複写定義の *upd_publishers* ファンクションに、*state* という整数パラメータを追加します。

使用法

- **alter function** を実行する前に、複写システムをクワイスしてください。システムをクワイスするには、Replication Server Manager を使用するか、『Replication Server トラブルシューティングガイド』で説明されている手順に従います。
- 1 つのユーザ定義ファンクションには、最大 255 のパラメータを指定できます。
- 更新中にファンクションを変更すると、予期しない結果が発生することがあります。ファンクションを変更する前に、影響を受けるデータはクワイスしておく必要があります。
- ユーザ定義ファンクションを変更した後、新しいパラメータを使用するファンクション文字列も変更してください。

- 複写定義に対するユーザ定義ファンクションを変更すると、プライマリテーブル内の複写定義すべてに対するユーザ定義ファンクションが変更されます。
- 複写ファンクションには **alter function** を使用しないでください。代わりに、**alter function rep def** を使用します。**alter function** は、「RSSD ストアドプロシージャ」で説明している非同期ストアドプロシージャにのみ使用します。

パーミッション

alter function には、“create object” パーミッションが必要です。

参照：

- admin quiesce_check (66 ページ)
- alter function string (188 ページ)
- create function (310 ページ)
- create function string (318 ページ)
- drop function (410 ページ)
- drop function string (412 ページ)

alter function replication definition

create function replication definition コマンドによって作成された既存のファンクション複写定義を変更します。

注意： **create function replication definition** と **alter function replication definition** は、サポートされなくなる予定です。 これらの代わりに、次のコマンドを使用することをお奨めします。

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

構文

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

パラメータ

- **function_rep_def –** 変更するファンクション複写定義の名前です。
- **deliver as –** 複写ファンクションを配信するデータベースで実行するストアドプロシージャの名前を指定します。*proc_name* は最大 200 文字の文字列です。このオプションを指定しない場合、ファンクションはファンクション複写定義と同じ名前のストアドプロシージャとして配信されます。
- **add –** ファンクション複写定義に追加するパラメータとそのデータ型を指定します。
- **@param_name –** 複写パラメータまたはサーチャブルパラメータのリストに追加するパラメータの名前です。各パラメータ名は @ 記号で始まる必要があります。
- **datatype –** パラメータリストに追加するパラメータのデータ型です。サポートされているデータ型とその構文のリストについては、「データ型」を参照してください。Adaptive Server のストアドプロシージャとファンクション複写定義には、*text*、*unitext*、*image* の各データ型のパラメータを含めることはできません。
- **add searchable parameters – define subscription** コマンドまたは **define subscription** コマンドの **where** 句で使用できる追加パラメータを指定します。
- **send standby –** ウォームスタンバイアプリケーションで、スタンバイデータベースにファンクションのすべてのパラメータを送信するか (**send standby all parameters**)、複写定義で指定されたパラメータだけを送信するか (**send standby replication definition parameters**) を指定します。デフォルトは、**send standby all parameters** です。

例

- **例 1 –** 3 つのパラメータを *titles_frep* ファンクション複写定義に追加します。これらは、*varchar* パラメータ (*@notes*)、*datetime* パラメータ (*@pubdate*)、*bit* パラメータ (*@contract*) です。

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

- **例 2 –** *titles_frep* ファンクション複写定義のサーチャブルパラメータのリストに、*@type* パラメータと *@pubdate* パラメータを追加します。

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

- **例 3 –** 送信先データベース (通常は、要求ファンクションの配信に使用されるプライマリデータベース) で、*newtitles* ストアドプロシージャとして配信されるように、*titles_frep* ファンクション複写定義を変更します。

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

使用法

- **alter function replication definition** は、複写パラメータやサーチャブルパラメータを追加したり、すべてのパラメータをウォームスタンバイに送信するかどうかを指定したり、送信先データベースで実行するストアドプロシージャに別の名前を指定したりすることによって、ファンクション複写定義を変更します。
- 変更するファンクション複写定義に指定する名前、パラメータ、データ型は、複写するストアドプロシージャと一致していなければなりません。複写したいパラメータだけを指定できます。
- **alter function replication definition** は、プライマリデータベース (ファンクション複写定義を作成したデータベース) を管理する Replication Server で実行してください。
- 同じパラメータ名を句の中で 2 回以上指定することはできません。
- パラメータを追加する場合は、ファンクション複写定義の分配に合わせて **alter function replication definition** を実行します。「ファンクション複写定義の変更」で説明されている手順に従って、エラーを避けてください。
- オプションの **deliver as** 句を使用すると、複写ファンクションを配信する送信先データベースで実行するストアドプロシージャの名前を指定できます。通常、このオプションは要求ファンクションの配信に使用します。詳細については、「**create connection**」を参照してください。

  詳細については、『Replication Server 管理ガイド 第 1 巻』の「**alter function replication definition**」の説明を参照してください。

ファンクション複写定義の変更:

1. 『Replication Server トラブルシューティングガイド』に記載された手順に従って、複写システムをクワイスします。
   最初にプライマリへの更新をクワイスして、すべてのプライマリ更新が複写システムで処理されたことを確認するのが理想的です。これができない場合、プライマリログにある古い更新には新しいパラメータの値は含まれず、代わりに null が使用されます。これは、次の手順 4 でファンクション文字列を変更する場合に考慮する必要があります。
2. プライマリサイトとレプリケートサイトで、ストアドプロシージャを変更します。
3. ファンクション複写定義を変更します。 レプリケートサイトに変更したファンクション複写定義が反映されるのを待ちます。
4. 必要に応じて、ファンクション複写定義に属するファンクション文字列を変更します。 レプリケートサイトに変更したファンクション文字列が反映されるのを待ちます。

5. 必要に応じて、レプリケートサイトのファンクション複写定義のサブスクリプションを変更してください。サブスクリプションを変更するには、**drop subscription** を使って削除した後に (マテリアライゼーションオプションを指定せずに) **create subscription** を使って再作成します。
   複写定義の変更は、現在のサブスクリプションには影響しません。新しいパラメータがファンクション複写定義に追加される場合、既存のすべてのサブスクリプションに対する新しい更新によって複写されます。
6. プライマリデータベースでデータへの更新をレジュームします。

パーミッション

alter function replication definition には、“create object” パーミッションが必要です。

参照：

- alter function string (188 ページ)
- create function replication definition (312 ページ)
- drop function replication definition (411 ページ)

alter function string

既存のファンクション文字列を置き換えます。

構文

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

例

- **例 1 – rs_update** カスタムファンクション文字列を変更します。変更は、NY_DS データサーバの `authors` テーブル (`rdb1` ターゲットデータベース内) が対象になります。

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

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

使用法

- **alter function string** の機能は **create function string** と同じです。ただし、最初に **drop function string** を実行する点が異なります。ファンクション文字列は 1 つのトランザクション内で削除されて再作成されます。これにより、ファンクション文字列が失われることによって発生するエラーを防ぎます。
- ファンクションのファンクション文字列を、ファンクション文字列クラスのプライマリサイトのクラススコープを使用して変更します。ファンクション文字列クラスのプライマリサイトの詳細については、「**create function string class**」を参照してください。
- 複写定義を作成したサイトで、複写定義スコープを持つファンクション (ユーザ定義ファンクションなど) のファンクション文字列を変更します。複写定義には、それぞれ固有のファンクション文字列セットがあります。
- スタンバイデータベースまたはレプリケートデータベースであるターゲットデータベースを制御する Replication Server にあるターゲットスコープファンクション文字列に対し、**alter function string** を実行します。
- **rs_select**、**rs_select_with_lock**、**rs_datarow_for_writetext**、**rs_get_textptr**、**rs_textptr_init**、**rs_writetext** の各ファンクション文字列では、Replication Server は変更する文字列を判断するために *function_string* 名を使用します。ファンクション文字列の作成時に *function_string* 名が指定されていた場合は、変更するファンクション文字列が見つかるように、**alter functionstring** を使用してファンクション文字列を指定します。
- 「**create function string**」では、**alter function sting** に使用できるキーワードとオプションのさまざまなパラメータについて説明しています。
- ファンクションのデフォルトファンクション文字列をリストアするには、**output** 句を省略してください。

パーミッション

alter function string には、“create object” パーミッションが必要です。

参照：

- alter connection (134 ページ)
- create connection (287 ページ)

- create function (310 ページ)
- create function string (318 ページ)
- create function string class (334 ページ)
- define subscription (395 ページ)
- drop function string (412 ページ)

alter function string class

基本クラスと派生クラスのどちらにするかを指定して、ファンクション文字列クラスを変更します。

構文

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

パラメータ

- **function_class –** 変更する既存のファンクション文字列クラスの名前です。
- **set parent to –** 既存のクラスを、変更するクラスの親として指定します。または、**null** キーワードを使用して、そのクラスを基本クラスに指定します。
- **parent_class –** 新しい派生クラスの親クラスとして指定する既存のファンクション文字列クラスの名前です。*rs_sqlserver_function_class* は親クラスとして使用できない場合があります。
- **null –** そのクラスを基本クラスに指定します。

例

- **例 1 –** *sqlserver2_function_class* が、親クラス *rs_default_function_class* からファンクション文字列クラスを継承して派生クラスになることを指定します。

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

- **例 2 –** *rpc_xact* という派生ファンクション文字列クラスを基本クラスにするように指定します。

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

使用法

- **alter function string class** は、派生ファンクション文字列クラスから基本クラスへの変更、派生クラスの親クラスの変更、基本クラスから派生クラスへの変更を行うときに使用します。
- 派生クラスのプライマリサイトは、その親クラスと同じです。親クラスのプライマリサイトで派生クラスを変更します。ただし、親クラスがシステム提供クラス *rs_default_function_class* または *rs_db2_function_class* である場合、派生クラスのプライマリサイトは、その派生クラスを作成した Replication Server になります。
- 「**create function string**」では、**alter function string class** について詳しく説明しています。
- ファンクション文字列クラス、ファンクション文字列、ファンクションの詳細については、『Replication Server 管理ガイド 第 2 巻』を参照してください。
- Replication Server は、複写システムを介して、変更されたファンクション文字列クラスを条件を満たすサイトに分配します。複写システムで通常の遅延時間が発生するため、変更内容がレプリケートサイトにすぐに反映されるわけではありません。

パーミッション

alter function string class には、“sa” パーミッションが必要です。

参照：

- alter connection (134 ページ)
- create connection (287 ページ)
- create function (310 ページ)
- create function string (318 ページ)
- create function string class (334 ページ)
- drop function string class (414 ページ)

alter logical connection

論理コネクションのディストリビュータスレッドを有効または無効にします。また、論理コネクションの属性を変更し、スタンバイデータベースへの **truncate table** の複写を有効または無効にします。

構文

```
alter logical connection
to logical_ds.logical_db {
```

```
set distribution {on | off} |
set logical_database_param to 'value'}
```

パラメータ

- **logical_ds –** 論理コネクションのデータサーバの名前です。
- **logical_db –** 論理コネクションのデータベースの名前です。
- **distribution on –** 論理コネクションのディストリビュータスレッドを起動します。
- **distribution off –** 論理コネクションのディストリビュータスレッドを停止します。
- **logical_database_param –** 論理コネクションに影響を与える設定パラメータの名前です。「表 19 : 論理コネクションに影響を与える設定パラメータ」では、**alter logical connection** で設定できるパラメータについて説明します。
- **value –** パラメータに対応する設定パラメータの設定です。*value* は文字列です。

表 19 : 論理コネクションに影響を与える設定パラメータ

logical_database_param	値
dist_stop_unsupported_cmd	**dist_stop_unsupported_cmd** を使用して、ダウンストリーム Replication Server がサポートしていないコマンドが検出されたときに、自動的にサスペンドするか、引き続き実行するように DIST を設定する。**dist_stop_unsupported_cmd** が on の場合、ダウンストリーム Replication Server がコマンドをサポートしていないときに DIST が自動的にサスペンドする。off の場合、DIST はサポートされていないコマンドを無視する。 **dist_stop_unsupported_cmd** パラメータの設定に関係なく、Replication Server はバージョンの低い Replication Server に送信できない上位バージョンのコマンドの最初のインスタンスを検出したときに、エラーメッセージをログに必ず記録する。 デフォルト値は off
materialization_save_interval	マテリアライゼーションキューのセーブインターバル。このパラメータは、ウォームスタンバイアプリケーション内のスタンバイデータベースに対してのみ使用する。 デフォルト値はスタンバイデータベースでは “strict”

logical_database_param	値
replicate_minimal_columns	Replication Server がすべてのトランザクションのすべての複写定義カラムを送信するか、スタンバイデータベースで更新オペレーションまたは削除オペレーションを実行する必要があるカラムだけを送信するかを指定する。値は “on” または “off”。 Replication Server は、複写定義のパラメータに **send standby** オプションが含まれていないか、複写定義が存在しない場合にのみ、スタンバイ時にこの値を使用する。 それ以外の場合は、複写定義内の "replicate minimal columns" パラメータまたは "replicate all columns" パラメータの値を使用する。 デフォルト値は on **dsi_compile_enable** を ‘on’ に設定すると、**replicate_minimal_columns** に設定した値は無視される。
save_interval	メッセージが送信先データサーバに正常に渡された後に、Replication Server がそのメッセージを保存しておく時間 (分単位)。詳細については、『Replication Server 管理ガイド 第 2 巻』を参照してください。 デフォルト値は 0 分
send_standby_repdef_cols	論理コネクション用にスタンバイデータベースに送信されるカラムを指定する。これは、Replication Server に対して、スタンバイデータベースに送信するテーブルカラムを指示する複写定義内の "send standby" オプションよりも優先される。以下の値のいずれかです。 • **on** - 照合する複写定義で指定されているテーブルカラムだけを送信する。複写定義内の "send standby" オプションは無視される。 • **off** - すべてのテーブルカラムをスタンバイデータベースに送信する。複写定義内の "send standby" オプションは無視される。 • **check_repdef** - "send standby" オプションの設定に基づいて、すべてのテーブルカラムをスタンバイデータベースに送信する。 デフォルト値は **check_repdef**
send_truncate_table	スタンバイデータベースへの **truncate table** の複写を有効にするか無効にするかを指定する。以下の値のいずれかです。 • **on** - スタンバイデータベースへの **truncate table** の複写を有効にする。デフォルト値。 • **off** - スタンバイデータベースへの **truncate table** の複写を無効にする。

logical_database_param	値
ws_sqldml_replication	SQL 文をウォームスタンバイのデータサーバに複写するかどうかを指定する。以下の値のいずれかです。 • **on** - SQL 文を複写する。複写されるデフォルトの文は **update**、**delete**、**insert select**、**select into**。 • **off** - すべての SQL 文を無視する。 **注意：** **ws_sqldml_replication** の優先度は SQL 複写のテーブル複写定義よりも低くなります。テーブル複写定義にテーブルの **send standby** 句が含まれている場合、この句によって、**select into** 以外の DML 文を複写するかどうかが決定されます。これは、**ws_sqldml_replication** パラメータの設定には関係ありません。

例

- **例 1 –** LDS.*pubs2* 論理コネクションのディストリビュータスレッドを停止します。

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

- **例 2 –** LDS.*pubs2* 論理コネクションのセーブインターバルを “0” に変更し、論理コネクションの DSI キューのメッセージを削除できるようにします。

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

- **例 3 –** スタンバイデータベースへの **truncate table** の複写を有効にします。

```
alter logical connection to LDS.pubs2
 set send_truncate_table to 'on'
```

使用法

- トランケートテーブルをウォームスタンバイデータベースにコピーするには、**send_truncate_table** オプションを “on” に設定します。
- **send_truncate_table** オプションは、アクティブデータベースとウォームスタンバイデータベースの両方が Adaptive Server バージョン 11.5 以降である場合にのみ “on” に設定します。
- **send_truncate_table to on** 句を指定すると、Replication Server は、複写するようマーク付けされたすべてのテーブルのウォームスタンバイデータベースに、**truncate table** の実行をコピーします。
- **alter logical connection** コマンドは、ウォームスタンバイアプリケーションの設定後にディストリビュータスレッドを無効にするときに使用します。データ

ベースを複写システムに追加すると、Replication Server では、データのサブスクリプションを処理するためのディストリビュータスレッドが作成されます。

- **set distribution off** 句は、論理コネクションのディストリビュータスレッドを無効にするために使用します。このオプションを使用するのは、データベースにウォームスタンバイが設定してあっても、そのデータベース内にはデータのサブスクリプションがなく、さらにそのデータベースが複写ストアドプロシージャを実行する送信元ではない場合です。このような論理データベースは、通常の複写に関与しないウォームスタンバイアプリケーションか、または論理レプリケートデータベースであると考えられます。
- **set distribution on** は、**set distribution off** を使用して論理コネクションのディストリビュータスレッドを無効にした後に、このディストリビュータスレッドを起動するために使用します。これを実行するのは、論理データベース内のデータの複写定義とサブスクリプションを作成する場合や、論理データベース内の複写ストアドプロシージャを開始する場合です。
- **suspend distributor** コマンドと **resume distributor** コマンドを使用すると、物理データベースコネクションまたは論理データベースコネクションのディストリビュータスレッドをサスペンドまたはレジュームできます。
- ウォームスタンバイアプリケーションの設定と管理の詳細については、『Replication Server 管理ガイド 第 1 巻』および『Replication Server 管理ガイド 第 2 巻』を参照してください。
- **configure replication server** コマンドを使用すると、現在の Replication Server を起点とするすべての論理コネクションに影響を与えるパラメータを設定できます。
- 論理コネクションが作成されると、論理コネクションの **save_interval** パラメータは、デフォルトで **'strict'** に設定されます。これにより、スタンバイデータベースにメッセージが適用される前に、そのメッセージを DSI キューから削除できなくなります。
  スタンバイデータベースが長時間使用できない場合は、Replication Server のキューが満杯になる可能性があります。このような状態を防ぐには、**save_interval** を **'strict'** から “0” (分) に変更します。これにより、Replication Server はキューを削除することができます。

  警告！ save_interval パラメータは、DSI キューだけに影響します。**materialization_save_interval** パラメータは、現在存在するマテリアライゼーションキューにだけ影響します。これらのパラメータは、ステーブルキューの領域不足による深刻な状況でのみリセットしてください。このパラメータを (**'strict'** から任意の分単位の時間に) リセットすると、スタンバイデータベースでメッセージのロスが発生する可能性があります。Replication Server では、このようなロスが検出されないため、ユーザ自身がスタンバイデータベースの整合性を確認する必要があります。
- 論理コネクションが作成されると、論理コネクションの **materialization_save_interval** パラメータは、デフォルトで **'strict'** に設定されま

す。これにより、メッセージがスタンバイデータベースに適用される前にマテリアライゼーションキューから削除できなくなります。
スタンバイデータベースが長時間使用できない場合は、Replication Server のキューが満杯になる可能性があります。このような状態を防ぐには、**materialization_save_interval** を **'strict'** から “0” (分) に変更します。これにより、Replication Server はキューを削除することができます。

参照：

- admin logical_status (64 ページ)
- configure replication server (238 ページ)
- create logical connection (338 ページ)
- resume distributor (439 ページ)
- suspend distributor (453 ページ)

alter partition

パーティションのサイズを変更します。

構文

```
alter partition logical_name [expand [size =size]]
```

パラメータ

- **logical_name –** パーティションの名前です。名前は識別子の規則に従う必要があります。この名前は、**drop partition** コマンドと **create partition** コマンドでも使用されます。
- **expand –** パーティションのサイズを増やすことを指定します。
- **size –** パーティションを増やすメガバイト数を指定します。デフォルト値は 2MB です。

例

- **例 1 –** この例では、論理パーティション *P1* のサイズを 50MB 増やします。
  ```
  alter partition P1 expand size = 50
  ```
- **例 2 –** この例では、論理パーティション *P2* のサイズを 2MB 増やします。
  ```
  alter partition P2
  ```

使用法

- **alter partition** を使用すると、ユーザは現在使用しているパーティションのサイズを拡張できます。この機能は、Replication Server でディスク領域を増やす必要があり、既存のパーティションの同じディスクにまだ使用できる領域がある場合に役立ちます。
- 物理ディスク領域が足りない場合、**alter partition** がアボートし、エラーメッセージが表示されます。パーティションに割り付けられる領域は、コマンドが適用される前と同じです。
- パーティションに割り付けることができる最大サイズは 1TB (約 1,000,000MB) です。

パーミッション

alter partition を実行できるのは "sa" ユーザだけです。

参照：

- admin disk_space (57 ページ)
- create partition (339 ページ)
- drop partition (417 ページ)

alter queue

16 キロバイトを超える大きいメッセージが検出されたときのステーブルキューの動作を指定します。このコマンドは、Replication Server のバージョンが 12.5 以降であり、Replication Server のサイトバージョンが 12.1 以前の場合にのみ適用されます。

構文

```
alter queue, q_number, q_type,
set sqm_xact_with_large_msg [to]     {skip | shutdown}
set sqm_cache_enable to “on | off”
set sqm_page_size to “numblocks"
set sqm_cache_size to "numpages"
```

パラメータ

- **q_number –** ステーブルキューのキュー番号です。
- **q_type –** ステーブルキューのキュータイプです。値は、アウトバウンドキューの場合は “0”、インバウンドキューの場合は “1” です。

- **sqm_xact_with_large_msg {skip | shutdown}** – 16 キロバイトを超えるメッセージが検出された場合に、SQM がそのメッセージをスキップするか、SQM が停止するかを指定します。
- **sqm_cache_enable to “on" | "off"** – ステーブルキューのキャッシュを有効または無効にします。キューレベルのキャッシュは、**configure replication server** を使用して設定されたサーバレベルのキャッシュよりも優先されます。**sqm_cache_enable** のデフォルト値は "on" です。
- **sqm_page_size** – ステーブルキューのページサイズを設定します。キューレベルでページサイズを設定すると、**configure replication server** を使用して設定されたサーバレベルのページサイズよりも優先されます。**sqm_page_size** のデフォルト値は 4 です。
- **"numblocks"** – ページ内の 16K ブロックの数を指定します。ページサイズを設定すると、Replication Server の I/O サイズも設定されます。たとえば、ページサイズを 4 に設定すると、64K チャンクでステーブルキューに書き込むよう Replication Server に指示されます。*numblocks* の許容値は 1 ～ 64 です。
- **sqm_cache_size** – ステーブルキューのキャッシュサイズを設定します。キューレベルでキャッシュサイズを設定すると、**configure replication server** を使用して設定されたサーバレベルのキャッシュサイズよりも優先されます。**sqm_cache_size** のデフォルト値は 16 です。
- **"numpages"** – キャッシュ内のページの数を指定します。値の範囲は 1 ～ 512 ページです。

例

- **例 1** – 大きいメッセージがキューに渡された場合に、キュー番号 2 を停止します。

```
alter queue, 2, 0, set sqm_xact_with_large_msg to
shutdown
```

使用法

- **sqm_cache_enable**、**sqm_page_size**、**sqm_cache_size** の各パラメータを変更した場合は、サーバを再起動して変更を有効にします。
- サイトバージョンが 12.5 以降の場合、**alter queue** は失敗します。

パーミッション

alter queue には、"sa" パーミッションが必要です。

参照：

- alter route (213 ページ)
- resume queue (441 ページ)

- resume route (442 ページ)

alter replication definition

既存の複写定義を変更します。

構文

```
alter replication definition {replication_definition
 |* with primary at data_server.database }
{with replicate table named [table_owner.]'table_name' | alter
[primary] owner from current_table_owner to new _table_owner  |add
column_name [as replicate_column_name]
    [datatype [null | not null]]
    [map to published_datatype] [quoted],... |
alter columns with column_name
    [as replicate_column_name] [quoted | not quoted],...|
alter columns with column_name
datatype [null | not null]
    [map to published_datatype],...|
    references {[table_owner.]table_name [(column_name) | null}]
alter columns column_name {quoted | not quoted}
add primary key column_name [, column_name]... |
drop primary key column_name [, column_name]... |
add searchable columns column_name [, column_name]... |
drop searchable columns column_name [, column_name]... |
drop column_name[, column_name] ... |
send standby [off | {all | replication definition} columns] |
replicate {minimal | all} columns |
replicate {SQLDML ['off'] | 'options'} |
replicate_if_changed column_name [, column_name]...|
always_replicate column_name [, column_name]... |
{with | without} dynamic sql |
alter replicate table name {quoted | not quoted}}
[with DSI_suspended]
```

パラメータ

- **{*replication_definition* | * } –**
  - *replication_definition* - 変更する 1 つの複写定義の名前を指定します。
  - * ワイルドカード文字 - 所有者の譲渡の影響を受ける複数の複写定義で所有者を変更する場合に、**with primary at *data_server. database*** および **alter [primary] owner from *current_table_owner* to *new _table_owner*** 句でのみ使用します。この場合、*current_table owner* は、すべての複写定義で同じです。
- **with replicate table named –** レプリケートデータベースのテーブルの名前を指定します。 *table_name* 最大 200 文字の文字列です。 *table_owner* は、テーブル名のオプション修飾子であり、テーブルの所有者を表します。 実際のテーブルの

所有者が、複写定義内に指定されたテーブル所有者と一致しない場合には、データサーバのオペレーションが失敗する可能性があります。

- **alter [primary] owner from *current_table_owner* to *new _table_owner* –** テーブルの所有者を変更する場合は、現在の所有者と新しい所有者を指定します。

  Replication Server が DDL コマンドを複写しない場合、プライマリテーブルの所有者のみを変更するには **primary** オプションを含めます。
- **with primary at *data_server.database* –** データサーバとプライマリデータを格納するデータベースを指定します。

  この句は ***** ワイルドカード文字および **alter [primary] owner from *current_table_owner* to *new _table_owner*** 句でのみ使用します。

  複写定義名ではなく ***** ワイルドカードを使用するときに、目的のプライマリデータベースで影響を受ける複写定義のみが確実に変更されるようにするには、**alter [primary] owner from *current_table_owner* to *new _table_owner*** を使用してテーブル所有者を変更するときに、**with primary at *data_server.database*** 句を含めます。 プライマリデータベースがウォームスタンバイアプリケーションの一部である場合、*data_server.database* は論理データサーバと論理データベースの名前になります。
- **add columns column_name –** 複写定義に追加するカラムとそのデータ型を指定します。 *column_name* は、複写カラムリストに追加するカラムの名前です。 カラム名は、複写定義に対してユニークである必要があります。

  add columns*declared_column_name* も同様です。「カラムレベルのデータ型変換の使用」を参照してください。
- **as replicate_column_name –** 複写定義に追加するカラムに対して、プライマリカラムからのデータが複写されるレプリケートテーブル内のカラム名を指定します。*replicate_column_name* は、プライマリテーブル内の指定されたカラムに対応する、レプリケートテーブル内のカラムの名前です。この句は、レプリケートカラムとプライマリカラムの名前が異なる場合に使用します。
- **datatype –** 複写定義のカラムリストに追加するカラムのデータ型、または変更する既存のカラムのデータ型です。サポートされているデータ型とその構文のリストについては、「データ型」を参照してください。

  プライマリテーブルの既存の複写定義にカラムがリストされている場合は、同じプライマリテーブルの以降の複写定義で同じデータ型を指定します。

  カラムにカラムレベルのデータ型変換を指定する場合は、*declared_datatype* を使用します。宣言したデータ型は、Replication Server のネイティブデータ型か、プライマリデータ型のデータ型定義でなければなりません。
- **null または not null –** *text*、*unitext*、*image*、*rawobject* カラムにのみ適用されます。レプリケートテーブルで null 値を許可するかどうかを指定します。デフォ

ルトの **not null** は、レプリケートテーブルが null 値を受け入れないことを示します。

text、*unitext*、*image*、*rawobject* の各カラムの null ステータスは、同じプライマリテーブルのすべての複写定義と一致するとともに、実際のテーブル内の設定とも一致する必要があります。同じプライマリテーブルの既存の複写定義に *text*、*unitext*、*image*、または *rawobject* カラムが含まれている場合、null ステータスの指定は任意です。

- **quoted | not quoted –** テーブル名またはカラム名が引用符付き識別子かどうかを指定します。レプリケートに引用符を必要とする各オブジェクトで、引用符付き句を使用します。
- **alter columns column_name –** 複写定義で変更するカラムとそのデータ型を指定します。*column_name* は、変更するカラムの名前です。カラム名は、複写定義に対してユニークである必要があります。

  カラムレベルのデータ型変換を指定する場合は、**alter columns***declared_column_name* を使用します。
- **map to published_datatype –** カラムレベルのデータ型変換後のカラムのデータ型を指定します。*published_datatype* は、Replication Server のネイティブデータ型またはパブリッシュデータ型のデータ型定義であることが必要です。
- **references table owner.table name column name –** プライマリデータベースで参照元テーブルとして追加または変更する、参照制約を持つテーブルの名前を指定します。参照を削除するには **null** オプションを使用します。*table_name* は最大 200 文字までの文字列です。*table_owner* はオプションで、テーブルの所有者を示します。*column name* はオプションです。実際のテーブル所有者が、複写定義内に指定されたテーブル所有者と一致しない場合には、データサーバのオペレーションが失敗する可能性があります。使用方法の詳細については、「create replication definition」コマンドの「参照制約のあるテーブルの扱い」を参照してください。
- **add/drop primary key –** プライマリキーカラムリストにカラムを追加または削除する場合に使用します。Replication Server は、レプリケートテーブルまたはスタンバイテーブルでの正しいローの検出をプライマリキーに依存しています。すべてのプライマリキーカラムを削除するには、まず対応する複写定義を変更して新しいプライマリキーを追加してから、テーブル内の古いプライマリキーカラムを削除します。すべてのプライマリキーが消失していると、DSI は停止します。プライマリキーの詳細については、「**create replication definition**」を参照してください。
- **add searchable columns column_name – create subscription** コマンドまたは **define subscription** コマンドの **where** 句で使用できる追加カラムを指定します。*column_name* は、サーチャブルカラムリストに追加するカラムの名前です。それぞれの句で、同じカラム名を 2 回以上指定することはできません。

text、*unitext*、*image*、*rawobject*、*rawobject in row* カラム、または暗号化カラムをサーチャブルカラムとして指定することはできません。

- **drop searchable columns column_name –** サーチャブルカラムリストから削除するカラムを指定します。サーチャブルカラムリストからカラムを削除できるのは、サブスクリプションまたはアーティクルの **where** 句で使用されていない場合だけです。
- **drop column_name –** 削除するカラムを指定します。
- **send standby –** ウォームスタンバイアプリケーションで、スタンバイデータベースに複写するときの複写定義の使用方法を指定します。この句とそのオプションの使用方法の詳細については、「スタンバイデータベースへの複写」を参照してください。
- **replicate minimal columns –** レプリケートデータベースで更新オペレーションまたは削除オペレーションを実行する必要があるカラムだけを Replication Server に送信します。すべてのカラムを複写するには、**replicate all columns** を使用します。
- **replicate SQLDML [‘off’] –** 指定された DML オプションの SQL 文の複写を有効または無効にします。
- **replicate 'options' –** 次の DML オペレーションの任意の組み合わせを複写します。
  - U - **update**
  - D - **delete**
  - I - **insert select**
- **replicate_if_changed – replicate_if_changed** カラムリストに追加する *text*、*unitext*、*image*、または *rawobject* カラムを指定します。同じプライマリテーブルの複数の複写定義が存在する場合、この句を使用して 1 つの複写定義を変更すると、同じプライマリテーブルのすべての複写定義が変更されます。
- **always_replicate – always_replicate** カラムリストに追加する *text*、*image*、または *rawobject* カラムを指定します。同じプライマリテーブルの複数の複写定義が存在する場合、この句を使用して 1 つの複写定義を変更すると、同じプライマリテーブルのすべての複写定義が変更されます。
- **with dynamic sql –** コマンドが条件を満たしており、使用できるキャッシュ領域が十分にある場合に、DSI で動的 SQL をテーブルに適用することを指定します。これは、デフォルト値です。

  動的 SQL を使用するためにコマンドが満たす必要のある条件については、『Replication Server 管理ガイド 第 2 巻』を参照してください。
- **without dynamic sql –** DSI で動的 SQL コマンドを使用できないことを指定します。
- **with DSI_suspended –** スタンバイ DSI (存在する場合) と、各サブスクリプション複写 DSI スレッドをサスペンドできるようにします。Replication Server は、

古いバージョンの複写定義のデータをすべてスタンバイデータベースまたはレプリケートデータベースに適用した後に、スタンバイデータベースまたはレプリケートデータベースの DSI スレッドをサスペンドします。

Replication Server が DSI スレッドをサスペンドした後は、ターゲットスキーマおよび任意のカスタムファンクション文字列を変更できます。DSI スレッドをレジュームすると、Replication Server は変更された複写定義を使用してプライマリの更新を複写します。

次の場合、**with DSI_suspended** を使用する必要はありません。

- 複写定義へのサブスクリプションがない。
- カスタムファンクション文字列を変更する必要がない。
- レプリケートデータベースまたはスタンバイデータベースのスキーマを変更する必要がない。

注意： サイトバージョンが 1550 より古いレプリケート Replication Server からのサブスクリプションがある場合、その Replication Server のレプリケート DSI スレッドはサスペンドされません。

例

- **例 1 –** *state* をサーチャブルカラムとして *authors_rep* 複写定義に追加します。

```
alter replication definition authors_rep
 add searchable columns state
```

- **例 2 –** *titles_rep* 複写定義を変更して、削除オペレーションと更新オペレーションの対象となる最小限のカラムだけが送信されるように指定します。

```
alter replication definition titles_rep
 replicate minimal columns
```

- **例 3 –** *titles_rep* 複写定義を変更して、ユーザ “joe” が所有する *copy_titles* というレプリケートテーブルで複写定義のサブスクリプションを作成できるように指定します。

```
alter replication definition titles_rep
 with replicate table named joe.'copy_titles'
```

- **例 4 –** *pubs_rep* 複写定義を変更して、プライマリカラム *pub_name* がレプリケートカラム *pub_name_set* に複写されるように指定します。

```
alter replication definition pubs_rep
alter columns with pub_name as pub_name_set
```

- **例 5 –** *hire_date* カラム値を *rs_db2_date* (プライマリ) フォーマットからネイティブデータ型 *smalldatetime* (レプリケート) フォーマットに変換するカラムレベル変換を導入します。

```
alter replication definition employee_repdef
alter columns with hire_date as rs_db2_date
map to smalldatetime
```

- **例 6 –** レプリケートサイトに送信される際に、*foo* という名前のテーブルを引用符で囲みます。

```
alter replication definition repdef
alter replicate table name foo quoted
```

- **例 7 –** カラム *foo_col2* から引用符付き識別子のマークを削除します。

```
alter replication definition repdef
alter columns "foo_col2" not quoted
```

- **例 8 –** 複写定義カラム名を pub_name_set に変更し、古いカラム名 pub_name を使用して、キューに現在あるデータを処理し、キュー内のデータが Replication Server によって処理された後にターゲット DSI をサスペンドするように Replication Server に指示します。DSI が再開されると、Replication Server は変更された複写定義をターゲットデータベースに使用するようになります。

```
alter replication definition pubs_rep
alter columns with pub_name as pub_name_set
with DSI_suspended
```

- **例 9 –** *address*、*city*、*state*、*zip* の各カラムを "authors" 複写定義から削除します。

```
alter replication definition authors
drop address, city, state, zip
```

- **例 10 –** テーブル複写定義に参照関係を追加します。

```
alter replication definition doctors_rep
   alter columns with tlid
   references doctors_main (logid)
```

- **例 11 –** Adaptive Server **alter... modify owner** コマンドを使用してテーブル所有者を mario から angela に変更後、**authors_repdef** 複写定義のテーブル所有者を変更するため、すぐにプライマリデータベースで **rs_send_repserver_cmd** ストアドプロシージャを実行します。

```
exec rs_send_repserver_cmd 'alter replication definition
authors_repdef
alter owner from mario to angela'
```

- **例 12 –** Adaptive Server **alter... modify owner** コマンドを使用して、現在 mario が所有者であるすべてのテーブルの新しい所有者を angela に変更後、現在のテーブル所有者が mario になっているすべての複写定義のテーブル所有者を angela に変更するため、すぐにプライマリデータベースで **rs_send_repserver_cmd** ストアドプロシージャを実行します。

```
exec rs_send_repserver_cmd 'alter replication definition *
with primary at NY_DS.ny_pdb1
alter owner from mario to angela'
```

使用法

- **alter replication definition** コマンドは、次の方法で複写定義を変更するときに使用します。
  - プライマリキーの追加または削除
  - ターゲットレプリケートテーブル名の変更
  - テーブル所有者の変更 - プライマリ Adaptive Server データベースでのオブジェクト所有権の譲渡への変更に応じて調整するため、**rs_send_repserver_cmd** ストアドプロシージャをプライマリデータベースで使用して、**alter replication definition** を実行します。『Replication Server 管理ガイド第 1 巻』の「RepAgent の管理と Adaptive Server のサポート」の「データベースオブジェクトの所有権の譲渡」を参照してください。
  - ターゲットレプリケートカラム名の変更
  - カラムの追加、対応するターゲットレプリケートカラム名の表示
  - サーチャブルカラムの追加または削除
  - ウォームスタンバイアプリケーションによる複写定義の使用方法の変更
  - カラムデータ型の変更
  - 複写対象 (すべてのカラムまたは最少カラム) の変更
  - *text*、*unitext*、*image*、または *rawobject* カラムの複写ステータスの変更
  - カラムレベルのデータ型変換の導入または削除
  - DSI で動的 SQL アプリケーションのテーブルを含めるまたは除外する
- **alter replication definition** は、複写定義のプライマリサイトで実行します。
- テーブルレベルの複写定義を使用しないで暗号化カラムを複写するデータベース複写定義では、INIT_VECTOR NULL と PAD NULL を使用して暗号化カラムの暗号化キーを定義します。
- プライマリ Replication Server のバージョンがレプリケート Replication Server のバージョンよりも高い混合バージョン環境では、レプリケート Replication Server によってサポートされ、サブスクリプションが作成される複写定義の変更をレプリケート Replication Server がサポートできない場合、その複写定義を変更することはできません。ただし、レプリケート Replication Server が複写定義をサポートしていても、サブスクリプションを作成しない場合には、その複写定義は変更され、レプリケート Replication Server から削除されます。
- LOB 圧縮データのサブスクリプションマテリアライゼーションのサポートは、複写定義でのカラムのデータ型の指定方法、また、Replication Server のバージョンによって異なります。『Replication Server 管理ガイド 第 1 巻』の「LOB 圧縮データのサブスクリプションマテリアライゼーション」を参照してください。
- SQL 文の複写の詳細については、「SQL 文の複写」を参照してください。
- 「**create replication definition**」には、**alter replication definition** コマンドのオプションに関する詳細情報が記載されています。

カラムの追加

- カラムを追加する場合は、**alter replication definition** を複写定義の分配に合わせて実行します。エラーを避けるには、「複写定義の変更手順」で説明されている手順に従ってください。
- 複写定義に追加するカラムに `identity` カラムが含まれている場合、Transact-SQL の **identity_insert** オプションを使用して、テーブルに対するオペレーションを行うには、レプリケートデータベースでメンテナンスユーザがテーブルの所有者であるか、“dbo” または “dbo” のエイリアスであるか、sa_role のパーミッションを持っている必要があります。適切なパーミッションがない場合、`identity` カラムの複写を進めることができないときに Replication Server ログにエラーメッセージが表示されます。
  (1 つまたは複数の複写定義を持つ) プライマリテーブルに含めることができる `identity` カラムは 1 つだけです。 ただし、**map to** オプションを使用すると、1 つまたは複数の複写定義で複数のカラムを `identity` データ型としてパブリッシュできます。
- 複写定義に追加するカラムに `timestamp` カラムが含まれている場合、レプリケートデータベースでメンテナンスユーザがテーブルの所有者 (または “dbo” か “dbo” のエイリアス) である必要があります。 プライマリテーブルには、`timestamp` カラムを 1 つだけ含めることができます。

カラムの削除

- サイトバージョンが 1550 より古いレプリケート Replication Server からのサブスクリプションがある場合、プライマリ Replication Server はカラムを削除するための複写定義の変更要求を拒否します。

  注意： 複写定義を変更してカラムを削除する場合、サイトバージョンが 1550 より古いレプリケート Replication Server ではオートコレクションまたは動的 SQL 設定のリセットが必要になることがあります。
- プライマリテーブルに複数の複写定義がある場合、**alter replication definition** はコマンドラインの *repdef_name* で指定した複写定義のカラムのみを削除します。
- **drop** パラメータは、テーブル複写定義のカラムを削除します。カラムがプライマリキーまたはサーチャブルカラムの一部である場合、**drop** はプライマリキーリストまたはサーチャブルカラムリストのカラムを削除します。次のようなカラムの場合、Replication Server は、カラムを削除するための複写定義の変更要求を拒否します。
  - 唯一のカラム
  - 複写定義の唯一のプライマリキーカラム
  - サブスクリプションまたはアーティクルの **where** 句内

    - アーティクルまたはサブスクリプションの **where** 句に指定されているサーチャブルカラムの前

カラムデータ型の変更

- カラムデータ型がサブスクリプションまたはアーティクルの **where** 句で使用されている場合、カラムデータ型を変更することはできません。
- *rs_address* データ型は変更できません。
- カラムデータ型を *text*、*unitext*、*image*、*rawobject*、または *rawobject in row* データ型に変更できるのは、カラムがプライマリキーまたはサーチャブルカラムでない場合だけです。
- カラムのパブリッシュデータ型を変更するには、宣言したデータ型と **map to** オプションの両方を指定します。
- プライマリテーブルに複数の複写定義がある場合は、カラムの宣言した型と null 入力可能性がテーブルのすべての複写定義で一貫している必要があります。
- データ型の変更方法については、『Replication Server 管理ガイド 第 1 巻』を参照してください。
- null 入力可能性の変更 (**null** または **not null**) は、*text*、*unitext*、*image*、*rawobject* カラムに対してのみ行うことができます。

カラムレベルのデータ型変換の使用

- カラムレベルのデータ型変換を有効にするには、使用しているプラットフォームの『Replication Server 設定ガイド』の説明に従って、異機種データ型サポート (HDS) オブジェクトを設定し、インストールしておく必要があります。
- *text*、*unitext*、*image*、または *rawobject* データ型は、基本データ型またはデータ型定義として使用することはできません。また、カラムレベル変換やクラスレベル変換の変換元または変換先として使用することもできません。
- *declared_datatype* は、Replication Server に配信される値のデータ型によって、次のように異なります。
    - Replication Agent が Replication Server の基本データ型を配信する場合、*declared_datatype* はその基本データ型になる。
    - Replication Agent がその他のデータ型を配信する場合、*declared_datatype* はプライマリデータベースの元のデータ型のデータ型定義である必要がある。
- *published_datatype* は、カラムレベル変換を行った後、クラスレベル変換を行う前の値のデータ型です。*published_datatype* は、Replication Server のネイティブデータ型、または他のデータベースにあるデータ型のデータ型定義である必要があります。
- 複数の複写定義で宣言されたカラムは、各複写定義内で同じ *declared_datatype* を使用する必要があります。*published_datatype* は異なってもかまいません。

すべてのカラムまたは最少カラムの複写

- 複写定義に **replicate minimal column** オプションを使用すると、削除オペレーションまたは更新オペレーションを実行する必要がある最小限のカラムのデータだけがレプリケート Replication Server に送信されます。すべてのカラムを複写するには、**replicate all columns** を指定します。この機能の詳細については、「**create replication definition**」を参照してください。

  注意： 複写定義に **replicate all columns** 句が含まれ、かつ **replicate minimal columns** コネクションプロパティが 'on' に設定されている場合、そのコネクションは最少数のカラムをレプリケートします。ターゲットデータベースにカラムをすべてレプリケートするには、DSI コネクションの **replicate minimal columns** 値を 'off' に設定します。

スタンバイデータベースへの複写

- Replication Server では、ウォームスタンバイアプリケーション内のスタンバイデータベースを保持するために、複写定義は必要ありません。複写定義を使用すると、スタンバイデータベースへの複写のパフォーマンスが向上する場合があります。この目的のためだけに、論理データベース内の各テーブルに複写定義を作成できます。
- この複写定義を使用してこのテーブルのトランザクションをスタンバイデータベースに複写するには、**send standby** に **off** 以外のオプションを指定して使用します。 複写定義のプライマリキーカラムと **replicate minimal columns** の設定は、スタンバイデータベースへの複写に使用されます。 このメソッドのオプションには、次のものがあります。
  - **send standby** または **send standby all columns** を使用すると、すべてのプライマリテーブルカラムをスタンバイデータベースに複写できる。
  - 複写定義のカラムだけをスタンバイデータベースに複写するには、**send standby replication definition columns** を使用します。
- スタンバイデータベースに複写するときに、このテーブルのどの複写定義も使用しないことを示すには、**send standby off** を使用します。テーブル内のすべてのカラムがスタンバイデータベースに複写され、スタンバイデータベースへの複写には、テーブルのすべての複写定義に含まれるすべてのプライマリキーカラムを統合したものが使用されます。論理コネクションの **replicate_minimal_columns** 設定によって、更新と削除の対象として最少カラムを送信するか、すべてのカラムを送信するかが決まります。「**alter logical connection**」を参照してください。

  テーブルの複写定義が存在しない場合は、テーブル内のすべてのカラムがスタンバイデータベースに複写され、Replication Server によってプライマリキーが作成されます。この場合には、**replicate_minimal_columns** が on になります。

参照制約のあるテーブルの扱い

参照制約 (外部キーやその他の検査制約など) のあるテーブルの指定には複写定義を使用できます。それによって、RTL または HVAR を有効にしたときに、

Replication Server にそれらのテーブルの存在が通知されます。『Replication Server 管理ガイド 第 2 巻』の「パフォーマンスチューニング」の「Advanced Services Option」の「Adaptive Server への High-Volume Adaptive Replication」および『Replication Server 異機種間複写ガイド』の「レプリケートデータサーバとしての SAP IQ」の「SAP IQ レプリケートデータベースの設定」を参照してください。

複写定義の変更手順

複写定義への変更を要求すると、Replication Server によって複写定義の変更とデータ複写の伝達が自動的に調整されます。複写定義の変更は、プライマリ Replication Server で直接要求するか、データベーススキーマの変更中に、**alter replication definition**、**alter applied replication definition**、または **alter request function replication definition** コマンドを使用してプライマリデータベースで要求できます。

プライマリデータベースログには変更する複写定義のデータは含まれませんが、プライマリ Replication Server で複写定義要求を直接発行できます。それ以外の場合は、**rs_send_repserver_cmd** ストアドプロシージャを使用してプライマリデータベースで複写定義要求を発行すると安全です。

データベースが **rs_send_repserver_cmd** をサポートしていない場合は、変更中のスキーマのデータローがプライマリデータベースログに含まれなくなるまで待機してからプライマリ Replication Server で **alter replication definition** 要求を実行する必要があります。

『Replication Server 管理ガイド 第 1 巻』の「複写テーブルの管理」の「複写定義の変更要求プロセス」を参照してください。

パーミッション

alter replication definition には、“create object” パーミッションが必要です。

参照：

- admin verify_repserver_cmd (101 ページ)
- alter function string (188 ページ)
- create replication definition (346 ページ)
- drop replication definition (419 ページ)
- rs_set_quoted_identifier (578 ページ)
- rs_send_repserver_cmd (735 ページ)
- rs_helprepversion (728 ページ)

alter request function replication definition

create request function replication definition コマンドによって作成されたファンクション複写定義を変更します。

構文

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

パラメータ

- **repdef_name –** 変更する要求ファンクション複写定義の名前です。
- **with replicate function named 'func_name' –** レプリケートデータベースで実行するストアドプロシージャの名前を指定します。この複写定義のレプリケートファンクション名は、プライマリファンクション名と異なる必要があります。*func_name* は、最大 255 文字の文字列です。
- **add –** ファンクション複写定義に追加するパラメータとそのデータ型を指定します。
- **@param_name –** 複写パラメータまたはサーチャブルパラメータのリストに追加するパラメータの名前です。各パラメータ名は、@ 文字で始まる必要があります。
- **datatype –** パラメータリストに追加するパラメータのデータ型です。Adaptive Server のストアドプロシージャとファンクション複写定義には、*text*、*unitext*、*rawobject*、*image* の各データ型のパラメータを含めることはできません。
- **add searchable parameters – where** 句 (**create subscription** または **define subscription** コマンド内) で使用できる追加パラメータを指定します。
- **send standby –** ウォームスタンバイアプリケーションで、スタンバイデータベースにファンクションのすべてのパラメータを送信するか (**send standby all parameters**)、複写定義で指定されたパラメータだけを送信するか (**send standby replication definition parameters**) を指定します。デフォルトは、**send standby all parameters** です。
- **with DSI_suspended –** スタンバイ DSI (存在する場合) と、各サブスクリプション複写 DSI スレッドをサスペンドできるようにします。Replication Server は、古いバージョンの複写定義のデータをすべてスタンバイデータベースまたはレプリケートデータベースに適用した後に、スタンバイデータベースまたはレプリケートデータベースの DSI スレッドをサスペンドします。

Replication Server が DSI スレッドをサスペンドした後に、ターゲットストアドプロシージャおよび任意のカスタムファンクション文字列を変更できます。DSI スレッドをレジュームすると、Replication Server は変更された複写定義を使用してプライマリの更新を複写します。

次の場合、**with DSI_suspended** を使用する必要はありません。

- 複写定義へのサブスクリプションがない。
- カスタムファンクション文字列を変更する必要がない。
- レプリケートデータベースまたはスタンバイデータベースのストアドプロシージャを変更する必要がない。

注意： サイトバージョンが 1550 より古いレプリケート Replication Server からのサブスクリプションがある場合、その Replication Server のレプリケート DSI スレッドはサスペンドされません。

例

- **例 1 –** *@notes*、*@pubdate*、*@contract* の各パラメータを **titles_frep** ファンクション複写定義に追加します。

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

- **例 2 – titles_frep** ファンクション複写定義のサーチャブルパラメータのリストに、*@type* パラメータと *@pubdate* パラメータを追加します。

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

- **例 3 – titles_frep** ファンクション複写定義を変更してレプリケートデータベースで **newtitles** ストアドプロシージャとして複写されるようにし、**alter request replication definition** の実行前に存在するプライマリデータがレプリケートデータベースに複写された後に、ターゲット DSI をサスペンドするように Replication Server に指示します。

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

使用法

- **alter request function replication definition** は、既存の要求ファンクション複写定義を変更するときに使用します。複写パラメータやサーチャブルパラメータを追加したり、ウォームスタンバイに送信するパラメータを選択したりできま

す。また、レプリケートデータベースで実行するストアドプロシージャに別の名前を指定することもできます。

- **alter request function replication definition** によって変更できるのは、**create request function replication definition** コマンドで作成された複写定義だけです。
- ファンクション複写定義を変更する場合、ファンクション複写定義に指定した名前、パラメータ、データ型が複写するストアドプロシージャと一致する必要があります。ファンクション複写定義で指定したパラメータだけが複写されます。
- 同じストアドプロシージャの複数のファンクション複写定義には、同じパラメータリストが必要です。新しいパラメータを追加すると、そのストアドプロシージャ用に作成されたすべてのファンクション複写定義に自動的に追加されます。
- **alter request function replication definition** コマンドは、ファンクション複写定義を作成したプライマリ Replication Server で実行します。
- 1 つの句の中で同じパラメータ名を 2 回以上指定することはできません。
- パラメータを追加するときは、ファンクション複写定義の分配に合わせて、**alter request function replication definition** を調整するように Replication Server に指示する必要があります。また、ストアドプロシージャと複写定義への変更も調整するように Replication Server に指示する必要があります。

  『Replication Server 管理ガイド 第 1 巻』の「複写テーブルの管理」の「複写定義の変更要求プロセス」を参照して複写定義を変更してください。
- レプリケートデータベースで実行するストアドプロシージャの名前を指定するには、**with replicate function named** 句を使用します。「**create request function replication definition**」を参照してください。

要求ファンクション複写定義の変更方法の詳細については、『Replication Server 管理ガイド 第 1 巻』を参照してください。

パーミッション

alter request function replication definition には、“create object” パーミッションが必要です。

参照：

- alter function string (188 ページ)
- alter applied function replication definition (129 ページ)
- create applied function replication definition (274 ページ)
- create request function replication definition (362 ページ)
- drop function replication definition (411 ページ)
- rs_send_repserver_cmd (735 ページ)
- rs_helprepversion (728 ページ)

alter route

現在の Replication Server からリモート Replication Server へのルートの属性を変更します。

構文

```
alter route to dest_replication_server {
    set next site [to] thru_replication_server |
    set username [to] 'user' set password [to] 'passwd' |
    set password [to] 'passwd' |
    set route_param [to] 'value' |
    set security_param [to] 'value' |
    set security_services [to] 'default'}
```

パラメータ

- **dest_replication_server –** ルートを変更する送信先 Replication Server の名前です。
- **thru_replication_server –** 送信先 Replication Server へのメッセージが通過する中間 Replication Server の名前です。
- **user –** ルートに使用するログイン名です。
- **passwd –** このログイン名に使用するパスワードです。
- **route_param –** ルートに影響するパラメータです。パラメータと値のリストについては、「表 20 : ルートに影響を与える設定パラメータ」を参照してください。
- **value –** *route_param* の設定値です。文字列を指定します。

表 20 : ルートに影響を与える設定パラメータ

route_param	値
disk_affinity	次のパーティションを割り当てるための割り付けヒントを指定する。現在のパーティションが満杯になった場合に、次のセグメントの割り付け先となるパーティションの論理名を入力する。 デフォルト値は off
rsi_batch_size	トランケーションポイントが要求される前に、別の Replication Server に送信されるバイト数。 デフォルト値は 256KB 最小値：1KB 最大値：128MB

route_param	値
rsi_fadeout_time	Replication Server が送信先 Replication Server とのコネクションをクローズするまでのアイドル時間 (秒単位)。 デフォルト値は-1 (Replication Server がコネクションをクローズしないように指定する)
rsi_packet_size	他の Replication Server との通信に使用するパケットサイズ (バイト単位)。値の範囲は 1,024 ～ 16,384 バイト。 デフォルト値は 4096 バイト
rsi_sync_interval	この秒数の間、RSI キューに新しいメッセージがない状態が続くと、トランケーションポイントが要求される。 この値は 0 よりも大きくなければならない。 デフォルト値は 60 秒
rsi_xact_with_large_msg	サイズの大きいメッセージが検出された場合のルートの動作を指定する。このパラメータは、レプリケートサイトのサイトバージョンが 12.1 以前で、直接ルートの場合にのみ適用される。指定できる値は “skip” または “shutdown”。 デフォルト値は shutdown
save_interval	メッセージが送信先 Replication Server に正常に渡された後に、Replication Server がそのメッセージを保存しておく時間 (分単位)。詳細については、『Replication Server 管理ガイド 第 2 巻』を参照してください。 デフォルト値は 0 分

- **security_param –** セキュリティパラメータの名前を指定します。**alter route** を使用して設定できるセキュリティパラメータのリストと説明については、「表 20 : ルートに影響を与える設定パラメータ」を参照してください。
- **set security_services [to] 'default' –** Replication Server のグローバル設定と一致させるために、コネクションのすべてのネットワークベースセキュリティ機能をリセットします。

例

- **例 1 –** 例 1 と例 2 では、Tokyo Replication Server (TOKYO_RS) から San Francisco Replication Server (SF_RS) と Sydney Replication Server (SYDNEY_RS) への直接ルートが存在します。次のコマンドを使用すると、1 つ直接ルートが間接ルートに変更されるため、TOKYO_RS は、SF_RS を経由して SYDNEY_RS 宛てのメッセージを渡します。

  SF_RS で次のコマンドを入力すると、新しい間接ルートで使用される SYDNEY_RS への直接ルートが作成されます。

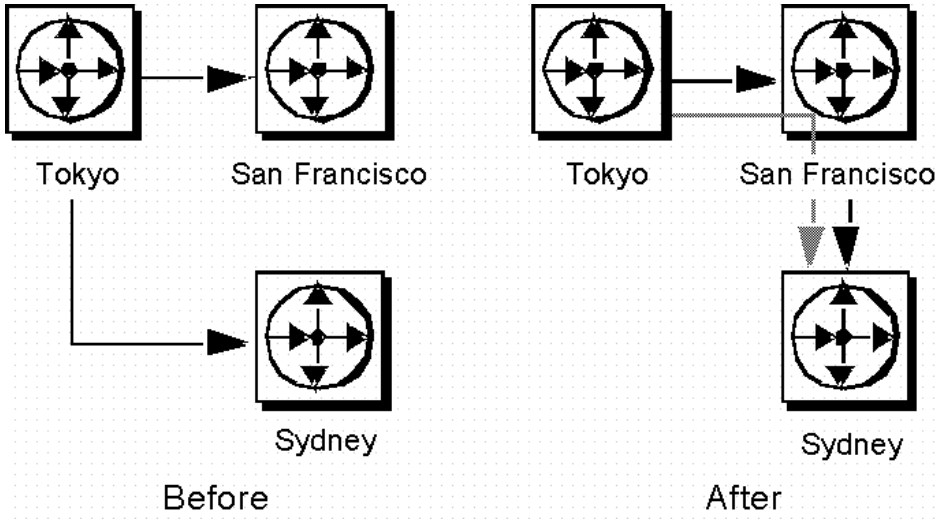
```
create route to SYDNEY_RS
set username SYDNEY_rsi_user
set password SYDNEY_rsi_passwd
```

- **例 2 –** TOKYO_RS で次のコマンドを入力すると、TOKYO_RS から SYDNEY_RS への直接ルートが間接ルートに変更され、中間 Replication Server として SF_RS が指定されます。

```
alter route to SYDNEY_RS
 set next site SF_RS
```

この図は、ルート指定スキームの変更前と変更後のルートを示しています。

図 1 : 例 1 と例 2 でルート指定を変更する前と変更した後



例 3 と例 4 では、TOKYO_RS は SF_RS を経由して SYDNEY_RS にメッセージを渡すのではなく、SYDNEY_RS にメッセージを直接送信するように、ルート指定をもう一度変更します。

- **例 3 –** TOKYO_RS で次のコマンドを入力すると、TOKYO_RS から SYDNEY_RS へのルートが間接ルートから直接ルートに変更されます。

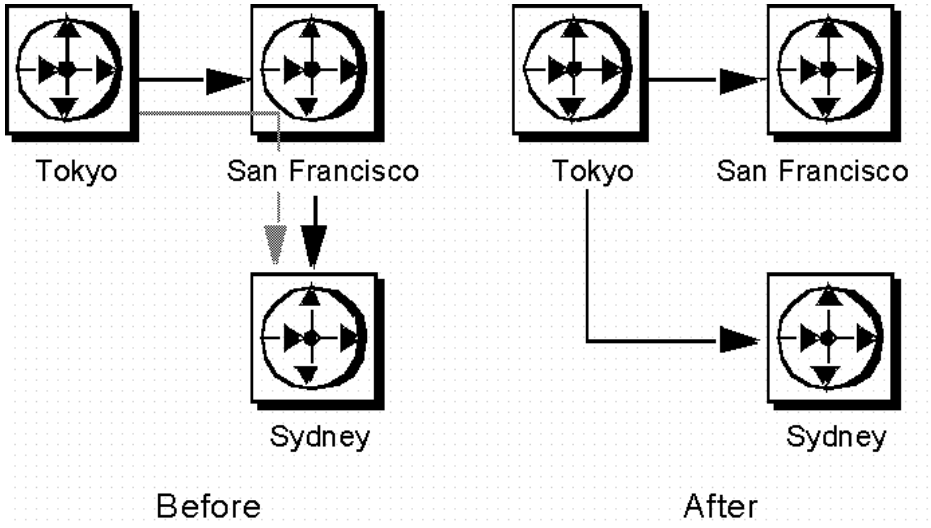
```
alter route to SYDNEY_RS
 set username SYDNEY_rsi
 set password SYDNEY_rsi_passwd
```

- **例 4 –** SF_RS で次のコマンドを入力すると、SF_RS から SYDNEY_RS への直接ルートが削除されます。

```
drop route to SYDNEY_RS
```

例 3 と例 4 のコマンドをともに使用すると、例 1 と例 2 の反映がキャンセルされます。この図は、2 番目のコマンドのセットが入力された後のルートを示しています。

図 2 : ルート指定の変更後



例 5 では、TOKYO_RS から SYDNEY_RS への直接ルート、SYDNEY_RS から SF_RS への直接ルート、TOKYO_RS から SYDNEY_RS を経由した SF_RS への間接ルートがあります。この例では、TOKYO_RS が SF_RS 宛てのメッセージを、別の Replication Server であるロサンゼルス内の LA_RS を経由して渡すように、このルート指定スキームを変更します。

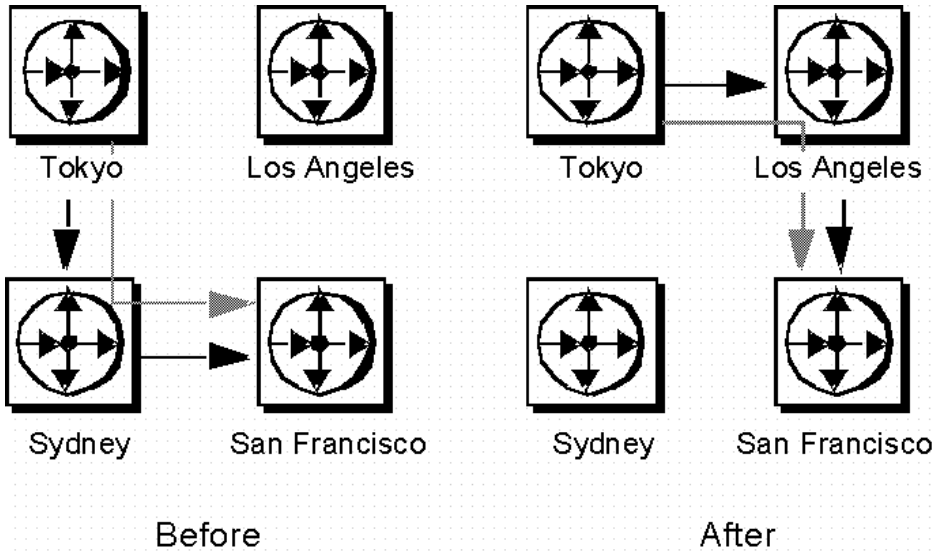
- **例 5 –** このコマンドを TOKYO_RS で入力すると、間接ルートの中間 Replication Server が、SYDNEY_RS から LA_RS に変更されます。

```
alter route to SF_RS
 set next site LA_RS
```

ルートを変更する前に、TOKYO_RS から LA_RS へと、LA_RS から SF_RS への直接ルートを作成しておく必要があります。

この図は、必要なコマンドを入力する前と後のルートを示しています。(SYDNEY_RS との間の直接ルートは削除されている可能性があるため、この図には示されていません)。

図 3 : 必要なコマンドの実行前と実行後



- **例 6 –** TOKYO_RS で次のコマンドを入力すると、TOKYO_RS から LA_RS への直接ルートのパスワードが変更されます。新しいパスワードは “LApass” です。

```
alter route to LA_RS
 set password LApass
```

  suspend route を使用して直接ルートをサスペンドしてから、ルートのパスワードを変更します。

- **例 7 –** LA_RS へのルートのセキュリティサービスを DCE に設定します。

```
suspend route to LA_RS

 alter route to LA_RS
 set security_mechanism to 'dce'

 resume route to LA_RS
```

使用法

- **alter route** は、次の変更を行う場合に使用します。
  - 直接ルートから間接ルートへの変更
  - 間接ルートから直接ルートへの変更
  - 既存ルート内の次の中間サイトの変更
  - 既存直接ルートの RSI ユーザのパスワードの変更
  - ルート設定パラメータの変更
  - ネットワークベースセキュリティのパラメータの変更

ルートの概要については、『Replication Server 管理ガイド 第 1 巻』を参照してください。

- **alter route** は、直接ルートの送信元となる Replication Server で実行します。
- **set next site***thru_replication_server* は、直接ルートを間接ルートに変更するとき、または間接ルートの中間サイトを変更するときに使用します。
- 直接ルートを間接ルートに変更する場合は、最初に送信元サイトから中間サイトへの直接ルートと、中間サイトから送信先サイトへの直接ルートを作成してください。この作業には、**create route** を使用します。
- 間接ルート内の中間サイトを変更する場合は、最初に新しい中間サイトから送信先サイトへの直接ルートと、新しい中間サイトから送信先サイトへの直接ルートを作成してください。この作業には、**create route** を使用します。
- 間接ルートには、1 つ以上の中間 Replication Server があります。たとえば、A_RS から D_RS への間接ルートの場合、中間サイトである B_RS と C_RS を経由することになります。
- 間接ルートを直接ルートに変更するには、**set next site** 句を指定していない **alter route** を使用して、送信先 Replication Server で使用するログイン名とパスワードを指定します。たとえば、間接ルート A_RS->B_RS->C_RS を、直接ルート A_RS->C_RS に変更できます。
- 1 つの中間サイトを次の中間サイトと交換する場合は、**alter route** を **set next site** 句を指定して実行します。たとえば、間接ルート A_RS->B_RS->C_RS->D_RS を、A_RS->C_RS->D_RS に変更できます。
- ルートパラメータは、**configure route** または **alter route** パラメータを使用して設定できます。
- ルート上のアクティビティを変更する前に、そのアクティビティをサスペンドするには、**suspend route** を使用します。

パスワードとユーザ名の設定

- **set username** *user* と **set password***passwd* は、間接ルートを直接ルートに変更する場合にのみ使用します。間接ルートのユーザ名またはパスワードは変更できません。変更しようとすると、間接ルートは直接ルートに変更されます。
- **set password***passwd* は、直接ルートのパスワードを変更する場合にのみ使用します。直接ルートのパスワードを変更する前に、**suspend route** を使用してください。

ルートパラメータ

- セーブインターバルを設定すると、システムは送信先 Replication Server でのパーティションまたはステーブルキューの失敗を許容できるようになります。**rebuild queues** コマンドを使用したリカバリ中に、バックログメッセージが送信先 Replication Server に送信されます。

セーブインターバルとステーブルキューのリカバリの詳細については、『Replication Server 管理ガイド 第 2 巻』を参照してください。

- パフォーマンスを最適化するために、**rsi_batch_size** パラメータ、**rsi_fadeout_time** パラメータ、**rsi_packet_size** パラメータ、**rsi_sync_interval** パラメータは、デフォルト値のままにしておくことをお奨めします。
- **alter route** を使用してルートパラメータを変更する前に、コネクションをサスペンドする必要があります。**alter route** コマンドを実行したら、変更を有効にするために、ルートをレジュームしてください。

ネットワークベースセキュリティのパラメータ

- ルートの両端では、同じセキュリティメカニズムとセキュリティ機能を備えた互換性のある SCL (Security Control Layer) ドライバを使用してください。各サーバについて、セキュリティ機能の選択と設定を行うのは複写システム管理者の仕事です。Replication Server は、リモートサーバとのコネクションを確立しようとする前に、そのサーバのセキュリティ機能の問い合わせは行いません。ルートの両端のセキュリティ機能に互換性がないと、そのコネクションは失敗します。
- **alter route** を使用すると、Replication Server からターゲット Replication Server への送信コネクションのネットワークベースセキュリティ設定を変更できます。**alter route** を使用して設定したセキュリティパラメータは、**configure replication server** を使用して設定したデフォルト値を上書きします。
- **unified_login** を “required” に設定すると、“sa” ユーザだけがクレデンシャルなしで Replication Server にログインできます。セキュリティメカニズムに問題が発生した場合でも、“sa” ユーザはパスワードを使用して Replication Server にログインし、**unified_login** を無効にできます。
- Replication Server には、複数のセキュリティメカニズムを装備できます。サポートされるメカニズムは、それぞれ `libtcl.cfg` ファイル内の SECURITY セクションにリストされています。
- メッセージの暗号化は、深刻なパフォーマンスの低下をともなう、負荷の高い処理です。通常は、特定のコネクションに対してだけ **msg_confidentiality** を “on” に設定してください。代わりに、**msg_integrity** などの負荷の低いセキュリティ機能を選択します。
- **alter route** を使用してセキュリティパラメータを変更する前に、コネクションをサスペンドする必要があります。**alter route** を実行したら、ルートをレジュームして変更を有効にしてください。

ルートの変更手順

注意： 設定パラメータを変更する場合、**alter route** を実行する前に必要な作業は、ルートのサスペンドだけです。

1. 複写システムをクワイスします。詳細については、『Replication Server トラブルシューティングガイド』を参照してください。
2. RepAgent を使用してデータベースを管理している各 Replication Server で、**suspend log transfer** を使用してログ転送をサスペンドします。
3. 送信元 Replication Server で、**alter route** コマンドを実行します。必要な数だけルートを変更できます。
4. **resume log transfer** を使用して、各 RSSD とユーザデータベースへの RepAgent コネクションをレジュームします。
   ルート変更手順の詳細については、『Replication Server 管理ガイド 第 1 巻』を参照してください。

パーミッション

alter route には、"sa" パーミッションが必要です。

参照：

- admin quiesce_check (66 ページ)
- admin quiesce_force_rsi (68 ページ)
- alter connection (134 ページ)
- alter logical connection (191 ページ)
- alter queue (197 ページ)
- configure connection (238 ページ)
- create logical connection (338 ページ)
- create replication definition (346 ページ)
- configure replication server (238 ページ)
- drop logical connection (416 ページ)
- create connection (287 ページ)
- create route (368 ページ)
- drop connection (407 ページ)
- drop route (421 ページ)
- resume log transfer (440 ページ)
- set proxy (448 ページ)
- suspend log transfer (454 ページ)
- suspend route (455 ページ)

alter schedule

コマンドを実行するスケジュールを有効または無効にします。

構文

alter schedule *sched_name* set [on|off]

パラメータ

- **sched_name –** 変更するスケジュールの名前です。
- **set [on | off} –** スケジュールを有効化または無効化します。デフォルトで、スケジュールは作成後にオンになります。

例

- **例 1 –** schedule1 を無効にするには、次のように入力します。

```
alter schedule schedule1 set off
```

使用法

Replication Server でスケジュールを有効化または無効化します。

パーミッション

alter schedule には、“sa” パーミッションが必要です。

参照：

- alter connection (134 ページ)
- drop schedule (423 ページ)
- configure replication server (238 ページ)
- create schedule (373 ページ)

alter subscription

同じ Replication Server を使用している同じレプリケートデータベースのレプリケートコネクション間で、再マテリアライズなしにサブスクリプションを移動します。サブスクリプションは、データベース複写定義、テーブル複写定義、ファンクション複写定義、またはパブリケーションに対して作成できます。

構文

```
alter subscription sub_name
     for {table_repdef | func_repdef | {publication pub | database
replication definition db_repdef}
     with primary at pri_dataserver.pri_database
     move replicate from data_server1.database1 to
data_server2.database2
```

パラメータ

- **sub_name –** サブスクリプションの名前です。この名前は識別子の規則に従う必要があります。サブスクリプションの名前は、複写定義 (適用される場合) と、レプリケートデータサーバおよびデータベースに対してユニークでなければなりません。
- **for table_rep_def –** サブスクリプションの対象となるテーブル複写定義を指定します。
- **for function_rep_def –** サブスクリプションの対象となるファンクション複写定義の名前を指定します。
- **for publication pub_name –** サブスクリプションの対象となるパブリケーションを指定します。
- **for database replication definition db_repdef –** サブスクリプションの対象となるデータベース複写定義を指定します。
- **with primary at *data_server.database* –** この句は、パブリケーションまたはデータベース複写定義のサブスクリプションに指定します。プライマリデータのロケーションを指定します。プライマリデータベースが論理コネクションを使用するウォームスタンバイアプリケーションの一部である場合、*data_server.database* は論理データサーバと論理データベースの名前になります。Multi-Path Replication システムを設定している場合、句内に代替プライマリコネクション名を指定することもできます。
- **レプリケートを *data_server1.database1* から *data_server2.database2* へ移動します –** *sub_name* サブスクリプションを *data_server1.database1* レプリケートコネクションから *data_server2.database2* コネクションへ移動するように指定します。

  レプリケートデータベースがウォームスタンバイアプリケーションの一部である場合、*data_server.database* は論理データサーバと論理データベースの名前になります。Multi-Path Replication システムを設定している場合、句内に代替レプリケートコネクション名を指定することもできます。

例

- **例 1 –** たとえば、**rep1** 複写定義の **sub1** サブスクリプションを `RDS.rdb1` コネクションから `RDS.rdb2` コネクションへ移動するには、次のように入力します。

```
alter subscription sub1 for rep1
move replicate from RDS.rdb1
to RDS.rdb2
```

使用法

- 複数の代替レプリケートコネクションを作成した場合は、**alter subscription** を使用して、レプリケートコネクション間でサブスクリプションを移動します。『Replication Server 管理ガイド 第 2 巻』の「パフォーマンスチューニング」の「Multi-Path Replication」で「コネクション間でのサブスクリプションの移動」を参照してください。
- **alter subscription** は、複写データを格納するデータベースの Replication Server で実行します。
- サブスクリプションの詳細とレプリケーションにおけるサブスクリプションの役割については、『Replication Server 管理ガイド 第 1 巻』を参照してください。
- プライマリ Replication Server のバージョンが 1570 より前の場合は、**alter subscription** を使用できません。代わりに、目的のコネクションでサブスクリプションを削除して再作成してください。
- 同じパスを使用してレプリケートしなければならない複数のサブスクリプションを移動するには、プライマリコネクションのログ転送をサスペンドし、サブスクリプションをすべて移動してからログ転送を再開してください。

パーミッション

create subscription を実行するには、次のログイン名とパーミッションが必要です。

- レプリケート Replication Server、プライマリ Replication Server、プライマリ Adaptive Server データベースで、同じログイン名とパスワードが必要です。
- このコマンドを入力するレプリケート Replication Server では、“create object” または “sa” パーミッションが必要です。
- プライマリ Replication Server では、“create object”、“primary subscribe”、または “sa” パーミッションが必要です。
- プライマリ Adaptive Server データベース内のプライマリテーブルに対しては、**select** パーミッションが必要です。
- プライマリ Araptive Server データベース内の **rs_marker** ストアドプロシージャに対しては、**execute** パーミッションが必要です。
- レプリケートデータベースのメンテナンスユーザには、レプリケートテーブルに対する **select**、**insert**、**update**、および **delete** の各パーミッションと、レプリケーションで使用されるファンクションに対する **execute** パーミッションが必要です。

参照：

- create alternate connection (269 ページ)
- create subscription (376 ページ)

alter user

ユーザのパスワードを変更します。

構文

```
alter user user
set password {new_password | null}
[verify password old_password]
   [set password_parameter to 'parameter_value']
```

パラメータ

- **user –** ユーザのログイン名です。
- **new_password –** パスワードを作成または変更する場合、新しいパスワード。
- **old_password –** *verify password* パラメータを使用する場合、現在のユーザパスワード。
- **parameter および parameter_value –** 設定できるパラメータおよび対応する値。

表 21 : パスワードパラメータ

password_parameter	**説明と*値***
password_expiration	パスワードの有効期限が切れてから経過した日数。 デフォルトのゼロ (0) は、パスワードの期限が切れないことを示します。 パスワードの期限が切れた場合、Replication Server はユーザアカウントをロックし、ユーザに通知します。パスワードを変更しなかった場合、切断後は管理者がパスワードをリセットするまでログインできなくなります。新しいパスワードは、パスワード要件をすべて満たす必要があります。 **rs_init** が **connect source** パーミッションまたは ID ユーザで作成するユーザのパスワードには、有効期限はありません。そのようなパスワードは、Replication Server でユーザ全員に対して設定された **password_expiration** のどのような設定でもオーバーライドします。データベース、他の Replication Server、および Replication Agent では、**connect source** パーミッションがあるユーザ ID を使用します。 管理者は、レプリケーションエージェントまたは RSI 向けに作成されたユーザのパスワードの有効期限が切れないよう、配慮してください。

例

- **例 1 –** ユーザ “louise” がパスワードを “EnnuI” から “somNIfic” に変更します。

```
alter user louise
 set password somNIfic
 verify password EnnuI
```

- **例 2 –** ユーザ jsmith のパスワードの有効期間を 60 日に変更します。

```
alter user jsmith
set password to newpass
set password_expiration to '60'
```

使用法

- Replication Server で ERSSD を使用している場合は、次のように ERSSD のプライマリユーザパスワードを変更できます。

```
alter user user
set password new_password
```

  このユーザ名が ERSSD のプライマリユーザ名と一致する場合、ERSSD は *rs_users* テーブルを更新し、パスワードを変更するために ERSSD で **sp_password** を発行して、設定ファイルの *RSSD_primary_pw_enc* 行を更新します。
- “sa” パーミッションを持つユーザは、**verify password** 句を省略できます。その他のユーザが各自のパスワードを変更するには、この句を指定する必要があります。
- **alter user** コマンドを使用して個別のユーザに対して指定されたパスワード設定は、**configure replication server** コマンドを使用して設定された値をオーバーライドします。
  「**configure replication server**」の「表 24 : パスワードパラメータ」表を参照してください。

パーミッション

alter user を使用して別のユーザのパスワードを変更するときには、"sa" パーミッションが必要です。

参照：

- create user (393 ページ)
- drop user (429 ページ)

assign action

DSI スレッドによって受信したデータサーバエラーまたは Replication Server エラーに、Replication Server のエラー処理アクションを割り当てます。

構文

```
assign action
{ignore | warn | retry_forever | retry_log | log | retry_stop |
stop_replication}
for error_class
to server_error1 [, server_error2]...
```

パラメータ

- **ignore –** エラーを無視して処理を続けるように、Replication Server を設定します。**ignore** は、データサーバのエラーコードが正常実行または重要度の低い警告を示す場合に使用してください。
- **warn –** トランザクションのロールバックまたは実行の割り込みをしないで、ログファイル内の警告メッセージを表示するように、Replication Server を設定します。
- **retry_forever –** 割り当て済みエラーアクションが修正されるまでコマンドの再試行を続けるように Replication Server に指示します。 エラーアクションは、**alter connection** で **retry_forever** に割り当てます。 エラーアクションが割り当てられていない場合、デフォルトでは、エラー重大度が 17 であるとエラーメッセージが表示されます。

  たとえば、エラー番号 1105 は次のように表示されます。

  ```
  DSI encountered error 1105 with Severity 17. It is mapped to
  RETRY_FOREVER. DSI will retry the transaction until the error is
  fixed
  ```
- **retry_log –** トランザクションをロールバックしてリトライするように、Replication Server を設定します。リトライの回数は、**alter connection** を使用して設定します。リトライ後もエラーが継続する場合は、Replication Server は例外ログにそのトランザクションを書き込み、次のトランザクションを実行します。
- **log –** 現在のトランザクションをロールバックし、それを例外ログに書き込んで次のトランザクションを実行するように、Replication Server を設定します。
- **retry_stop –** トランザクションをロールバックしてリトライするように、Replication Server を設定します。リトライの回数は、**alter connection** を使用して設定します。リトライ後もエラーが続く場合、Replication Server はデータベースの複写をサスペンドします。

- **stop_replication –** 現在のトランザクションをロールバックし、データベースの複写をサスペンドするように Replication Server に指示します。この動作は、**suspend connection** を使用したときの動作と同じです。
- **error_class –** アクションを割り当てるエラークラスの名前です。
- **server_error –** データサーバまたは Replication Server のエラー番号です。

例

- **例 1 –** データサーバエラー 5701 と 5703 を無視するように、Replication Server に指示します。

```
assign action ignore
 for pubs2_db_err_class
 to 5701, 5703
```

- **例 2 –** Replication Server でローカウントエラー (エラー番号 5186) が発生した場合、警告します。

```
assign action warn
for rs_repserver_error_class to 5186
```

  ローカウントエラーが発生した場合、次のエラーメッセージが表示されます。

```
DSI_SQLDML_ROW_COUNT_INVALID 5186
Row count mismatch for the SQL Statement Replication
command executed on 'mydataserver.mydatabase'. The
command impacted 10 rows but it should impact 15 rows."
```

- **例 3 –** Replication Server で、エラー番号 1105 によって示されるリソースの問題が発生した場合、再試行を繰り返します。

```
assign action retry_forever
for rs_repserver_error_class to 1105
```

使用法

- **assign action** は、データサーバによって返されたエラーを処理する方法を Replication Server に指示するときに使用します。このコマンドは、以前にデータサーバエラーに割り当てられたアクションを上書きします。
- **assign action** は、**create error class** が実行されたプライマリサイトで実行します。
- Replication Server 15.6 以降では、ローカウントの検証エラーメッセージにテーブル名が表示されます。『Replication Server 管理ガイド 第 2 巻』の「エラーと例外の処理」の「データサーバのエラー処理」の「非 SQL 文の複写ローカウントの検証」の「ローカウントの検証エラーメッセージにテーブル名が表示される」を参照してください。

- エラークラスを使用する複写を作成する前に、エラークラスに対してアクションを割り当てておかなければなりません。アクティブな複写にアクションを割り当てると、予期しない結果が発生することがあります。
- データサーバエラーにアクションが割り当てられていない場合は、デフォルトのアクション **stop_replication** が実行されます。Replication Server エラーの場合、実行されるデフォルトのアクションは、発生したエラーのタイプによって異なります。サポートされている Replication Server エラー、およびこれらのエラーに関するデフォルトのアクションのリストについては、「表 22 : Replication Server エラークラスのエラー番号の最新情報」を参照してください。
- エラー状態に適したエラーアクションを割り当てるようにしてください。たとえば、**begin transaction** コマンドが失敗したときにデータサーバによって返されるエラーに **ignore** アクションを割り当てると、後続の **commit** または **rollback** コマンドによって、予測しないエラーが発生する可能性があります。
- データサーバは、Client/Server Interfaces のエラー処理メカニズムを通じて、Replication Server にエラーを返します。警告およびエラーメッセージは、Replication Server のログファイルに書き込まれます。
- Replication Server は、エラーアクションを、複写システムを通じて条件に合うサイトに分配します。通常の複写システムの遅延時間が原因で、変更はすぐには表示されません。

複数のエラーに対する処理

- オペレーションで複数のエラーが発生すると、Replication Server は、それらのエラーに対して最も重大なアクションを選んで実行します。たとえば、トランザクションがロールバックされ、**retry_log** アクションが割り当てられていることを 1 つのエラーが示し、トランザクションログが満杯で、**stop_replication** アクションが割り当てられていることを別のエラーが示す場合、トランザクションから両方のエラーが返されると、Replication Server は **stop_replication** アクションを実行します。次に、重大度の低い順にエラーアクションを示します。
  1.**ignore**
  2.**warn**
  3.**retry_forever**
  4. **retry_log**
  5. **log**
  6. **retry_stop**
  7. **stop_replication**

rs_sqlserver_error_class のエラーアクション

- Adaptive Server の事前に定義されたエラーアクションが、*rs_sqlserver_error_class* エラークラスに提供されています。

- *rs_sqlserver_error_class* に異なるエラーアクションを割り当てるには、まずエラー・クラスのプライマリサイトを選択します。そのサイトの Replication Server にログインし、**create error class** を使用してエラークラスを作成してください。

rs_repserver_error_class のエラーアクション

- Replication Server の事前に定義されたエラーアクションが、**rs_repserver_error_class** エラークラスに提供されています。
- **rs_repserver_error_class** に異なるアクションを割り当てるには、まずエラー・クラスのプライマリサイトを選択します。プライマリサイトの Replication Server にログインし、**create replication server error class** を使用してエラークラスを作成します。
- 有効な Replication Server エラー、およびこれらのエラーに関するデフォルトのアクションのリストについては、「Replication Server エラークラスのエラー番号の最新情報」の表を参照してください。

表 22 : Replication Server エラークラスのエラー番号の最新情報

server_error	エラーメッセージ	デフォルトのエラーアクション	説明
5185	`Row count mismatch for the command executed on 'data-server.database'. The command impacted x rows but it should impact y rows.`	**stop_replication**	このメッセージは、SQL 文の複写、ストアドプロシージャ、またはオートコレクションが有効になっているロー変更の一部ではないコマンドがデータサーバに送られた後、影響を受けたロー数が予期されたロー数とは異なる場合に表示される。
5186	`Row count mismatch for the command executed on 'data-server.database'. The command impacted x rows but it should impact y rows.`	**stop_replication**	影響を受けたローの数が想定された数と異なる場合の SQL 文の複写におけるローカウントの検証エラー。

server_error	エラーメッセージ	デフォルトのエラーアクション	説明
5187	`Row count mismatch for the autocorrection delete command executed on 'data-server.database'. The command deleted x rows but it should delete y rows.`	**stop_replication**	このメッセージは、オートコレクションが有効な場合、delete コマンドがデータサーバに送られた後、影響を受けたロー数が予期されたロー数とは異なる場合に表示される。
5193	`You cannot enable autocorrection if SQL Statement Replication is enabled. Either enable SQL Statement Replication only or disable SQL StatementReplication before you enable autocorrection.`	**stop_replication**	SQL 文の複写が有効になっている場合、オートコレクションを有効にできない。オートコレクションを有効にする前に、SQL 文の複写を有効にするか、SQL 文の複写を無効にする。
5203	`Row count mismatch on 'dataserver.database'. The delete command generated by dsi_command_convert deleted x rows, whereas it should delete y rows.`	**stop_replication**	このメッセージは、削除されたロー数が予期された削除ロー数とは異なる場合に表示される。

- **rs_repserver_error_class** の詳細については、「エラークラスとファンクションクラス」の表を参照してください。

エラーアクションの表示

rs_helperror ストアドプロシージャは、指定したデータサーバのエラー番号にマップされた Replication Server のエラーアクションを表示します。

パーミッション

assign action には、"sa" パーミッションが必要です。

参照：

- alter error class (182 ページ)
- configure connection (238 ページ)
- create connection (287 ページ)
- create error class (308 ページ)
- drop error class (409 ページ)

- rs_helperror (705 ページ)
- suspend connection (452 ページ)

check publication

パブリケーションのステータスと、そのパブリケーションに含まれるアーティクルの数を検出します。

構文

```
check publication pub_name
with primary at data_server.database
```

パラメータ

- **pub_name –** チェックするパブリケーションの名前です。
- **with primary at data_server.database –** プライマリデータのロケーションを指定します。プライマリデータベースがウォームスタンバイアプリケーションの一部である場合、*data_server.database* は論理データサーバと論理データベースの名前になります。

例

- **例 1 –** パブリケーション *pubs2_pub* のステータスを確認します。プライマリデータベースは TOKYO_DS.*pubs2* です。

```
check publication pubs2_pub
 with primary at TOKYO_DS.pubs2
```

使用法

- **check publication** は、パブリケーションのステータスとそのパブリケーションに含まれるアーティクルの数を検出するときに使用します。
  パブリケーションの詳細については、『Replication Server 管理ガイド 第 1 巻』を参照してください。
- **check publication** は、レプリケートデータベースを管理する Replication Server、またはプライマリデータベースを管理する Replication Server で実行します。
- レプリケート Replication Server で **check publication** を実行した場合、パブリケーションは、現在のユーザ名とパスワードを使用して、プライマリ Replication Server でチェックされます。プライマリ Replication Server では、パブリケーションについて現在の情報を表示する場合と同じログイン名とパスワードが必要です。

- サブスクリプションのステータスを確認するには、**check subscription** を使用します。詳細については、「**check subscription**」コマンドを参照してください。

check publication によって返されるメッセージ

- プライマリ Replication Server またはレプリケート Replication Server で **check publication** を実行すると、次のいずれかのメッセージが返されます。

```
Publication pub_name for primary database
data_server.database is valid. The number of
articles in the publication is number_articles.
Publication pub_name for primary database
data_server.database is invalid. The number of
articles in the publication is number_articles.
```

- レプリケート Replication Server で **check publication** を実行すると、プライマリ Replication Server に接続できない場合に、次のメッセージが返されます。

```
Failed to get publication information from primary.
```

パーミッション

このコマンドは、すべてのユーザが実行できます。レプリケート Replication Server でこのコマンドを入力するユーザには、プライマリ Replication Server でも同じログイン名とパスワードが必要です。

参照：

- check subscription (233 ページ)
- create publication (341 ページ)
- validate publication (524 ページ)

check schema map

(レプリケートデータベースが SAP HANA データベースである場合のみ)
Replication Server で定義されたプライマリスキーママップとレプリケートスキーママップをすべて表示します。

構文

```
check schema map [[from primary_data_server.primary_database.
[from_schema|NULL]]|[to replicate_data_server.replicate_database.
[to_schema|NULL]]]
```

パラメータ

- ***primary_data_server.primary_database*** – プライマリデータサーバとデータベースの名前。

- ***from_schema*** – プライマリデータベースのスキーマ。
- ***replicate_data_server.replicate_database*** – レプリケートデータサーバとデータベースの名前。
- ***to_schema*** – レプリケートデータベースのスキーマ。
- ***NULL*** – *from_schema* および *to_schema* の代わりに NULL を使用すると、それぞれすべてのプライマリスキーマまたはすべてのレプリケートスキーマが指定されます。

例

- **例 1** – この例では、プライマリ Oracle データベースまたはプライマリ Adaptive Server データベースのスキーママッピングを表示します。

```
check schema map from PDS.PDB.SAPSR3
```

- **例 2** – この例では、プライマリ IBM DB2 UDB データベースのスキーママッピングを表示します。

```
check schema map from PDS.PDB.SAP<SID>
```

- **例 3** – この例では、レプリケートデータベースのスキーママッピングすべてを表示します。

```
check schema map to RDS.RDB.NULL
```

- **例 4** – この例では、レプリケート SAP HANA データベースのスキーママッピングを表示します。

```
check schema map to RDS.RDB.tableowner
```

使用法

- **check schema map** コマンドは、Replication Server で定義されているスキーママッピングをチェックするために使用します。
- 複写中にスキーママッピングが変更された場合、**check schema map** コマンドを使用して現在のマッピング関係をチェックできます。 後で要件に応じてマッピングを変更できます。

check subscription

複写定義またはパブリケーションのサブスクリプションのマテリアライゼーションステータスを検出します。

構文

```
check subscription sub_name
    for {table_rep_def | function_rep_def |
    [publication pub_name | database replication definition
```

```
db_repdef]
    with primary at data_server.database}
    with replicate at data_server.database
```

パラメータ

- **sub_name –** ステータスを調べるサブスクリプションの名前です。
- **for table_rep_def –** サブスクリプションの対象となるテーブル複写定義の名前を指定します。
- **for function_rep_def –** サブスクリプションの対象となるファンクション複写定義の名前を指定します。
- **for publication pub_name –** サブスクリプションの対象となるパブリケーションの名前を指定します。
- **database replication definition db_repdef –** サブスクリプションの対象となるデータベース複写定義の名前を指定します。
- **with primary at data_server.database –** プライマリデータのロケーションを指定します。プライマリデータベースがウォームスタンバイアプリケーションの一部である場合、*data_server.database* は論理データサーバと論理データベースの名前になります。この句を使用するのは、パブリケーションのサブスクリプションの場合だけです。
- **with replicate at data_server.database –** レプリケートデータのロケーションを指定します。レプリケートデータベースがウォームスタンバイアプリケーションの一部である場合、*data_server.database* は論理データサーバと論理データベースの名前になります。

例

- **例 1 –** 複写定義 *titles_rep* のサブスクリプション *titles_sub* のステータスを確認します。ここでのレプリケートデータベースは SYDNEY_DS.*pubs2* です。

```
check subscription titles_sub
 for titles_rep
 with replicate at SYDNEY_DS.pubs2
```

- **例 2 –** パブリケーション *pubs2_pub* のサブスクリプション *pubs2_sub* のステータスを確認します。ここでのプライマリデータベースは TOKYO_DS.*pubs2*、レプリケートデータベースは SYDNEY_DS.*pubs2* です。

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

使用法

- **check subscription** は、サブスクリプションのマテリアライゼーションまたはマテリアライゼーション解除の実行中、またはパブリケーションサブスクリプ

ションのリフレッシュ処理中に、サブスクリプションのステータスを検出するときに使用します。サブスクリプションは、テーブル複写定義、ファンクション複写定義、またはパブリケーションに対して作成できます。
サブスクリプションの詳細については、『Replication Server 管理ガイド 第 1 巻』を参照してください。

- **check subscription** は、レプリケートデータが格納されるデータベースを管理する Replication Server、またはプライマリデータベースを管理する Replication Server で実行します。

  check subscription の結果は、コマンドが実行される場所によって異なります。Replication Server がプライマリデータベースとレプリケートデータベースの両方を管理している場合、**check subscription** は 2 つのステータスメッセージを返します。
- パブリケーションのステータスを確認するには、**check publication** を使用します。詳細については、「**check publication**」コマンドを参照してください。
- 『Replication Server トラブルシューティングガイド』では、**check subscription** を使用したサブスクリプションのモニタの詳細について説明しています。

check subscription によって返されるメッセージ

- レプリケート Replication Server で **check subscription** を実行すると、次のいずれかのメッセージが返されます。
  ウォームスタンバイアプリケーションでは、出力が 2 行になることがあります。1 つはアクティブレプリケートデータベースのステータスを示し、もう 1 つはスタンバイレプリケートデータベースのステータスを示します。

INVALID	*sub_name* は存在しません
REMOVING	レプリケート側でシステムテーブルからサブスクリプション *sub_name* を削除しています。
DEMATERIALIZING	レプリケート側でサブスクリプション *sub_name* のマテリアライゼーションを解除しています (DEMATERIALIZING)。
VALID	レプリケート側でサブスクリプション *sub_name* は有効 (VALID) です。
VALIDATING	レプリケート側でサブスクリプション *sub_name* は検証中 (VALIDATING) です。
MATERIALIZED	レプリケート側でサブスクリプション *sub_name* はマテリアライゼーションが完了しています (MATERIALIZED)。

ACTIVE	レプリケート側でサブスクリプション *sub_name* はアクティブ (ACTIVE) です。
ACTIVATING	レプリケート側でサブスクリプション *sub_name* はアクティブ化中 (ACTIVATING) です。
ACTIVATING	レプリケートのスタンバイ側でサブスクリプション *sub_name* はアクティブ化中 (ACTIVATING) です。
QCOMPLETE and ACTIVE	レプリケート側でサブスクリプション *sub_name* はアクティブ (ACTIVE) で、マテリアライゼーションキューはすべて完了しました。
QCOMPLETE	サブスクリプション *sub_name* のマテリアライゼーションキューはすべて完了しました。
ACTIVE and not QCOMPLETE	レプリケート側でサブスクリプション *sub_name* はアクティブ (ACTIVE) ですが、マテリアライゼーションキューは完了していません。
DEFINED	レプリケート側でサブスクリプション *sub_name* が定義されました。

- 上記のメッセージに加えて、レプリケート Replication Server で **check subscription** を実行すると、次のいずれかのメッセージが返される場合があります。

ERROR	マテリアライゼーションまたはマテリアライゼーション解除の実行中に、サブスクリプション *sub_name* でリカバリできないエラーが発生しました。 詳細については、エラーログを参照してください。
PENDING	同じ複写定義またはレプリケーションデータベースに対して別のサブスクリプションの作成または削除の実行中です。 サブスクリプション *sub_name* は、前の要求が完了した後に実行されます。
RECOVERING	マテリアライゼーションまたはマテリアライゼーション解除の実行中に、サブスクリプション *sub_name* でリカバリできないエラーが発生しました。 これはサブスクリプションデーモン (dSub) によってリカバリされます。

- プライマリ Replication Server で **check subscription** を実行すると、次のいずれかのメッセージが返されます。

INVALID	*subscription_name* は存在しません。
DEMATERIALIZING	サブスクリプション *sub_name* はプライマリ側でマテリアライゼーションを解除しています (DEMATERIALIZING)。
VALID	サブスクリプション *sub_name* はプライマリ側で有効 (VALID) です。
ACTIVE	サブスクリプション *sub_name* はプライマリ側でアクティブ (ACTIVE) です。
ACTIVATING	サブスクリプション *sub_name* はプライマリ側でアクティブ化中 (ACTIVATING) です。
DEFINED	サブスクリプション *sub_name* がプライマリ側で定義されました。

- 初期ロードフェーズで **direct_load** オプションを使用して作成したサブスクリプションに対して **check subscription** を実行すると、次のようなステータスが返されます。

```
Subscription sub_name is ACTIVE at the replicate.
Subscription sub_name is ACTIVE at the primary.
Subscriptions sub_name progress: initial loading, xx% done, xxxxx
commands remaining.
```

- キャッチアップフェーズで **direct_load** オプションを使用して作成したサブスクリプションに対して **check subscription** を実行すると、次のようなステータスが返されます。

```
Subscription sub_name has been MATERIALIZED at the replicate.
Subscription sub_name is VALID at the primary.
Subscriptions sub_name progress: catchup, xx% done, xxxxx commands
remaining.
```

パーミッション

このコマンドは、すべてのユーザが実行できます。

参照：

- activate subscription (48 ページ)
- check publication (231 ページ)
- create subscription (376 ページ)
- define subscription (395 ページ)
- drop subscription (424 ページ)
- validate subscription (525 ページ)

configure connection

データベースコネクションの属性を変更します。

注意： **configure connection** の動作は、**alter connection** コマンドと同じです。

構文

構文については、「**alter connection**」を参照してください。

使用法

使用法については、「**alter connection**」を参照してください。

configure logical connection

論理コネクションの属性を変更します。

注意： **configure logical connection** は、**alter logical connection** と同じです。

構文

構文については、「**alter logical connection**」を参照してください。

使用法

使用法については、「**alter logical connection**」を参照してください。

configure replication server

ネットワークベースセキュリティをはじめとする Replication Server の特性を設定します。ERSSD を設定します。

構文

```
configure replication server {
    set repserver_param to 'value' |
    set route_param to 'value' |
    set database_param to 'value' |
    set logical_database_param to 'value' |
    set password_param to 'value' |
 maintenance_user_password_param to 'value' |
    set security_param to 'value' |
    set id_security_param to 'value' |
```

```
    set security_services [to] 'default'} |
    set user_authentication_source to 'value' |
    set parameter to 'parameter_value' |
```

パラメータ

- **repserver_param –** Replication Server に影響するパラメータです。「表 23 : Replication Server 設定パラメータ」および「表 30 : ERSSD 設定パラメータ」を参照してください。
- **value –** 設定パラメータの設定です。

表 23 : Replication Server 設定パラメータ

repserver_param	値
audit_enable	コマンド監査を有効にするには **audit_enable** を on にします。デフォルトは off です。
audit_dest	コマンド監査を有効にする場合は、コマンド監査ログのファイル名とパスを指定します。 デフォルトは log です。 Unix では、ファイルが存在しない場合、0600 のパーミッションを持つログファイルが Replication Server によって作成されます。0666 などの別のパーミッションを持つ独自のログファイルを UNIX で作成した場合は、そのパーミッションが Replication Server によって保持されます。

repserver_param	値
block_size to '*value***' with shutdown**	ステーブルキュー構造体で使用される連続メモリブロックのバイト数であるキューブロックサイズを指定します。 有効な値の範囲： 16KB、32KB、64KB、128KB、または 256KB デフォルト値は 16KB **注意：** コマンドを実行してブロックサイズを変更すると、Replication Server が停止します。Replication Server 15.6 より前のバージョンでブロックサイズを指定した後は、“**with shutdown**” 句を含める必要があります。バージョン 15.6 以降では、“**with shutdown**” 句はオプションです。キューブロックサイズの変更を有効にするために Replication Server を再起動する必要はありません。このパラメータは、**configure replication server** コマンドのみを使用して変更してください。そうしないとキューが破損します。 ライセンス： Advanced Services オプションで個別にライセンス供与されます。 手順については、『Replication Server 管理ガイド 第 2 巻』の「パフォーマンスチューニング」の「Advanced Services Option」を参照してください。
.block_sub_for_repdef_in_pub	アーティクルおよびパブリケーションで使用される複写定義へのサブスクリプションをブロックするか、それとも許可するかを指定する。 有効な値は off および on。 デフォルトは off。 このパラメータはプライマリ Replication Server で有効化する。サーバの再起動は必要ない。
cm_fadeout_time	Replication Server が RSSD とのコネクションをクローズするまでのアイドル時間 (秒単位)。この値を -1 にすると、コネクションは永久にクローズしなくなる。 デフォルト値は 300 秒 最小値：1 秒 最大値：2,147,483,648 秒
cm_max_connections	コネクションマネージャで使用できる送信コネクションの最大数。この値は 0 よりも大きくすること。 デフォルト値は 64

repserver_param	値
current_rssd_version	この RSSD でサポートされる Replication Server のバージョン。Replication Server は、起動時にこの値をチェックする。 **注意：** このパラメータの値は変更しないこと。この値は、アップグレード時またはダウングレード時に **rs_init** プログラムによって修正する。 デフォルト値は該当なし
deferred_queue_size	Open Server の遅延キューの最大サイズ。Open Server の制限を超える場合は、最大サイズを増やす。**deferred_queue_size** には 0 より大きい値を指定する。 **注意：** このパラメータの変更を有効にするには、Replication Server を再起動する必要がある。 デフォルト値は 2,048 (Linux および HPIA32 の場合)、1,024 (その他のプラットフォームの場合)
dist_direct_cache_read	ディストリビュータ (DIST) スレッドを有効にしてステーブルキュースレッド (SQT) キャッシュから SQL 文を直接読み取ります。これにより、SQT からの負荷、および SQT と DIST の間の依存性が減少し、SQT と DIST の両方の効率が向上します。 デフォルト値は off ライセンス：Advanced Services オプションで個別にライセンス供与されます。『Replication Server 管理ガイド 第 2 巻』の「パフォーマンスチューニング」の「Advanced Services Option」を参照してください。
ha_failover	Replication Server から Adaptive Server への新しいデータベース接続に対して、SAP フェールオーバサポートを有効または無効にする。 値は次のとおり。 • on - フェールオーバを有効にする。 • off - フェールオーバを無効にする。 デフォルト値は off
id_server	この Replication Server の ID サーバの名前。 **注意：** このパラメータの値は変更しないでください。**id_server** は、**rs_init** の実行時に設定されるので、Replication Server をアップグレードまたはダウングレードする場合のみ、**rs_init** プログラムによって修正します。 デフォルト値は該当なし

repserver_param	値
init_sqm_write_delay	キューに読み込みが行われている場合のステーブルキューマネージャに対する書き込み遅延。 デフォルト値は 100 ミリ秒
init_sqm_write_max_delay	キューに読み込みが行われていない場合のステーブルキューマネージャに対する書き込み遅延の最大値。 デフォルト値は 1,000 ミリ秒
mat_load_tran_size	直接ロードマテリアライゼーションの際に、初期ロードの最適なトランザクションまたはバッチサイズを指定する。 有効な値は 10 ～ 2147483646 デフォルト値: 10000
max_mat_load_threads	マテリアライズされるテーブルごとの最大ロードスレッド数を指定する。 有効な値は 1 ～ 80 デフォルト値: 5 Replication Server は、テーブルごとに 1 個のロードスレッドで直接ロードマテリアライゼーションを開始し、必要に応じてこのパラメータで指定された数までスレッドを追加生成する。 **max_mat_load_threads** は、ローカルの SAP Replication Server とデータベースの接続パラメータである。 このパラメータと **num_concurrent_subs** の値によって直接ロードマテリアライゼーションのリソース使用が制御される。 サブスクリプションが **num_of_select selects** を使用して作成され、**num_of_select** が 1 より大きい場合は、適用されるスレッドは選択されたスレッド間で均等に分けられる。 選択した各スレッドの適用スレッドの最大数は 20。
mem_reduce_malloc	大きな単位でメモリを割り付けできるようにすることで、メモリ割り付け回数を削減し、Replication Server のパフォーマンスを向上させます。 デフォルト値は off ライセンス：Advanced Services オプションで個別にライセンス供与されます。『Replication Server 管理ガイド 第 2 巻』の「パフォーマンスチューニング」の「Advanced Services Option」を参照してください。

repserver_param	値
mem_thr_dsi	DSI スレッドによる SQT キャッシュの入力を停止する合計メモリのパーセンテージを指定します。 デフォルト値は **memory_limit** 値の 80%。 範囲：1 - 100
mem_thr_exec	EXEC スレッドによる RepAgent からのコマンドの受信を停止する合計メモリのパーセンテージを指定します。 デフォルト値は **memory_limit** 値の 90%。 範囲：1 - 100
mem_thr_sqt	SQT スレッドでキャッシュからの最大トランザクションをフラッシュする合計メモリのパーセンテージを指定します。 デフォルト値は **memory_limit** 値の 85%。 範囲：1 - 100
mem_warning_thr1	この値を超えると最初の警告メッセージが生成される、合計メモリのスレッショルドパーセンテージを指定します。「**memory_limit**」を参照してください。 デフォルト値は **memory_limit** 値の 80%。 範囲：1 - 100
mem_warning_thr2	この値を超えると 2 番目の警告メッセージが生成される、合計メモリのスレッショルドパーセンテージを指定します。「**memory_limit**」を参照してください。 デフォルト値は **memory_limit** 値の 90%。 範囲：1 - 100
memory_control	メモリを大量に必要とするスレッドのメモリ制御動作を管理します。「**memory_limit**」を参照してください。 指定できる値は、次のとおり。 • on - メモリ制御を有効化する • off - メモリ制御を無効化する デフォルト値は on

repserver_param	値
memory_limit	Replication Server が使用できる合計メモリの最大値 (メガバイト単位)。 その他のいくつかの設定パラメータの値は、**memory_limit** によって示された、メモリプールから使用可能なメモリ量に直接関連する。これらの設定パラメータには、**md_sqm_write_request_limit**、**queue_dump_buffer_size**、**sqt_max_cache_size**、**sre_reserve**、**sts_cachesize** などがある。 デフォルト値は 2,047MB 32 ビット版 Replication Server の場合： • 最小値 - 0 • 最大値 - 2,047MB 64 ビット版 Replication Server の場合： • 最小値 - 0 • 最大値 - 2,147,483,647 **memory_control** の状態： • on - メモリ消費が **memory_limit** を超過しても Replication Server は停止しない • off - メモリ消費が **memory_limit** を超過すると Replication Server は自動的に停止する メモリ使用量を監視し、必要に応じて **memory_limit** を増加してください。 Replication Server は、**memory_limit** で指定された使用可能なメモリを超過したときに停止します。メモリ使用量を監視し、必要に応じて **memory_limit** を増加してください。 Replication Server でメモリを大量に必要とするスレッドは、次のとおりである。 • DSI • EXEC • SQT これらのスレッドは、メモリ使用量チェックを実行してから新しいデータを受信または処理することで、メモリ制御を実行します。メモリ制御時にメモリ使用量が多いことが判明すると、次の動作によりスレッド機能が調整されます。 • スレッドによる新しいデータのグループ化を停止し、既存データのクリーニングと処理を行います。または、 • 空きメモリが確保されるまで新しいデータを受信しないよう、スレッドをスリープモードにします。

repserver_param	値
	2,047MB より大きい値に設定されていた場合は、ダウングレードすると、オーバフローから保護するために 2,047MB にリセットされる。
minimum_rssd_version	この RSSD を使用できる Replication Server の最小バージョン。**current_rssd_version** が Replication Server のバージョンよりも大きい場合は、Replication Server の起動時にこの値がチェックされる。 **注意：** このパラメータの値は変更しないこと。この値は、アップグレード時またはダウングレード時に **rs_init** プログラムによって修正する。 デフォルト値は該当なし
nrm_thread	Replication Server が、ログ転送言語 (LTL: Log Transfer Language) コマンドを正規化してパックし、それと並行して RepAgent エグゼキュータスレッドによる解析を行うことができる、NRM スレッドを有効化します。NRM スレッドとの並列処理によって RepAgent エグゼキュータスレッドは応答時間を短縮します。NRM スレッドは RepAgent エグゼキュータスレッドから分離したスレッドです。 **configure replication server** コマンドを使用して **nrm_thread** を on に設定してから、**exec_nrm_request_limit** を使用します。 デフォルトは off。 ライセンス: Advanced Services オプションで個別にライセンス供与される。『Replication Server 管理ガイド 第 2 巻』の「パフォーマンスチューニング」の「Advanced Services Option」を参照。
num_client_connections	受信クライアントコネクション (クライアントからサーバへのコネクション) の最大許容数。Open Server の制限を超える場合は、最大数を増やす。30 以上の値を指定する。 デフォルト値は 30
num_concurrent_subs	同時に発生するサブスクリプションのマテリアライゼーション要求、またはマテリアライゼーション解除要求の最大許容数 (この制限は、アトミックマテリアライゼーションとノンアトミックマテリアライゼーションにだけ適用され、バルクマテリアライゼーションには適用されない)。他の要求が満たされた後には、最大数を超える要求も満たされる。最小値は 1。 デフォルト値は 10

repserver_param	値
num_msg_queues	Open Server メッセージキューの最大許容数。Open Server の制限を超える場合は、最大数を増やす。**num_threads** の設定値よりも大きい値を指定する。 デフォルト値は 300
num_msgs	Open Server メッセージキューのメッセージの最大許容数。Open Server の制限を超える場合は、最大数を増やす。 デフォルト値は 91,136
num_mutexes	Open Server ミューテックスの最大許容数。Open Server の制限を超える場合は、最大数を増やす。**num_threads** の設定値よりも大きい値を指定する。 デフォルト値: 1024
num_stable_queues	ステーブルキューの最大許容数 (HP9000 の場合のみ)。各ステーブルキューは、32,768 バイトの共有メモリを使用する。ステーブルキューの最小許容数は 32。 各スタンバイデータベースコネクションは、さらに 16,384 バイトの追加共有メモリを使用する。スタンバイデータベースコネクションは、2 つで 1 つの追加ステーブルキューと見なされる。 デフォルト値は 32
num_threads	Open Server スレッドの最大許容数。Open Server の制限を超える場合は、最大数を増やす。20 以上の値を指定する。 デフォルト値は 150
oserver	現在の Replication Server の名前。 **注意：** このパラメータの値は変更しないこと。現在の Replication Server 名は、**rs_init** を使用して Replication Server をインストールしたときに指定したものである。 デフォルト値は該当なし
password_encryption	このパラメータは今後廃止される。**create user** コマンドまたは **alter user** コマンドを使用して、パスワードのセキュリティを実装する。すべての Replication Server ユーザに対してサーバレベルでパスワードのパラメータを設定するには、**configure replication server** を使用する。

repserver_param	値
prev_min_rssd_version	**rs_init** インストールアップグレードに従って、この値には **minimum_rssd_version** の前の値が入っている。 **注意：** このパラメータの値は変更しないこと。アップグレードまたはダウングレードを行うときに、**rs_init** がこの値を修正する。 デフォルト値は該当なし
prev_rssd_version	**rs_init** インストールアップグレードに従って、この値には **current_rssd_version** の前の値が入っている。 **注意：** このパラメータの値は変更しないこと。アップグレードまたはダウングレードを行うときに、**rs_init** がこの値を修正する。 デフォルト値は該当なし
queue_dump_buffer_size	**sysadmin dump_queue** コマンドによって使用されるコマンドの最大長(バイト単位)。指定した長さよりも長いコマンドはトランケートされる。値の範囲は、1,000 ～ 32,768。 デフォルト値は 32,768 バイト
quotes_in_identifiers	RepAgent から送信されたテーブルまたはカラム識別子に引用符が使用されている場合の Replication Server の動作を指定する。 有効な値は次のとおりである。 • block - コマンドが実行されたときにテーブル名およびカラム名の引用符を検索する。 • ignore - コマンドが実行されたときにテーブル名およびカラム名の引用符を無視する。 デフォルト: block **quotes_in_identifiers** が block の場合、識別子に引用符が含まれるコマンドが RepAgent によって送信されると、エラーメッセージが Replication Server ログファイルに書き込まれ、RepAgent はサスペンドされる。 このパラメータはすぐに有効になり、再起動は必要ない。
rec_daemon_sleep_time	リカバリデーモンのスリープ時間を指定する。このデーモンは、ウォームスタンバイアプリケーションや他のいくつかのオペレーションで “strict” セーブインターバルメッセージを処理する。 デフォルト値は 2 分
rssd_error_class	RSSD のエラークラス。 デフォルト値は rs_sqlserver_error_class

repserver_param	値
send_enc_password	RSSD との最初のコネクションを除く、すべての Replication Server クライアントコネクションで暗号化パスワードが使用されるようにする。値は “on” または “off”。 詳細については、『Replication Server 管理ガイド 第 1 巻』を参照してください。 デフォルト値は off
send_timestamp_to_standby	複写定義が存在しない場合に、`timestamp` カラムをレプリケートデータベースに送信するかどうかを指定する。値は on または off。 詳細については、『Replication Server 管理ガイド 第 1 巻』を参照してください。 デフォルト値は off
smp_enable	対称型マルチプロセッシング (SMP) を有効にする。Replication Server スレッドが、Replication Server によって内部的にスケジュールされるか、オペレーティングシステムによって外部的にスケジュールされるかを指定する。Replication Server スレッドが内部的にスケジュールされる場合、使用可能なマシンプロセッサの数にかかわらず、Replication Server で使用できるプロセッサは 1 つに制限される。値は “on” または “off”。 デフォルト値は on
sqm_async_seg_delete	セグメントを削除する専用デーモンを有効にし、インバウンドとアウトバウンドのキュー処理のパフォーマンスを向上させるには、**sqm_async_seg_delete** を on に設定します。 デフォルト値は on このパラメータ設定の変更を有効にするには、Replication Server を再起動する必要があります。 **sqm_async_seg_delete** が on の場合、Replication Server に大規模なパーティションが必要になる可能性があります。**alter partition** を使用してパーティションを拡大してください。 • 『Replication Server 設定ガイド』の「Replication Server のインストールと設定の準備」の「複写システムのプラン作成」の「各 Replication Server の最初のディスクパーティション」 • 『Replication Server 管理ガイド 第 1 巻』の「Replication Server の技術的概要」の「Replication Server でのトランザクション処理」の「ステーブルキュー」の「ステーブルキューのパーティション」 を参照してください。

repserver_param	値
sqm_cache_enable	サーバワイドなステーブルキューのキャッシュを設定する。値は on または off。 デフォルト値は on
sqm_cache_size	サーバワイドなステーブルキューのキャッシュサイズを設定する。ページ数を一重引用符または二重引用符で囲む。 範囲は 1 ～ 4096。 デフォルト値は 16
sqm_page_size	サーバワイドなステーブルキューのページサイズをページあたりのブロック単位で設定する。ページサイズを一重引用符または二重引用符で囲む。たとえば、ページサイズを 4 に設定すると、64K チャンクでステーブルキューに書き込むように Replication Server に指示される。 ページサイズを設定すると、Replication Server の I/O サイズも設定される。値の範囲は、1 ～ 64。 デフォルト値は 4
sqm_recover_segs	Replication Server が RSSD をリカバリ QID 情報で更新する前に割り付けるステーブルキューセグメント数を指定する。 『Replication Server 管理ガイド 第 2 巻』の「パフォーマンスチューニング」の「割り付けられるステーブルキューセグメント数の指定」を参照してください。 パフォーマンスを向上させるには **sqm_recover_segs** の値を大きくすることを推奨する。 デフォルト値は 1 最小値：1 最大値：2,147,483,648
sqm_warning_thr1	最初の警告を生成するために使用されるパーティションセグメント (ステーブルキューの領域) の割合 (パーセント)。値の範囲は、1 ～ 100。 デフォルト値は 75
sqm_warning_thr2	2 番目の警告を生成するために使用されるパーティションセグメントの割合 (パーセント)。値の範囲は、1 ～ 100。 デフォルト値は 90
sqm_warning_thr_ind	単一のステーブルキューが警告を生成するために使用する合計パーティション領域の割合 (パーセント)。値の範囲は、51 ～ 100。 デフォルト値は 70

repserver_param	値
sqm_write_flush	書き込みオペレーションが完了する前に、メモリバッファに書き込まれたデータをディスクにフラッシュするかどうかを指定する。以下の値のいずれかです。 • on - メモリバッファに書き込まれたデータをディスクにフラッシュする。 • off - メモリバッファに書き込まれたデータをディスクにフラッシュしない。 • dio - ダイレクト I/O を有効にし、Replication Server がファイルシステムをバッファリングしなくてもディスクに対して読み書きできるようにする。Solaris (SPARC) と Linux でのみ使用できる。 デフォルト値: オン
sqt_init_read_delay	SQT スレッドが SQM 読み込みを待機し、コマンドキュー内の新しい命令のチェックを開始するまでスリープする時間の長さ。それぞれのスリープ時間の終了時にコマンドキューが空の場合、SQT はスリープ時間を **sqt_max_read_delay** に設定されている値を超える直前まで、繰り返し倍の値を設定していく。 デフォルト値: 1 ミリ秒 最小値: 0 ミリ秒 最大値: 86,400,000 ミリ秒 (24 時間)
sqt_max_cache_size	ステーブルキュートランザクション (SQT) インタフェースキャッシュメモリの最大値 (バイト単位)。 32 ビット版 Replication Server の場合： • デフォルト - 1,048,576 • 最小値 - 0 • 最大値 - 2,147,483,647 64 ビット版 Replication Server の場合： • デフォルト - 20,971,520 • 最小値 - 0 • 最大値 - 2,251,799,813,685,247 2,147,483,647 バイトより大きい値に設定されていた場合は、ダウングレードすると、オーバフローから保護するために 2,147,483,647 バイトにリセットされる。

repserver_param	値
sqt_max_read_delay	SQT スレッドがコマンドキューに新しい命令があるかどうかをチェックするまで SQM 読み込みを待機している間、SQT スレッドがスリープする最長時間。 デフォルト値は 1 ミリ秒 最小値：0 ミリ秒 最大値：86,400,000 ミリ秒 (24 時間)
sre_reserve	新しいサブスクリプションに割り付ける追加領域の合計。たとえば、100 (100%) は現在の領域の 2 倍を意味する。値の範囲は、0 ～ 500。 複写定義の **sre_reserve** パラメータを更新するには、*rs_config* システムテーブルに直接挿入するか、このテーブルを直接更新する。 デフォルト値は 0
stats_reset_rssd	RSSD で以前のサンプリングデータをトランケートするか、新しい情報で上書きするかを示す。 値：on - 古いサンプリングデータを新しい情報で上書きする。 off - 前のサンプリングデータを維持する。 デフォルト値は on
stats_sampling	サンプリングカウンタを有効にする。 デフォルト値は off
stats_show_zero_counters	**admin stats** コマンドで、指定したサンプリング時間の監視数が 0 のカウンタをレポートするかどうかを指定する。 値は、次のとおり。 • on - 監視数が 0 のカウンタをレポートする。 • off - 監視数が 0 のカウンタをレポートしない。 デフォルト値は off
sts_cachesize	キャッシュされた RSSD システムテーブルごとにキャッシュされるローの合計数。この値をアクティブな複写定義の数まで増やすと、Replication Server は負荷の高いテーブル検索を実行しなくなる。 デフォルト値は 1000

repserver_param	値
sts_full_cache_*RSSD_system_table_name*	完全にキャッシュする RSSD システムテーブルを指定する。完全にキャッシュされたテーブルでは、簡単な **select** 文に対して RSSD へのアクセスは不要になる。 **sts_full_cache_** を入力し、スペースを空けずにテーブル名を指定する。 デフォルトで **sts_full_cache** が on に設定され、Replication Server によって完全にキャッシュされるテーブル: • `rs_clsfunctions` • `rs_repobjs` • `rs_users` 完全にキャッシュできるすべての RSSD テーブルのリストについては、『Replication Server 管理ガイド 第 2 巻』の「パフォーマンスチューニング」の「チューニングパラメータの使用についての注意事項」の「システムテーブルのキャッシュ」の「キャッシュできるシステムテーブル」を参照してください。
sub_daemon_sleep_time	サブスクリプションデーモンが、ウェイクアップしてサブスクリプションをリカバリするまでにスリープする時間 (秒単位)。値の範囲は、1 ～ 31,536,000。 デフォルト値は 120 秒
unicode_format	`U&"` フォーマットの Unicode データの送信をサポートする。これにより、Replication Server における UTF-8 文字セットの制限がなくなる。 **unicode_format** を次の値のいずれかに設定する。 • `string` - Unicode 文字を文字列フォーマットに変換する。 たとえば、文字列 "hello" は "hello" として送信される。 この場合、Replication Server では UTF-8 が必須である。 • `ase` - Unicode 文字を `U&''` フォーマットで送信する。 たとえば、文字列 "hello" は "U&'¥0068¥0065¥006c¥006c¥006f' " として送信される。 2 バイト Unicode 値は、Adaptive Server Enterprise が要求するネットワーク順序で送信される。 この場合、Replication Server では UTF-8 以外の文字セットを使用できる。 デフォルト値は文字列

repserver_param	値
varbinary_strip_trailing_zeros	**varbinary_strip_trailing_zeros** を on に設定し、後続ゼロを `varbinary` 値から削除します。off に設定すると、Replication Server は `varbinary` 値の後続ゼロを複写します。 パラメータの変更を有効にするために、Replication Server を再起動する必要もコネクションをサスペンドしてから再開する必要もありません。 デフォルト値は on
varchar_truncation	プライマリ Replication Server またはレプリケート Replication Server で、*varchar* カラムのトランケーションを有効にする。両方の Replication Server で文字セット変換が行われる場合は、レプリケート Replication Server で **varchar_truncation** を設定する。 デフォルト値は off

- **route_param –** ルートに影響します。ルートパラメータのリストと説明については、「表 20 : ルートに影響を与える設定パラメータ」を参照してください。**configure replication server** は、送信元 Replication Server を始点とするすべてのルートのパラメータ値を設定します。
- **database_param –** コネクションに影響します。コネクションパラメータのリストと説明については、「表 18 : データベースコネクションに影響を与えるパラメータ」を参照してください。**configure replication server** は、送信元 Replication Server を始点とするすべてのコネクションのパラメータ値を設定します。
- **logical_database_param –** 論理コネクションに影響します。パラメータのリストと説明については、「表 20 : ルートに影響を与える設定パラメータ」を参照してください。**configure replication server** は、送信元 Replication Server を始点とするすべての論理コネクションのパラメータ値を設定します。
- **password_param –** パスワードセキュリティパラメータに影響を与えます。 パラメータのリストと説明については、「表 24 : パスワードパラメータ」を参照してください。

表 24 : パスワードパラメータ

password_parameter	**説明と*値***
min_password_len	最小文字数。 • 0 - 最小長なし。 • 範囲 - 6 ～ 16 (デフォルトは 6)。

password_parameter	説明と*値*
max_password_len	最大文字数。**max_password_len** は常に **min_password_len** より大きい値に設定します。 範囲 - 13 ～ 30 (デフォルトは 30)。
password_lowercase_required	小文字が必須であるかどうか。 • True - 必須。 • False - 不要 (デフォルト)。
password_uppercase_required	大文字が必須であるかどうか。 • True - 必須。 • False - 不要 (デフォルト)。
password_numeric_required	数値が必須であるかどうか。 • True - 必須。 • False - 不要 (デフォルト)。
password_special_required	特殊文字が必須であるかどうか。 • True - 必須。 • False - 不要 (デフォルト)。

simple_passwords_allowed	このオプション (または "simple_passwords_allowed") を false に設定した場合、Replication Server ではパスワードにユーザ名またはユーザパスワード辞書の値を含めることを許可しません。
	• True - 許可 (デフォルト)。
	• False - 許可しない。
	パスワード辞書は RSSD の `rs_dictionary` システムテーブルに作成できます。テーブルにはデフォルト値は格納されません。独自のスクリプトを作成して値をテーブルに挿入する必要があります。例：
	`insert into rs_dictionary (words) values ("abcd");`
	`insert into rs_dictionary (words) values ("1234");`

password_parameter	説明と*値*
disallowed_prev_passwords	ユーザが自分のパスワードを変更するときに再使用できない以前のパスワードの数。 • 0 - 許可される以前のパスワード。 • 範囲 - 0 ～ 32,767 (デフォルトは 0)。 管理者がパスワードをリセットする場合、パラメータ値はユーザパスワードに適用されません。
password_expiration	パスワードの有効期限が切れてから経過した日数。 • 0 - パスワードの有効期限は切れません (デフォルト)。 • 範囲 - 0 ～ 32,767。 **password_expiration** は、**alter user** および **create user** とともに使用できます。 パスワードの有効期限が切れると、Replication Server はアカウントをロックし、パスワードの有効期限が切れたことをユーザに通知します。ユーザはこのパスワードをリセットしないと、管理者がパスワードをリセットしないかぎり、いったん接続を解除された以降はログインできなくなります。新しいパスワードは、パスワード要件をすべて満たす必要があります。 **rs_init** が **connect source** パーミッションまたは ID ユーザで作成するユーザのパスワードには、有効期限はありません。そのようなパスワードは、Replication Server でユーザ全員に対して設定された **password_expiration** のどのような設定でもオーバーライドします。データベース、他の Replication Server、および Replication Agent では、**connect source** パーミッションがあるユーザ ID を使用します。 管理者は、レプリケーションエージェントまたは RSI 向けに作成されたユーザのパスワードの有効期限が切れないようにパスワードを設定します。
initial_password_expiration	最初のパスワードの有効期限が切れてから経過した日数。 • 0 - 最初のパスワードの有効期限は切れません。 • 範囲 - 0 ～ 32,767 (デフォルトは 0)。 ユーザの最初のパスワードは、ユーザを作成するか、ユーザのパスワードをリセットするときに管理者が設定するパスワードです。

password_parameter	**説明と*値***
max_failed_logins	アカウントをロックする前に Replication Server が許可するログイン要求の最大失敗回数。 • 0 - アカウントは決してロックされません。 • 範囲 - 0 ～ 32,767 (デフォルトは 0)。 Replication Server は、**password_lock_interval** に設定されている時間間隔に従ってアカウントをロックします。
password_lock_interval	ユーザが **max_failed_logins** に設定されているログイン最大試行回数に達した場合にアカウントがロックされる分数。 • 0 - アカウントは管理者がパスワードをリセットするまでロックされます。 • 範囲 - 0 ～ 32,767 (デフォルトは 0)。
unused_login_expiration	ユーザアカウントが期限切れになるまでの未使用の日数。 • 0 - 未使用のアカウントは期限切れになりません。 • 範囲 - 0 ～ 32,767 (デフォルト)。 Replication Server は、**unused_login_expiration** より長く未使用になっているアカウントをロックする。 管理者はパスワードをリセットすることでこのアカウントを再度アクティブにできる。

- ***maintenance_user_password_param*** – メンテナンスユーザのパスワードセキュリティに影響します。 次を参照してください。 表 25 : メンテナンスユーザのセキュリティに影響するパラメータ (257 ページ)

表 25 : メンテナンスユーザのセキュリティに影響するパラメータ

maintenance_user_password_param	*値*
hide_maintuser_pwd	メンテナンスユーザのパスワード保護を設定して、メンテナンスユーザのアクセスを Replication Server のみに制限する。 **hide_maintuser_pwd** は次の場合に on に設定する。 • レプリケート Adaptive Server データベースへの既存の接続のメンテナンスユーザに対して新しいパスワードを定期的に生成する。 • 新しく作成するデータベース接続のレプリケート Adaptive Server データベースでパスワードを変更し、その後定期的に再作成する。 • メンテナンスユーザに再入力したすべてのパスワードを変更して暗号化する。 デフォルトは off。
maintuser_pwd_expiration	メンテナンスユーザのパスワード有効期間を設定する。 パスワードの有効期限が切れると、Replication Server は自動的にパスワードを変更する。 **maintuser_pwd_expiration** にゼロ以外の値を設定する前に、**hide_maintuser_pwd** を on に設定して、メンテナンスユーザのパスワード保護を有効化しておく必要がある。 範囲 - 0 〜 32,767 日 デフォルト - メンテナンスユーザのパスワード有効期間のデフォルト値は、Replication Server の **password_expiration** オプションに設定されている値。 **password_expiration** のデフォルト値は 0 日。つまり、パスワードに有効期限がない。

- **security_param –** ネットワークベースセキュリティに影響します。「表 26 : ネットワークベースセキュリティに影響を与えるパラメータ」を参照してください。

表 26 : ネットワークベースセキュリティに影響を与えるパラメータ

security_param	*値*
msg_confidentiality	Replication Server が暗号化データを送受信するかどうかを示す。“required” に設定すると、送信データが暗号化される。“not required” に設定すると、Replication Server は、送信されてくる暗号化データも非暗号化データも受け入れる。 デフォルト値は not_required

security_param	*値*
msg_integrity	データの改ざんに対するチェックを行うかどうかを示す。 デフォルト値は not_required
msg_origin_check	データの送信元を確認するかどうかを示す。 デフォルト値は not_required
msg_replay_detection	データが傍受および再送されていないことを確認するために、データをチェックするかどうかを示す。 デフォルト値は not_required
msg_sequence_check	データが送信順に受信されていることを確認するために、データをチェックするかどうかを示す。 デフォルト値は not_required
mutual_auth	コネクションが確立される前に、リモートサーバが身元を証明する必要があるかどうかを示す。 デフォルト値は not_required
security_mechanism	この経路で使用できるサードパーティのセキュリティメカニズムの名前。 デフォルト値は `libtcl.cfg` の SECURITY セクションで最初にリストされているメカニズム **注意：** このパラメータは、ExpressConnect for HANA データベースなどの ASE 以外および IQ 以外のコネクタには適用されません。 これらのコネクタの詳細については、『Replication Server 異機種間複写ガイド』を参照してください。
send_enc_password	RSSD との最初のコネクションを除く、すべての Replication Server クライアントコネクションで暗号化パスワードが使用されるようにする。値は “on” または “off”。 デフォルト値は off
unified_login	Replication Server がリモートデータサーバにログインし、受信ログインを受け入れる方法を示す。 値は、次のとおり。 • “required” - リモートサーバには、必ずクレデンシャルを使用してログインする。 • “not_required” - リモートサーバには、必ずパスワードを使用してログインする。 デフォルト値は not_required

security_param	*値*
use_security_services	Replication Server に、セキュリティサービスを使用するかどうかを指示する。**use_security_services** が “off” の場合、セキュリティ機能は無効となる。 **注意：** このパラメータは **configure replication server** によってのみ設定できます。このパラメータは、ExpressConnect for HANA データベースなどの ASE 以外および IQ 以外のコネクタには適用されません。これらのコネクタの詳細については、『Replication Server 異機種間複写ガイド』を参照してください。
use_ssl	Replication Server でセッションベースの SSL セキュリティが有効になっているかどうかを示す。 値は、次のとおり。 • “on” - Replication Server で SSL が有効になっている。 • “off” - Replication Server で SSL が有効になっていない。 デフォルト値は off

- **id_security_param –** ID サーバのネットワークベースセキュリティに影響します。これらのパラメータのリストと説明については、「表 27 : ID サーバに接続するためのセキュリティパラメータ」を参照してください。

表 27 : ID サーバに接続するためのセキュリティパラメータ

security_param	*値*
id_msg_confidentiality	Replication Server が暗号化されたデータパケットを送受信するかどうかを示す。“required” に設定すると、送信データが暗号化される。“not required” に設定すると、Replication Server は、送信されてくる暗号化データも非暗号化データも受け入れる。 デフォルト値は not required
id_msg_integrity	データパケットの改ざんをチェックするかどうかを示す。 デフォルト値は not required
id_msg_origin_check	データパケットの送信元を確認するかどうかを示す。 デフォルト値は not required
id_msg_replay_detection	データパケットが傍受および再送されていないことを確認するために、データパケットをチェックするかどうかを示す。 デフォルト値は not required

security_param	*値*
id_ msg_sequence_ check	データパケットが送信順に受信されていることを確認するために、データパケットをチェックするかどうかを示す。 デフォルト値は not required
id_mutual_auth	Replication Server がコネクションを確立する前に、ID サーバに身元の証明を要求する。 デフォルト値は not required
id_security_mech	サポートされているセキュリティメカニズムの名前を指定する。 サポートされているセキュリティメカニズムは、`libtcl.cfg` ファイルの SECURITY セクションにリストされている。名前が指定されない場合、Replication Server はデフォルトのメカニズムを使用する。 デフォルト値はリストされている最初のメカニズム
id_unified_login	Replication Server が ID サーバに接続する方法を示す。値は、次のとおり。 • required - 常にクレデンシャルを使用して ID サーバにログインしようとする。 • not required - 常にパスワードを使用して ID サーバにログインしようとする。 **注意：** **unified_login** を “required” に設定すると、"“sa” ユーザだけがクレデンシャルなしで Replication Server にログインできます。セキュリティメカニズムに問題が発生した場合でも、“sa” ユーザはログインし、**unified_login** を無効にできます。 デフォルト値は not required

- **set security_services [to] 'default' –** Replication Server のグローバル設定と一致させるために、コネクションのすべてのネットワークベースセキュリティ機能をリセットします。**use_security_services** 機能は再設定しません。

  Replication Server が複数のセキュリティメカニズムをサポートしている場合は、**set security_services [to] 'default'** コマンドによって、セキュリティメカニズムもデフォルト (`libtcl.cfg` ファイルの SECURITY セクションにリストされている最初のメカニズム) に設定されます。
- **user_authentication_source –** ユーザ認証メカニズムを設定します。

表 28 : ユーザ認証パラメータ

パラメータ	説明
rs	LDAP ユーザ認証を無効化し、ログイン要求の認証に `rs_users` テーブルユーザのクレデンシャルを使用する。
any	ログイン要求の認証に LDAP サーバのユーザクレデンシャルを使用するよう Replication Server に指示する。 LDAP サーバにユーザアカウントが存在しない場合、または LDAP サーバが停止している場合は、Replication Server は `rs_users` ユーザクレデンシャルを使用してログイン要求を認証する。 このオプションは、`rs_users` システムテーブル認証から LDAP 認証へ移行するときに使用する。
ldap	ユーザアカウントの認証に LDAP サーバのみを使用するよう Replication Server に指示する。 ユーザは LDAP サーバ上に存在している必要がある。 このオプションを使用すると、LDAP サーバは既存のユーザが有効な LDAP ユーザアカウントを持たない場合に、Replication Server によって認証されないようにする。 **注意：** LDAP ユーザ認証に加え Kerberos 認証も有効な場合、Replication Server はログイン要求の認証に Kerberos を使用する。

例

- **例 1 –** 暗号化フォーマットでデータを送信するように Replication Server を設定します。

```
configure replication server
 set id_msg_confidentiality to 'required'
```

- **例 2 –** グローバル設定と一致するように、すべてのセキュリティ機能を設定します。

```
configure replication server
 set security_services to 'default'
```

- **例 3 –** 現在の Replication Server を始点とするすべてのルートに対して、**rsi_save_interval** パラメータを 2 分に変更します。

```
suspend route to each_dest_replication_server

 configure replication server
 set rsi_save_interval to '2'
```

```
resume route to each_dest_replication_server
```

- **例 4 –** キューブロックサイズを 64 に設定します。

  ```
  configure replication server
  set block_size to '64'
  ```

  (省略可能) ブロックサイズの設定で **with shutdown** 句を使用して、プライマリ Replication Server を停止します。

  ```
  configure replication server
  set block_size to '64' with shutdown
  ```

- **例 5 –** すべてのユーザのパスワードの最小の長さを 8 文字に設定します。

  ```
  configure replication server
  set min_password_len to '8'
  ```

- **例 6 –** すべてのユーザのパスワードの有効期間を 90 日に設定します。

  ```
  configure replication server
  set password_expiration to '90'
  ```

- **例 7 –** LDAP ユーザ認証を使用するように Replication Server を設定します。

  ```
  configure replication server
  set user_authentication_source to 'ldap'
  ```

- **例 8 –** `rs_users` と LDAP ユーザ認証の両方を使用するように Replication Server を設定します。

  ```
  configure replication server
  set user_authentication_source to 'any'
  ```

  このコマンドを実行する前に、ユーザアカウントが `rs_users` テーブルに存在している必要があります。

- **例 9 –** アーティクルおよびパブリケーションで使用される複写定義へのサブスクリプションをブロックするよう Replication Server を設定します。

  ```
  configure replication server
  set block_sub_for_repdef_in_pub to 'on'
  ```

使用法

- 各パラメータには、設定値と実効値の 2 種類の値があります。設定値は、Replication Server の再起動時に使用される値です。実行値は、Replication Server によって現在使用されている値です。Replication Server を起動した時点では、2 つの値は同じです。
- 設定値は、RSSD の *rs_config* システムテーブルに格納されます。
- **"set block_size to 'block_size' with shutdown"** Replication Server パラメータを使用してキューブロックサイズを設定すると、Replication Server は自動的に停止します。新しいブロックサイズは Replication Server を再起動した後に有効になり

ます。『Replication Server 管理ガイド 第 2 巻』の「パフォーマンスチューニング」の「キューのブロックサイズの増加」を参照してください。

- **varchar_truncation** を使用すると、プライマリ Replication Server またはレプリケート Replication Server で *varchar* カラムをトランケートできます。受信した *varchar* データが複写定義で指定されているカラム長を超えた場合、次のようになります。

表 29 : varchar_truncation

	プライマリ Replication Server で設定される varchar_truncation	レプリケート Replication Server で設定される varchar_truncation
varchar_truncation を “on” に設定	Replication Server は、受信データを複写定義で指定されている長さにトランケートする。	Replication Server は、受信データを複写定義で指定されている長さにトランケートする。
varchar_truncation を “off” に設定	RepAgent により Replication Server ログにメッセージが記録され、Replication Server は複写定義で指定されているカラム長を超えるローを無視する。	Replication Server は Replication Server ログにメッセージを出力し、DSI が停止する。

- SAP フェールオーバサポートを有効にするには、**ha_failover** を使用します。ASE サーバのフェールオーバが発生すると、Replication Server から ASE へのすべてのコネクションが失敗します。 Replication Server は、コネクションをリトライします。 **ha_failover** を on に設定すると、新しいコネクションが新しい ASE サーバにフェールオーバできるようになります。
- ERSSD 設定パラメータを使用して、バックアップタイム、ディレクトリロケーション、RepAgent 名を設定します。

表 30 : ERSSD 設定パラメータ

ERSSD 設定パラメータ	値	デフォルト
erssd_backup_start_time	バックアップ開始時刻。 指定形式は、12 時間制の「hh:mm AM」か「hh:mm PM」、または 24 時間制の「hh:mm」。	デフォルト値は 01:00 AM
erssd_backup_start_date	バックアップ開始日。 指定形式は「MM/DD/YYYY」。	デフォルト値は現在の日付

ERSSD 設定パラメータ	値	デフォルト
erssd_backup_interval	データベースとログのバックアップ間隔。 指定形式は「nn hours」、「nn minutes」、「nn seconds」のいずれか。	デフォルト値は 24 hours
erssd_backup_dir	バックアップファイルを格納するロケーション。 ディレクトリのフルパスを指定する。このパスを設定すると、バックアップがすぐに実行される。	デフォルト値はトランザクションログミラーと同じディレクトリ。初期値は **rs_init** で指定する。
erssd_ra	現在のサイトから別の Replication Server へのルートを作成するために、Replication Agent 名を設定する。このサーバ名は、interfaces 名に含まれている必要がある。	*erssd_name*_**ra**

Replication Server パラメータ

- Replication Server パラメータは、ローカルの Replication Server に影響するデフォルト値を指定します。
- Replication Server パラメータは、静的です。これらのパラメータを有効にするには、Replication Server を再起動します。

ルートパラメータ

- ルートパラメータは、送信元 Replication Server で開始されるすべてのルートのデフォルト値を指定します。
- **alter route** を使用して個々のルートの値を設定すると、**configure replication server** を使用して指定したデフォルト値を上書きできます。
- 現在の Replication Server で開始されるすべてのルートをサスペンドしてから、**configure replication server** コマンドを実行してください。パラメータを変更したら、すべてのルートをレジュームして変更を有効にする必要があります。

データベースパラメータ

- データベースパラメータは、送信元 Replication Server で開始されるすべてのコネクションのデフォルト値を指定します。
- **alter connection** を使用して個々のコネクションの値を設定すると、**configure replication server** を使用して指定したデフォルト値を上書きできます。
- 現在の Replication Server で開始されるすべてのコネクションをサスペンドしてから、**configure replication server** を実行してください。パラメータを変更した

ら、すべてのコネクションをレジュームして変更を有効にする必要があります。

論理データベースパラメータ

- 論理データベースパラメータは、送信元 Replication Server で開始される論理コネクションのデフォルト値を指定します。
- **configure logical connection** を使用して特定の論理コネクションの値を設定すると、**configure replication server** を使用して指定したデフォルト値を上書きできます。
- 論理データベースパラメータは、動的です。これらのパラメータはただちに有効になります。

ネットワークベースのセキュリティのパラメータ

- **use_security_services** と **use_ssl** を除き、**configure replication server** を使用して設定されたセキュリティパラメータは動的です。これらのパラメータはただちに有効になります。
- **use_security_services** と **use_ssl** は静的です。これらの値を変更した場合は、Replication Server を再起動して変更を有効にしてください。
- **configure replication server** を使用して設定したデフォルトのネットワークベースセキュリティのパラメータは、現在の Replication Server に関連するすべての送受信経路の値を指定します。
- **configure replication server** を使用して指定したデフォルトのセキュリティ設定は、**alter route** または **alter connection** を使用して個々の送信経路のセキュリティ値をリセットすることによって無効にできます。
- **unified_login** を “required” に設定すると、“sa” ユーザだけがクレデンシャルなしで Replication Server にログインできます。セキュリティメカニズムがダウンした場合でも、“sa” ユーザはパスワードを使用して Replication Server にログインし、**unified_login** を無効にできます。
- Replication Server は、複数のセキュリティメカニズムをサポートできます。サポートされている各メカニズムは、`libtcl.cfg` ファイルの SECURITY セクションにリストされています。
- ルートの両端では、同じセキュリティメカニズムとセキュリティ設定を使用する互換性のある SCL (Security Control Layer) ドライバを使用してください。 各サーバについて、セキュリティ機能の選択と設定を行うのは複写システム管理者の仕事です。 Replication Server は、リモートサーバとのコネクションを確立しようとする前に、そのサーバのセキュリティ機能の問い合わせは行いません。 経路の両端のセキュリティ機能に互換性がないと、ネットワークコネクションは失敗します。
- メッセージの暗号化は、深刻なパフォーマンスの低下をともなう、負荷の高い処理です。通常は、特定の経路に対してのみ **msg_confidentiality** を “required” に

設定してください。代わりに、**msg_integrity** などの負荷の低い機能を選択してセキュリティを確保します。

パーミッション

configure replication server には、“sa” パーミッションが必要です。

参照：

- admin security_property (70 ページ)
- admin security_setting (72 ページ)
- alter connection (134 ページ)
- alter route (213 ページ)
- configure connection (238 ページ)
- configure route (266 ページ)
- create connection (287 ページ)
- create route (368 ページ)
- set proxy (448 ページ)

configure route

現在の Replication Server からリモート Replication Server へのルートの属性を変更します。

注意： **configure route** は、**alter route** コマンドと同じです。

構文

構文については、「**alter route**」コマンドを参照してください。

使用法

使用法については、「**alter route**」コマンドを参照してください。

connect

Replication Server を、その RSSD、ID サーバ、リモート Replication Server、またはリモートデータサーバのゲートウェイにします。

構文

```
connect [to] [rssd | idserver | srv_name | ds_name.db_name]
```

パラメータ

- **rssd –** Replication Server を、その RSSD のゲートウェイにします。 設定ファイルの *RSSD_primary_user* エントリと *RSSD_primary_pw* エントリをゲートウェイが使用できるようにします。 **rssd** は **connect to** オプションのデフォルトです。
- **idserver –** Replication Server を、その ID サーバのゲートウェイにします (Replication Server 自体が ID サーバでない場合)。 設定ファイルの *ID_user* エントリと *ID_pw* エントリをゲートウェイが使用できるようにします。
- **srv_name –** ゲートウェイを接続するリモート Replication Server の名前です。 ゲートウェイは、RSI を使用してリモートサーバにログインするため、リモートサーバへの直接ルートが必要です。
- **ds_name.db_name –** ゲートウェイを接続するリモートデータサーバおよびデータベースの名前です。Replication Server ゲートウェイは、メンテナンスユーザを通じてリモートデータサーバにログインします。これにより、指定されたデータベースのメンテナンスユーザに許可されているタスクを実行できるようになる。ただし、接続先のデータサーバで定義された他のデータベースにはアクセスできない。

  Replication Server ゲートウェイは、Adaptive Server と、Enterprise Connect Data Access (ECDA) を必要としない SAP® IQ およびレプリケートサーバに直接接続できるようにします。 いずれかの ExpressConnect 製品を使用してこれらのサーバと通信している場合は、Replication Server ゲートウェイを使用してレプリケートサーバに接続することはできません。

例

- **例 1 –** Replication Server ost_replinuxvm_02 から RSSD ost_replinuxvm_01.emb へのゲートウェイコネクションを作成するため、**connection to** コマンドを発行します。

  ```
  isql -Usa -P -S ost_replinuxvm_02
  1> connect to
  2> go
  ```

  ```
  Gateway connection to 'ost_replinuxvm_01.emb' is created.
  ```

  show server コマンドにより、コネクションを確認します。

  ```
  1> show server
  2> go
  ```

  ```
  ost_replinuxvm_01.emb
  ```

- **例 2 –** Replication Server ost_replinuxvm_02 から Replication Server ost_replinuxvm_03 に接続します。

  ```
  isql -Usa -P -S ost_replinuxvm_02
  1> connect to ost_replinuxvm_03
  2> go
  ```

show server コマンドにより、コネクションを確認します。

```
1> show server
2> go
```

```
ost_replinuxvm_03
```

- **例 3 –** Adaptive Server ost_replinuxvm_01.pdb へのゲートウェイコネクションを作成します。

```
isql -Usa -P -S ost_replinuxvm_02
1> connect to ost_replinuxvm_01.pdb1
2> go
```

```
Gateway connection to 'ost_replinuxvm_01.pdb1' is
created.
```

```
1> select db_name()
2> go
```

```
------------------------------
pdb1
(1 row affected)
```

使用法

- **connect** コマンドを発行するには、Replication Server への初回ログインのための **sa** 役割が必要です。
- オプションを指定せずに **connect** コマンドを発行すると、RSSD へのゲートウェイコネクションが作成されます。
- ゲートウェイとして動作している場合、Replication Server は、RSSD プライマリユーザ名とパスワードを使用して RSSD に、ID サーバのユーザ名とパスワードを使用して ID サーバに、リモートサーバ ID (RSI) を使用してリモート Replication Server にログインします。 Replication Server 自体にアクセスするとき、この情報を複数回提供する必要はありません。
- ゲートウェイで作成されたカスケードコネクションは、コネクションスタックで保持され、最初の **connect** コマンドを発行した Replication Server がスタックの一番下に置かれます。
- Replication Server は、それ自体に直接接続できません。ただし、カスケードコネクションを使用することで、この問題に対処できます。
- Replication Server ゲートウェイを使用する場合、Replication Server は文字セットの変換を実行できないため、クライアントとサーバで同じロケールセットを使用してください。

パーミッション

Replication Server をゲートウェイに変えるには、“sa” パーミッションが必要です。

参照：

- disconnect (402 ページ)
- show connection (449 ページ)
- show server (450 ページ)

create alternate connection

代替プライマリコネクションまたは代替レプリケートコネクションか、代替アクティブコネクションまたは代替スタンバイコネクションを追加し、コネクションの設定パラメータを設定します。

構文

```
create alternate connection to data_server.database
named conn_server.conn_db
[set error class [to] error_class]
[set function string class [to] function_class]
[set username [to] user]
[set password [to] passwd]
[set database_param [to] 'value' [set database_param [to]
'value']...]
[set security_param [to] 'value' [set security_param [to]
'value']...]
[with {log transfer on | primary only}]
[as {active | standby} for conn_lds.conn_ldb]
```

パラメータ

- **data_server –** 代替プライマリコネクションまたは代替レプリケートコネクションを追加するデータベースを格納しているデータサーバです。
- **database –** 代替プライマリコネクションまたは代替レプリケートコネクションを追加するデータベースです。
- **conn_server.conn_database –** 代替プライマリコネクションまたは代替レプリケートコネクションの名前です。
  - 代替レプリケートコネクションでは、*conn_server* が *dataserver* と異なる場合は、interface ファイルに *conn_server* のエントリが必要です。
  - *conn_server* が *dataserver* と同じ場合は、*conn_db* が *database* と異なるようにしてください。
  - 各プライマリコネクション名は、複写システム内のすべてのプライマリコネクション名でユニークにしてください。各レプリケートコネクション名は、複写システム内のすべてのレプリケートコネクション名でユニークにしてください。
  - 代替プライマリコネクションを代替 Replication Agent のパスにバインドするには、*conn_server.conn_db* が Replication Agent から Replication Server までの

Replication Agent パスの名前と一致し、*conn_server* が *dataserver* と同じになるようにしてください。

- **error_class –** データベースのエラーを処理するエラークラスです。
- **function_class –** データベースのオペレーションに使用するファンクション文字列クラスです。
- **user –** 代替レプリケートコネクションのデータベースの Replication Server メンテナンスユーザのログイン名です。Replication Server は、複写データを管理するのにこのログイン名を使用します。ネットワークベースセキュリティを有効に設定していない場合には、ユーザ名を指定する必要があります。
- **passwd –** メンテナンスユーザのログイン名のパスワードです。ネットワークベースセキュリティメカニズムが有効になっていない場合は、パスワードを指定する必要があります。
- **database_param –** Replication Server からのデータベースコネクションに影響を与えるパラメータです。**alter connection** または **create connection** に使用するパラメータと同じパラメータを使用できます。
- **value –** オプションの値を持つ文字列です。
- **security_param –** ネットワークベースのセキュリティに影響を与えるパラメータです。**alter connection** または **create connection** に使用するパラメータと同じパラメータを使用できます。
- **log transfer on –** *dataserver.database* で指定したデータベースへの代替プライマリコネクションと代替レプリケートコネクションを、両方とも *conn_server.conn_db* で指定した名前で作成するように Replication Server に指示します。
- **primary only –** *conn_server.conn_db* で指定した名前で、プライマリデータベースへの代替プライマリコネクションのみを作成するように Replication Server に指示します。
- **as {active | standby} for conn_lds.conn_ldb –** *conn_lds.conn_ldb* という名前の代替論理コネクションを作成した場合は、ウォームスタンバイペアのアクティブデータベースまたはスタンバイデータベースへの代替コネクションを作成するように Replication Server に指示します。

例

- **例 1 –** SALES_DS データサーバの pdb データベースへの SALES_DS.pdb_conn2 という代替プライマリコネクションを作成します。

```
create alternate connection to SALES_DS.pdb
named SALES_DS.pdb_conn2
with primary only
go
```

- **例 2 –** FINANCE_DS データサーバの rdb レプリケートデータベースへの FINANCE_DS2.rdb_conn2 という代替レプリケートコネクションを作成します。

```
create alternate connection to FINANCE_DS.rdb
named FINANCE_DS2.rdb_conn2
go
```

- **例 3 –** IQSRVR SAP IQ データサーバの lqdb レプリケートデータベースへの IQSRVR.lqdb_conn2 という代替レプリケートコネクションを作成します。この場合、プライマリデータベースは Adaptive Server で、dbmaint2 は IQSRVR.lqdb_conn2 のメンテナンスユーザです。

```
create alternate connection to IQSRVR.iqdb
named IQSRVR.iqdb_conn2
using profile rs_ase_to_iq; standard
set username to dbmaint2
set password to dbmaint2pwd
go
```

  使用可能な SAP IQ マルチプレックス ノードへの代替コネクションも作成できます。 コネクション名がユニークであることを確認します。

  IQSRVR SAP IQ ノードの iqbd2 データベースへの iqdb2_conn1 代替コネクションを作成するには、次のコマンドを実行します。

```
create alternate connection to IQSRVR.iqbd2
named IQSRVR2.iqdb2_conn1
using profile rs_ase_to_iq; standard
set username to dbmaint3
set password to dbmaint3pwd
go
```

使用法

- **set function string class**、**set username**、**set password** は **alter connection** と **create connection** の既存の句で、代替コネクションを作成するときに使用できます。
  - これらの句を省略すると、代替レプリケートコネクションは、デフォルトのレプリケートコネクションを使用して設定した値を継承します。
  - デフォルトの接続を制御する (コントローラ) Replication Server とは異なる (現在の) Replication Server に代替コネクションを作成するときに、これらの句を省略すると、現在の Replication Server はエラーを返します。
  - 代替レプリケートコネクションは、同じ Replication Server がデフォルトと代替の両方のコネクションを制御する場合にのみ、デフォルト接続から値を継承できます。
  - 代替コネクションのメンテナンスユーザを設定しない場合は、デフォルト接続のメンテナンスユーザが継承されます。代替コネクションは、代替コネクションに指定した新しいメンテナンスユーザを使用します。

- **set***database_param* と **set***security_param* は **alter connection** と **create connection** の既存の句で、既存のコネクションのオプションパラメータを指定する場合に使用できます。
  - 代替コネクションに対して設定した値は、デフォルト接続から継承された値またはデフォルト値で上書きされます。
  - 代替コネクションは、同じ Replication Server が代替とデフォルトの両方のコネクションを制御する場合にのみ、デフォルト接続から値を継承できます。
- データベースを管理しているドメインと同じレプリケーションドメインに属する Replication Server で **create alternate connection** を実行します。
- ウォームスタンバイアプリケーションのための代替論理コネクションを作成するには、**create alternate logical connection** と **create alternate connection** を使用します。
- **alter connection** を使用すると、コネクションの属性を変更できます。メンテナンスユーザのパスワードが変更されている場合は、**alter connection** を使用して新しいパスワードを入力します。
- 『Replication Server 管理ガイド 第 2 巻』の「パフォーマンスチューニング」の「Multi-Path Replication」を参照してください。

パーミッション

create alternate connection には “sa” パーミッションが必要です。

参照：

- admin show_connections (77 ページ)
- alter connection (134 ページ)
- create connection (287 ページ)
- create connection using profile (294 ページ)
- drop connection (407 ページ)

create alternate logical connection

デフォルトの論理コネクションへの代替論理コネクションを追加します。
Replication Server は、論理コネクションを使用してウォームスタンバイアプリケーションを管理します。

構文

```
create alternate logical connection to logical_ds.logical_db
named conn_lds.conn_ldb
```

パラメータ

- **logical_ds –** デフォルト論理コネクションの論理データサーバの名前です。
- **logical_db –** デフォルト論理コネクションの論理データベースの名前です。
- **conn_lds.conn_ldb –** 代替論理コネクションの名前です。
  - *conn_lds* が *logical_ds* と同じ場合は、*conn_ldb* が *logical_db* と異なるようにしてください。
  - 各代替論理コネクション名 *conn_lds.conn_ldb* は、複写システム内のすべての代替論理コネクション名でユニークにしてください。

例

- **例 1 –** LDS 論理データサーバ内の `logicaldb` 論理データベースへの *lds.conn_logicaldb2* という代替論理コネクションを作成します。

```
create alternate logical connection to LDS.logicaldb
named lds.conn_logicaldb2
```

使用法

- 代替論理コネクションを作成する前に、**create logical connection** を使用してデフォルトの論理コネクションを作成する必要があります。『Replication Server 管理ガイド 第 2 巻』の「ウォームスタンバイアプリケーションの管理」の「ASE ウォームスタンバイデータベースの設定」で「作業 1: 論理コネクションの作成」を参照してください。
- 代替論理コネクションを作成して設定するには、『Replication Server 管理ガイド 第 2 巻』の「パフォーマンスチューニング」の「Multi-Path Replication」で「ウォームスタンバイ環境での複数のレプリケーションパス」を参照してください。
- さまざまな Replication Server を使用して、デフォルトの論理コネクションと代替論理コネクションを制御できます。アクティブとスタンバイの両方のデータベースが複数の Replication Agent の使用をサポートしている必要があります。
- 代替論理コネクションを作成すると、**create alternate connection** を使用して、アクティブデータベースとスタンバイデータベースへの代替コネクションを作成できます。
- 複写定義とサブスクリプションには、デフォルト論理コネクション名を使用します。

パーミッション

create alternate logical connection には、“sa” パーミッションが必要です。

参照：

- create alternate connection (269 ページ)
- create logical connection (338 ページ)

create applied function replication definition

複写するストアドプロシージャの適用ファンクション複写定義とユーザ定義ファンクションを作成します。適用ファンクションは、レプリケートデータベースでメンテナンスユーザによって適用されます。

構文

```
create applied function replication definition repdef_name
with primary at dataserver.database
[with all functions named 'func_name' |
[[with primary function named 'func_name']
[with replicate function named 'func_name']]]
([@param_name datatype [, @param_name datatype]…])
[searchable parameters (@param_name [, @param_name]…)]
[send standby {all | replication definition} parameters]
```

パラメータ

- **repdef_name –** 適用ファンクション複写定義の名前です。名前は識別子の規則に従う必要があります。
- **with primary at –** プライマリデータサーバとプライマリデータベースを指定します。
- **dataserver –** プライマリデータのあるデータサーバの名前です。プライマリデータベースがウォームスタンバイアプリケーションの一部である場合、*dataserver* は論理データサーバ名になります。
- **database –** プライマリデータのあるデータベースの名前です。プライマリデータベースがウォームスタンバイアプリケーションの一部である場合、*database* は論理データベース名になります。
- **with all functions named –** プライマリデータベースとレプリケートデータベースのストアドプロシージャ名を指定します。
- **'func_name' –** ファンクション名です。*func_name* は、最大 255 文字の文字列です。
- **with primary function named –** プライマリデータベースのストアドプロシージャ名を指定します。**with primary function named** を使用すると、複写定義名と異なるプライマリファンクションの名前を指定できます。プライマリファンクション名を指定しない場合、Replication Server はプライマリファンクションの名前として複写定義名を使用します。
- **with replicate function named –** レプリケートデータベースで実行するストアドプロシージャの名前を指定します。レプリケートファンクション名を指定しな

い場合、Replication Server はレプリケートファンクションの名前として複写定義名を使用します。

- **@param_name –** ファンクションからのパラメータ名です。1 つの句の中で同じパラメータ名を 2 回以上指定することはできません。パラメータとそのデータ型の指定は必須ではありませんが、パラメータを指定するかどうかに関係なく、この句はカッコで囲んでください。
- **datatype –** ファンクションのパラメータのデータ型です。データ型とその構文のリストについては、「データ型」を参照してください。Adaptive Server のストアドプロシージャとファンクション複写定義には、*text*、*unitext*、*rawobject*、*image* の各データ型のパラメータを含めることはできません。
- **searchable parameters – where** 句 (**define subscription**、**create subscription**、または **create article**) で使用できるパラメータのリストを指定します。**searchable parameters** 句を含める場合は、カッコで囲む必要があります。
- **send standby –** ウォームスタンバイアプリケーションで、スタンバイデータベースにファンクションのすべてのパラメータを送信するか (**send standby all parameters**)、複写定義で指定されたパラメータだけを送信するか (**send standby replication definition parameters**) を指定します。デフォルトは、**send standby all parameters** です。

例

- **例 1 – titles_frep** というファンクションに対して、同じ名前の適用ファンクション複写定義を作成します。プライマリデータは、*pubs2* データベース (*LDS* データサーバ) にあります。

```
create applied function replication definition titles_frep
with primary at LDS.pubs2
(@title_id varchar(6), @title varchar(80), @type char(12), @pub_id
char(4),
@price money, @advance money, @total_sales int)
searchable parameters (@title_id, @title)
```

- **例 2 – titles_frep** というファンクションに対して、同じ名前の適用ファンクション複写定義を作成します。レプリケートデータベースのストアドプロシージャに、**upd_titles** という名前を指定します。

```
create applied function replication definition titles_frep
with primary at LDS.pubs2
with replicate function named 'upd_titles'
(@title_id varchar(6), @title varchar(80), @type char(12), @pub_id
char(4),
@price money, @advance money, @total_sales int)
searchable parameters (@title_id, @title)
```

- **例 3 – titles_frep** という適用ファンクション複写定義 (このファンクションの名前は **upd_titles_prim**) を作成します。プライマリデータベースのストアドプロ

シージャには **upd_titles_prim**、レプリケートデータベースのストアドプロシージャには **upd_titles** という名前を指定します。

```
create applied function replication definition titles_frep
with primary at LDS.pubs2
with primary function named 'upd_titles_prim'
with replicate function named 'upd_titles'
(@title_id varchar(6), @title varchar(80), @type char(12), @pub_id
char(4),
@price money, @advance money, @total_sales int)
searchable parameters (@title_id, @title)
```

使用法

- **create applied function replication definition** は、複写するストアドプロシージャを記述するときに使用します。適用ファンクション複写定義と要求ファンクション複写定義の違いは、適用ファンクション複写定義を使用して複写されたファンクションは、レプリケートサイトでメンテナンスユーザが実行するのに対し、要求ファンクション複写定義を使用して複写されたファンクションは、プライマリサイトでプライマリファンクションを実行するユーザと同じユーザがレプリケートサイトで実行する点です。複写ストアドプロシージャの概要については、『Replication Server 管理ガイド 第 1 巻』を参照してください。
- プライマリファンクションの適用ファンクション複写定義を作成する場合は、そのファンクションに次の 2 つの条件を満たす既存のファンクション複写定義がまだないことを確認してください。
  - **create function replication definition** コマンドを使用して作成されている。
  - そのファンクション複写定義が、Replication Server 15.0.1 以前のバージョンでサブスクリプションのない要求ファンクション複写に使用されている。

  上記の両方の条件に該当する場合、既存の要求ファンクション複写定義は無効になります。Replication Server 15.0.1 以前の適用ファンクション複写定義の詳細については、『Replication Server 管理ガイド 第 2 巻』を参照してください。
- **create applied function replication definition** は、プライマリデータが格納されているデータベースを管理する Replication Server で実行します。
- **create applied function replication definition** を実行する前に、次のことを確認してください。
  - ファンクション複写定義の名前が、複写システム内でユニークであること。Replication Server は、**create applied function replication definition** の使用時に、この要件を常に適用できるわけではありません。
  - Replication Server とプライマリデータベース間にコネクションが存在すること。**create connection** を参照してください。

    rs_init を使用してコネクションを作成することもできます。詳細については、使用しているプラットフォームの『Replication Server インストールガイド』と『Replication Server 設定ガイド』を参照してください。

    - ファンクション複写定義に指定した名前、パラメータ、データ型が、関連するストアドプロシージャの名前、パラメータ、データ型と一致していること。ファンクション複写定義で指定したパラメータだけが複写されます。
- テーブル複写定義に対応する複写ストアドプロシージャとは異なり、ファンクション複写定義に対応するストアドプロシージャでは、テーブルを更新する必要はありません。そのため、複写データに関連しないトランザクションを複写できます。
  ストアドプロシージャの詳細については、「RSSD ストアドプロシージャ」を参照してください。複写ストアドプロシージャの 2 つのタイプの詳細については、「**sp_setrepproc**」を参照してください。
- Replication Server は、新しいファンクション複写定義を、条件を満たしているサイトへ複写システムを介して分配します。複写システムで通常の遅延時間が発生するため、条件を満たすすべてのサイトに変更内容がすぐに反映されるわけではありません。

ユーザ定義ファンクションとファンクション文字列

- 適用ファンクション複写定義を作成すると、Replication Server は、対応するユーザ定義ファンクションを自動的に作成します。同様に、**rs_sqlserver_function_class** では、Replication Server はユーザ定義ファンクションのデフォルトのファンクション文字列を自動的に作成します。
- **rs_sqlserver_function_class** とユーザ定義ファンクション文字列クラスのファンクション文字列は、**create function string** を使用してカスタマイズできます。
- ユーザ定義ファンクションを使用するユーザ定義の各基本ファンクション文字列クラスと、これらのクラスから継承した各派生クラスでは、**create function string** を使用してファンクション文字列を作成します。ファンクション文字列では、レプリケートデータサーバに適した言語を使用して、ストアドプロシージャまたは RPC を呼び出す必要があります。
- ファンクション文字列クラス、ファンクション文字列、ファンクションの概要については、『Replication Server 管理ガイド 第 2 巻』を参照してください。

with primary at 句

with primary at 句は、プライマリデータサーバとプライマリデータベースを指定するときに使用します。プライマリデータベースは、呼び出されるストアドプロシージャを格納するデータベースです。

with replicate function named 句

レプリケートデータベースで実行するストアドプロシージャの名前を指定するには、**with replicate function named** 句を使用します。ファンクション複写定義を作成または変更するときに **with replicate function named** を使用しない場合、ファンクションはファンクション複写定義と同じ名前のストアドプロシージャとして配信されます。ウォームスタンバイデータベースのストアドプロシージャは、アク

ティブデータベースのストアドプロシージャと同じ名前であるため、**with replicate function named** は無視されます。

往復複写では、データベースは別のデータベースにデータ変更要求を送信し、そのデータ変更を要求側のデータベースに複写できます。適用ファンクション複写定義と要求ファンクション複写定義の両方を使用して、往復複写を設定する方法の詳細については、『Replication Server 管理ガイド 第 1 巻』を参照してください。

HDS パラメータの適用ファンクション複写定義

パラメータ値のデータ型を変更するファンクション複写定義は作成できませんが、HDS データ型定義を使用して適用ファンクション複写定義のパラメータを宣言できます。宣言したパラメータは、クラスレベル変換の対象となります。

HDS の詳細については、『Replication Server 管理ガイド 第 1 巻』を参照してください。

ファンクション複写定義の変更

- パラメータまたはサーチャブルパラメータを既存の適用ファンクション複写定義に追加するには、**alter applied function replication definition** を使用します。ファンクションに別のレプリケート名を指定することもできます。
- ファンクション複写定義内のパラメータを削除したり名前を変更したりするには、ファンクション複写定義のすべてのサブスクリプションを削除します。サブスクリプションを削除したら、ファンクション複写定義を削除して再作成します。

ファンクション複写定義のサブスクリプションの作成

適用ファンクション複写定義のサブスクリプションを作成するには、**without materialization** 句を指定した **create subscription** を使用するか、**define subscription** と、バルクマテリアライゼーションを含むその他のコマンドを使用します。

ファンクション複写定義とテーブル複写定義

- 適用ファンクションを使用してストアドプロシージャを複写するときは、複写ストアドプロシージャの影響を受けるテーブルのテーブル複写定義とサブスクリプションを作成します。これにより、通常のトランザクションと、テーブルに影響するストアドプロシージャの実行が確実に複写されます。ただし、DML が複写済みとしてマーク付けされたストアドプロシージャ内にある場合、DML は複写されません。この場合、テーブルのサブスクリプションをすでに作成していても、ストアドプロシージャのサブスクリプションを作成します。
- 同じテーブルに対してファンクション複写定義とテーブル複写定義を使用する場合は、テーブル複写定義のサブスクリプションを使用してテーブルデータをマテリアライズします。**create subscription** を使用するときに、**without materialization** 句を指定して、ファンクション複写定義のサブスクリプションを作成します。

複数の複写定義の作成

- 1 つのプライマリファンクションに対して複数の適用ファンクション複写定義を作成し、それぞれ異なるレプリケートファンクションによってサブスクリプションを作成できるように各複写定義をカスタマイズできます。詳細については、『Replication Server 管理ガイド 第 1 巻』を参照してください。
- 1 つのプライマリファンクションに対して作成された各適用ファンクション複写定義では、同じ名前とデータ型の同じパラメータを使用する必要があります。
- 適用ファンクション複写定義で、複写定義とプライマリファンクションに異なる名前を指定した場合、この複写定義のサブスクリプションを作成できるのは、バージョン 15.1 以降の Replication Server だけです。
- 1 つのプライマリファンクションは、適用ファンクション複写定義または要求ファンクション複写定義を持つことができますが、この両方を持つことはできません。**create function replication definition** コマンドを使用して作成されたファンクション複写定義は、ファンクション複写定義が作成されたプライマリ Replication Server では適用ファンクションと見なされます。
- ウォームスタンバイデータベースでは、ストアドプロシージャはアクティブデータベースと同じ名前であるため、**with replicate function** 句は無視されます。適用ファンクション複写定義のいずれかが **send standby replication definition parameters** 句を指定して作成されている場合、ファンクション複写定義で指定されたパラメータがスタンバイデータベースに配信されます。それ以外の場合は、プライマリファンクションのすべてのパラメータが配信されます。
- MSA 環境では、**send standby** 句を指定して作成したプライマリファンクションのファンクション複写定義が存在しない場合、レプリケートデータベースに配信されるファンクションには、プライマリファンクションと同じ名前が使用され、プライマリファンクションのすべてのパラメータが含まれます。それ以外の場合は、レプリケートデータベースに配信されるファンクションには、ファンクション複写定義の **with replicate function named** 句で指定された名前が使用され、同じファンクション複写定義で指定されたパラメータが含まれます。

パーミッション

create applied function replication definition には、“create object” パーミッションが必要です。

参照：

- alter function string (188 ページ)
- alter applied function replication definition (129 ページ)
- alter request function replication definition (210 ページ)
- create connection (287 ページ)
- create function string (318 ページ)

- create request function replication definition (362 ページ)
- define subscription (395 ページ)
- drop function replication definition (411 ページ)
- sp_setrepproc (673 ページ)
- rs_send_repserver_cmd (735 ページ)

create article

テーブル複写定義またはファンクション複写定義のアーティクルを作成し、そのアーティクルを含めるパブリケーションを指定します。

構文

```
create article article_name
for pub_name
with primary at data_server.database
with replication definition {table_rep_def | function_rep_def}
[where {column_name | @param_name}
{< | > | >= | <= | = | &} value
[and {column_name | @param_name}
{< | > | >= | <= | = | &} value]...
[or where {column_name | @param_name}
{< | > | >= | <= | = | &} value
[and {column_name | @param_name}
{< | > | >= | <= | = | &} value]...]...]
```

パラメータ

- **article_name –** アーティクルの名前です。識別子の規則に従い、パブリケーション内でユニークな名前にしてください。
- **for pub_name –** アーティクルを含むパブリケーションの名前です。
- **with primary at data_server.database –** プライマリデータのロケーションを指定します。プライマリデータベースがウォームスタンバイアプリケーションの一部である場合、*data_server.database* は論理データサーバと論理データベースの名前になります。
- **with replication definition table_rep_def –** アーティクルの対象となるテーブル複写定義の名前を指定します。
- **with replication definition function_rep_def –** アーティクルの対象となるファンクション複写定義の名前を指定します。
- **各パラメータの意味は、次のとおりです。 –** このアーティクルを含むパブリケーションへのサブスクリプションを使用して複写されるカラム値またはパラメータ値の基準を設定します。**where** 句が何も指定されない場合は、すべてのローまたはパラメータが複写されます。

  where 句は 1 つ以上の単純比較で構成されます。単純比較では、関係演算子 **<**、**>**、**<=**、**>=**、**=**、または **&** のいずれかを使用して、サーチャブルカラムまたは

サーチャブルパラメータがリテラル値と比較されます (**&** 演算子は、*rs_address* データ型のカラムまたはパラメータでのみサポートされます)。また、キーワード **and** を使用して比較を結合できます。

where 句で使用されるカラム名またはパラメータ名は、テーブル複写定義の **searchable columns** リストまたはファンクション複写定義の **searchable parameters** リストにも含まれている必要があります。

1 つのアーティクル内に、複数の **where** 句をキーワード **or** で区切って指定できます。

アーティクル内の **where** 句の最大サイズは 255 文字です。

- **column_name –** テーブル複写定義を含むアーティクルの場合、プライマリテーブルからのカラム名です。
- **@param_name –** ファンクション複写定義を含むアーティクルの場合、複写ストアドプロシージャからのパラメータ名です。
- **value –** 指定したカラムまたはパラメータの値です。各種データ型の値の入力フォーマットについては、「データ型」を参照してください。

  式で使用されるカラム名またはパラメータ名は、複写定義の **searchable columns** リストまたは **searchable parameters** リストに含まれている必要があります。

例

- **例 1 –** 複写定義 *titles_rep* に基づいて、パブリケーション *pubs2_pub* の *titles_art* というアーティクルを作成します。

```
create article titles_art
 for publication pubs2_pub
 with primary at TOKYO_DS.pubs2
 with replication definition titles_rep
```

- **例 2 –** 前の例と同様に、パブリケーション *pubs2_pub* の *titles_art* というアーティクルを作成します。このコマンドの **where** 句では、*type* カラムに “popular_comp” を設定することで、一般に普及しているコンピュータマニュアルのローだけが複写されるようになっています。

```
create article titles_art
 for publication pubs2_pub
 with primary at TOKYO_DS.pubs2
 with replication definition titles_rep
 where type = 'popular_comp'
```

- **例 3 –** 前の例と同様に、パブリケーション *pubs2_pub* の *titles_art* というアーティクルを作成します。このコマンドには、一般に普及しているコンピュータマニュアルのローと伝統的な料理の本のローの両方を一緒に複写する、2 つの **where** 句が含まれています。

```
create article titles_art
 for publication pubs2_pub
 with primary at TOKYO_DS.pubs2
 with replication definition titles_rep
 where type = 'popular_comp'
 or where type = 'trad_cook'
```

使用法

- **create article** は、指定したパブリケーションを使用してデータを複写する複写定義を指定するときに使用します。オプションの **where** 句は、複写するデータを決定するのに役立ちます。
- プライマリデータが格納されているデータベースを管理する Replication Server で、**create article** コマンドを実行してください。
- **create article** を使用すると、アーティクルの対象となるパブリケーションが自動的に不確定化されます。パブリケーションを確定化するまでは、新しいサブスクリプションを作成できません。サブスクリプションをリフレッシュするまでは、新しいアーティクルのデータを複写できません。
- 複写定義、アーティクル、パブリケーションの処理の詳細については、『Replication Server 管理ガイド 第 1 巻』を参照してください。
  パブリケーションのサブスクリプションを作成する方法の詳細については、『Replication Server 管理ガイド 第 1 巻』の「サブスクリプションの管理」を参照してください。
- パブリケーションのサブスクリプションを作成またはリフレッシュした場合にのみ、Replication Server はパブリケーションとそのアーティクルについての情報をレプリケートサイトに分配します。

create article を使用するための条件

- **create article** を実行する前に、次の条件を確認してください。
  - 作成しているアーティクルのパブリケーションがすでに存在している。
  - アーティクルの複写定義がすでに存在している。

新しいパブリケーションへのアーティクルの追加

- パブリケーションを作成したら、**create article** を使用してアーティクルを作成し、パブリケーションに割り当てます。アーティクルは、テーブル複写定義またはファンクション複写定義と、親パブリケーションを指定します。サブスクリプションを作成するレプリケートサイトの必要性に応じて、オプションで **where** 句を指定することもできます。
  パブリケーションを確定化したり、パブリケーションのサブスクリプションを作成したりするには、パブリケーションに 1 つ以上のアーティクルが含まれている必要があります。詳細については、**create publication** コマンドを参照してください。

アーティクルとサブスクリプション

- パブリケーションのサブスクリプションを作成する場合、Replication Server はそのアーティクルごとに内部サブスクリプションを作成します。
- アーティクルの複数の **where** 句を **or** キーワードで区切って指定することによって、サブスクリプションごとに **where** 句を 1 つしか指定できないという Replication Server の制限に対処できます。パブリケーションサブスクリプションには **where** 句を指定できないので、代わりにアーティクル内で **where** 句を使用してください。

サブスクリプションがあるパブリケーションへのアーティクルの追加

- 既存のパブリケーションに新しいアーティクルを追加したり、パブリケーションからアーティクルを削除したりする場合、そのパブリケーションは不確定化されます。既存のアーティクルの複写は引き続き影響を受けませんが、新しいアーティクルの複写を開始するには、次のようにしてください。
  - パブリケーションへの変更が終了したら、パブリケーションを確定化する。
  - 次に、パブリケーションのサブスクリプションをリフレッシュする。

  パブリケーションサブスクリプションをリフレッシュする 2 つの方法の詳細については、**create subscription** コマンドと **define subscription** コマンドを参照してください。 また、**validate publication** コマンドも参照してください。

パーミッション

create article には、“create object” パーミッションが必要です。

参照：

- check publication (231 ページ)
- create applied function replication definition (274 ページ)
- create publication (341 ページ)
- create replication definition (346 ページ)
- create request function replication definition (362 ページ)
- create subscription (376 ページ)
- define subscription (395 ページ)
- drop article (403 ページ)
- drop publication (418 ページ)
- validate publication (524 ページ)

create auto partition path

用途に応じて自動的にサイズが変更されるパーティションを作成します。

構文

```
create auto partition path logical_name
on 'physical_path'
with auto expand size = size
max size = max_size
```

パラメータ

- ***logical_name*** – 自動でサイズ変更可能な Replication Server パーティションに対する論理パーティションパス名。 Replication Server は用途に応じて自動的にパーティションファイルを作成し、ファイルを論理パーティションパスに割り当てます。 名前は識別子の規則に従う必要があります。 *logical_name* は、**drop auto partition path** コマンドおよび **alter auto partition path** コマンドでも、自動でサイズ変更可能なパーティションの指定に使用されます。
- ***physical_path*** – Replication Server が自動的に作成するパーティションファイルの実ロケーションです。 有効な物理パスを指定し、*physical_path* を一重引用符で囲む必要があります。
- **size** – Replication Server が自動でサイズ変更可能なパーティションで自動作成できるパーティションファイルに対して設定できるメガバイト単位のサイズ。
  - 最小値 - 16MB
  - 最大値 - 1,048,576MB
- ***max_size*** – 自動でサイズ変更可能なパーティションで自動作成されるすべてのパーティションファイルの合計サイズに対して設定できるメガバイト単位の上限。
  - 最小値 - 16MB
  - 最大値 - 2,147,483,647MB

例

- **例 1** – パーティションファイルサイズを 100MB として `usr/user1` パスで `auto_uxp` 論理パーティションパスを作成して、パーティションファイルの使用率が 80% に達するたびに新しい 100MB のパーティションファイルを `auto_uxp` に追加して、自動的に作成されるすべてのパーティションファイルの合計を 102,400MB に制限します。

```
create auto partition path  auto_uxp on '/usr/user1'
with auto expand size=100 max size=102400
```

- **例 2 –** パーティションファイルサイズを 100MB として `c:¥repserver¥partitions¥auto` Windows ディレクトリで `auto_winp` 論理パーティションパスを作成して、パーティションファイルの使用率が 80% に達するたびに新しい 100MB のパーティションファイルを `auto_uxp` に追加して、自動的に作成されるすべてのパーティションファイルの合計を 102,400MB に制限します。

```
create auto partition path  auto_winp on 'c:¥repserver¥partitions
¥auto'
with auto expand size=100 max size=102400
```

使用法

- Replication Server は、パーティションをステーブルメッセージキューに使用します。 メッセージキューは、Replication Server がデータを関連する宛先に送信するまでデータを保持します。
- ローデバイス上に自動でサイズ変更可能なパーティションを作成することはできません。
- 自動でサイズ変更可能なパーティションファイルは、オペレーションシステムファイルでのみ作成できます。
- **auto expand size** と **max size** の値を指定する必要があります。Replication Server ではこれらのパラメータのデフォルト値は指定されないためです。
- Replication Server はパーティションの使用率が 80% に達するたびに新しいパーティションファイルを自動的に作成します。
- パーティションの消費量が下がった場合、Replication Server は自動的にディスク容量を温存します。自動でサイズ変更可能なパーティションで追加された最後のパーティションファイルが空になると、これを削除します。 Replication Server では、ディスク消費量の変動が大きい場合に、応答時間と複写パフォーマンスに影響が出ないように、最後のパーティションファイルを除いた合計パーティション使用率が 50% 未満になるまで、パーティションの縮小を遅らせます。

  Replication Server は、削除されたパーティションファイルを削除しますが、ファイルシステムからは物理パーティションファイルを削除しません。 ディスク領域をリサイクルするには、物理ファイルを手動で削除する必要があります。
- Replication Server は、自動的に作成したパーティションファイルに "*logical_name*"_"*partition_number*" 形式に基づいて名前を付けます。このとき、*partition number* は、Replication Server が追加するファイルごとに、自動的に生成し、0000000001 から 2147483647 までの範囲で順次増分する 10 桁の数値です。 Replication Server は、この形式の名前のパーティションファイルでのみ、パーティションを大きくしたり小さくしたりできます。 他のパー

ティションはすべて、Replication Server 管理者によって手動で管理する必要があります。

警告！ パーティション物理ロケーション上の *logical name_partition number* 形式の名前のファイルは、Replication Server により使用されていない場合を除いて、データが含まれている可能性があるので、手動で削除しないでください。 **admin disk_space** を使用して、Replication Server がそのパーティションファイルを使用しているかどうかチェックできます。 システム管理者またはテクニカルサポートにお問い合わせください。

- 自動でサイズ変更可能なパーティションを複数作成することはできますが、各パーティションは一意の物理パスまたは物理名と、十分なディスク容量を持つ必要があります。
- **create partition** で新規パーティションを作成する場合、既存の自動でサイズ変更可能なパーティションの論理名または物理パスは使用できません。
- **create auto partition path** では利用可能なディスク領域は確認されないため、パーティションを作成する前に、十分なディスク領域があることを確認してください。
- **disk_affinity** パラメータを使用して、パーティション関係として自動でサイズ変更可能なパーティションを選択することはできません。『Replication Server 管理ガイド 第 2 巻』の「ステーブルキューのディスクパーティションの選択」を参照してください。

パーミッション

create auto partition path を使用するには、ディスクパーティションまたはオペレーティングシステムファイルは、“sybase” ユーザが所有するようにします。このユーザには、パーティションに対する読み込み/書き込みパーミッションが必要です。 “sybase” 以外のユーザには、このパーティションに対する読み込み/書き込みパーミッションを付与しないでください。

参照：

- admin auto_part_path (53 ページ)
- alter auto partition path (133 ページ)
- drop auto partition path (405 ページ)
- rs_helppartition (714 ページ)
- admin disk_space (57 ページ)
- create partition (339 ページ)

create connection

複写システムにデータベースを追加し、コネクションの設定パラメータを設定します。 Adaptive Server データベースへのコネクションを作成するには、**rs_init** を使用します。 Adaptive Server 以外のデータベースのコネクションを作成するには、**create connection using profile** コマンドを参照してください。

構文

```
create connection to data_server.database
set error class [to] error_class
set function string class [to] function_class
set username [to] user
[set password [to] passwd]
[set dsi_connector_sec_mech [to] hdbuserstore]
[set replication server error class [to] rs_error_class]
[set database_param [to] 'value' [set database_param [to]
'value']...]
[set security_param [to] 'value' [set security_param [to]
'value']...]
[with {log transfer on, dsi_suspended}]
[as active for logical_ds.logical_db |
as standby for logical_ds.logical_db
[use dump marker]]
```

パラメータ

- **data_server** – 複写システムに追加するデータベースを持つデータサーバです。
- **database** – 複写システムに追加するデータベースです。
- **error_class** – データベースのエラーを処理するエラークラスです。
- **function_class** – データベースのオペレーションに使用するファンクション文字列クラスです。
- **user** – データベースの Replication Server メンテナンスユーザのログイン名です。Replication Server は、複写データを管理するのにこのログイン名を使用します。ネットワークベースセキュリティを有効に設定していない場合には、ユーザ名を指定する必要があります。
- **passwd** – メンテナンスユーザのログイン名のパスワードです。ネットワークベースセキュリティメカニズムが有効になっていない場合は、パスワードを指定する必要があります。
- **dsi_connector_sec_mech** – DSI コネクタのセキュリティメカニズムを指定します。

- **rs_error_class –** データベースの Replication Server エラーを処理するエラークラスです。デフォルトは **rs_repserver_error_class** です。
- **database_param –** Replication Server からのデータベースコネクションに影響を与えるパラメータです。パラメータと値については、表 18 : データベースコネクションに影響を与えるパラメータで説明されています。
- **value –** オプションの値を持つ文字列です。
- **security_param –** ネットワークベースのセキュリティに影響を与えるパラメータです。 **create connection** を使用して設定できるセキュリティパラメータのリストと説明については、「ネットワークベースのセキュリティに影響を与えるパラメータ」を参照してください。 このパラメータは ASE 以外、IQ 以外のコネクタには適用されません。
- **log transfer on –** コネクションがプライマリデータの送信元であるか、または複写ファンクションの送信元であるかを示します。この句を指定すると、Replication Server はインバウンドキューを作成し、データベースの RepAgent コネクションを受け入れる準備をします。このオプションを省略すると、コネクションは RepAgent からの入力を受け入れることができません。
- **dsi_suspended –** DSI スレッドをサスペンドした状態でコネクションを開始します。 DSI は後でレジュームできます。 このオプションは、Replication Server コネクションをサポートしていない、SAP 以外のデータサーバに接続する場合に有用です。
- **as active for –** コネクションが論理コネクションのアクティブデータベースへの物理コネクションであることを示します。
- **as standby for –** コネクションが論理コネクションのスタンバイデータベースへの物理コネクションであることを示します。
- **logical_ds –** 論理コネクションのデータサーバの名前です。
- **logical_db –** 論理コネクションのデータベースの名前です。
- **use dump marker –** アクティブデータベースからの一連のトランザクションにある有効な複写マーカより後ろにある最初のダンプマーカを受け取った後、スタンバイデータベースにトランザクションを適用するように、Replication Server に指示します。このオプションの指定がない場合、Replication Server は有効な複写マーカの後に受け取ったトランザクションを適用します。

  注意： MSA 複写でプラットフォーム間の dump と load (XPDL) 機能を使用する場合、マテリアライズに **use dump marker** 句を使用しないでください。

例

- **例 1 –** SYDNEY_DS データサーバの *pubs2* データベースのコネクションを作成します。データベースのエラー処理には、*ansi_error* エラークラスを使用します。データ操作オペレーションには *sqlserver_derived_class* ファンクション文字列クラスのファンクション文字列を使用します。コネクションは、ログイン名

pubs2_maint とパスワード *pubs2_maint_ps* を使用して、*pubs2* データベースにログインします。

```
create connection to SYDNEY_DS.pubs2
 set error class ansi_error
 set function string class sqlserver_derived_class
 set username pubs2_maint
 set password pubs2_maint_pw
```

- **例 2 –** 例 1 と同様のコネクションを作成します。ただし、この例では、*tokyo_rs_error* Replication Server エラークラスでそのコネクションの Replication Server エラーを処理し、**with log transfer** 句が指定されています。これにより、コネクションは RepAgent からの入力を受け入れることができるようになります。このコネクションは、プライマリデータを格納するデータベース、または複写ファンクションの送信元となるデータベースとのコネクションです。

```
create connection to TOKYO_DS.pubs2
set error class ansi_error
set function string class sqlserver_derived_class
set username pubs2_maint
set password pubs2_maint_pw
set replication server error class tokyo_rs_error
with log transfer on
```

使用法

- **create connection** は、複写システムにデータベースを追加するときに使用します。 通常、このコマンドは、コネクションを SAP 以外のデータベースに追加する場合に使用します。 Adaptive Server データベースへの標準接続を作成するには、**rs_init** を使用します。
- 異機種データ型サポート (HDS) を使用してプライマリデータベースのデータ型をレプリケートデータベースのデータ型に変換するコネクションを作成するには、コネクションの作成と HDS のインストールの両方を行う、スクリプトを使用することもできます。 この手順については、使用しているプラットフォーム用の『Replication Server 設定ガイド』を参照してください。
- **create connection** は、データベースを管理する Replication Server で実行します。
- Replication Server は、データベースコネクション情報を、条件を満たしているサイトへ複写システムを介して分配します。複写システムで通常の遅延時間が発生するため、変更はすぐにはレプリケートサイトに反映されません。
- デフォルトのエラークラス *rs_sqlserver_error_class* を使用する場合でも、エラークラスを指定する必要があります。
- Replication Server エラークラスが新しい Replication Server エラークラスでないかぎり、指定する必要はありません。デフォルトの Replication Server のエラークラスは *rs_repserver_error_class* です。
- 1 つのデータベースに許可されるコネクションは 1 つだけです。これは、各データベースをその *rs_idnames* システムテーブル内に登録する ID サーバに

よって強制されています。データベースのコネクションを作成する場合、ID サーバが使用できなければなりません。
- SAP 以外のデータサーバのクラスレベル変換をアクティブにするには、**set function string class [to]** *function_class* を使用します。

データベースコネクションパラメータ

- Replication Server の設定パラメータは、*rs_config* システムテーブルに格納されています。データベースコネクションパラメータ (*rs_config* システムテーブル内) の詳細については、『Replication Server 管理ガイド 第 1 巻』を参照してください。
- 並列 DSI スレッドの設定の詳細については、『Replication Server 管理ガイド 第 2 巻』を参照してください。
- **assign action** を使用すると、データサーバの特定のエラーが原因で失敗したトランザクションをリトライできるようになります。

dump_load 設定パラメータ

- **dump_load** を “on” に設定する前に、**rs_dumpdb** ファンクションと **rs_dumptran** ファンクションのファンクション文字列を作成してください。Replication Server は、システムによって提供されるクラスやそのクラスから継承された派生クラスでは、これら 2 つのファンクションのファンクション文字列は生成しません。

save_interval 設定パラメータ

- **save_interval** を設定すると、データベースがバックアップからリストアされた後、データベースを再同期するために使用される DSI キューにトランザクションが保存されます。セーブインターバルの設定は、レプリケートデータの保持、または複写ファンクションの受信を行うデータベースのウォームスタンバイを設定する場合にも使用できます。**sysadmin restore_dsi_saved_segments** を使用すると、バックログトランザクションをリストアできます。

エラークラスとファンクションクラス

- 表 31 : エラークラスとファンクションクラスに、Replication Server が Replication Server とデータベースコネクションに提供する、エラークラスとファンクションクラスを示します。

表 31 : エラークラスとファンクションクラス

クラス名	説明
rs_repserver_error_class	Replication Server 用に割り当てられたエラーアクション。

クラス名	説明
rs_sqlserver_error_class	Adaptive Server データベース用に割り当てられたエラーアクション。
rs_sqlserver_function_class	Adaptive Server データベースのファンクション文字列クラス。 ファンクション文字列継承には関与できない。 Replication Server は、ファンクション文字列を自動的に生成する。
rs_default_function_class	Adaptive Server データベースのファンクション文字列クラス。 ファンクション文字列は修正できない。 このクラスは、親クラスとして指定できるが、派生クラスとしては指定できない。 Replication Server は、ファンクション文字列を自動的に生成する。
rs_db2_error_class	DB2 データベースのエラークラス。
rs_db2_function_class	DB2 データベースのファンクション文字列クラス。 ファンクション文字列は修正できない。 このクラスは、親クラスとして指定できるが、派生クラスとしては指定できない。 Replication Server は、ファンクション文字列を自動的に生成する。
rs_hanadb_error_class	HANA DB2 データベースのエラークラス。
rs_hanadb_function_class	HANA DB2 データベースのファンクション文字列クラス。 ファンクション文字列は修正できない。 このクラスは親クラスとして指定できるが、その派生クラスは親クラスからクラスレベル変換を継承できない。 Replication Server は、ファンクション文字列を自動的に生成する。
rs_iq_error_class	SAP IQ データベースのエラークラス。
rs_iq_function_class	SAP IQ データベースのファンクション文字列クラス。 ファンクション文字列は修正できない。 このクラスは親クラスとして指定できるが、その派生クラスは親クラスからクラスレベル変換を継承できない。 Replication Server は、ファンクション文字列を自動的に生成する。
rs_msss_error_class	Microsoft SQL Server データベースのエラークラス。
rs_ms_function_class	Microsoft SQL Server データベースのファンクション文字列クラス。 ファンクション文字列は修正できない。 このクラスは親クラスとして指定できるが、その派生クラスは親クラスからクラスレベル変換を継承できない。 Replication Server は、ファンクション文字列を自動的に生成する。
rs_oracle_error_class	Oracle データベースのエラークラス。

クラス名	説明
rs_oracle_function_class	Oracle データベースのファンクション文字列。 ファンクション文字列は修正できない。 このクラスは親クラスとして指定できるが、その派生クラスは親クラスからクラスレベル変換を継承できない。 Replication Server は、ファンクション文字列を自動的に生成する。
rs_udb_error_class	UDB データベースのエラークラス。
rs_udb_function_class	UDB データベースのファンクション文字列クラス。 ファンクション文字列は修正できない。 このクラスは親クラスとして指定できるが、その派生クラスは親クラスからクラスレベル変換を継承できない。 Replication Server は、ファンクション文字列を自動的に生成する。

注意： Replication Server が生成したデフォルトのファンクション文字列を持つファンクション文字列クラスの場合でも、**rs_dumpdb** および **rs_dumptran** システムファンクションは、最初は定義されていません。コーディネートダンプを使用する場合には、これらのファクションのファクション文字列を作成する必要があります。スタンバイデータベース上では、コーディネートダンプを実行できないことにも注意してください。ファンクション文字列の使用方法の詳細については、『Replication Server 管理ガイド 第 2 巻』を参照してください。**rs_dumpdb** ファンクションと **rs_dumptran** ファンクションの詳細については、「Replication Server システムファンクション」を参照してください。

ユーザ名とパスワード

- コネクションの作成時に、メンテナンスユーザのログイン名とパスワードを指定します。メンテナンスユーザのログイン名には、データベース内の複写データを管理するために必要なパーミッションが、すべて付与されていなければなりません。

注意： 複写システムにある 2 つのサイトのデータベース名が同じである場合、メンテナンスユーザのログイン名は別にする必要があります。 デフォルトのログイン名は、**rs_init** によって作成され、*DB_name*_maint になります。 システムの設定時に、いずれかのログイン名を変更して、それぞれがユニークになるようにします。

ウォームスタンバイアプリケーション

- ウォームスタンバイアプリケーションの論理コネクションを作成するには、**create logical connection** を使用します。
- ウォームスタンバイアプリケーションでは、アクティブデータベースとスタンバイデータベースのコネクションに、**log transfer on** が必要です。
- ウォームスタンバイアプリケーションにおけるデータベースのファンクション文字列クラスは、データベースがアクティブデータベースの場合にのみ使用さ

れます。Replication Server は、スタンバイデータベースに対しては、*rs_default_function_class* を使用します。

コネクション属性の変更

- **alter connection** を使用すると、コネクションの属性を変更できます。
- メンテナンスユーザのパスワードが変更されている場合は、**alter connection** を使用して新しいパスワードを入力します。

ネットワークベースセキュリティのパラメータ

- これらのパラメータは ASE 以外、IQ 以外のコネクタには適用されません。
- コネクションの両端では、同じセキュリティメカニズムとセキュリティ機能を備えた互換性のある SCL (Security Control Layer) ドライバを使用してください。また、リモートサーバは、**set proxy** または同等のコマンドをサポートしている必要があります。各サーバについて、セキュリティ機能の選択と設定を行うのは複写システム管理者の仕事です。Replication Server は、リモートサーバとのコネクションを確立しようとする前に、そのサーバのセキュリティ機能の問い合わせは行いません。コネクションの両端のセキュリティ機能に互換性がないと、コネクションは失敗します。
- **create connection** を使用すると、Replication Server からターゲットデータサーバへの送信コネクションのセキュリティ設定を指定できます。**create connection** を使用して設定したセキュリティ機能は、**configure replication server** を使用して設定したセキュリティ機能よりも優先されます。
- **unified_login** を “required” に設定すると、“sa” パーミッションを持つ複写システム管理者だけがクレデンシャルなしで Replication Server にログインできます。セキュリティメカニズムに問題が発生した場合でも、複写システム管理者はパスワードを使用して Replication Server にログインし、**unified_login** を無効にできます。
- Replication Server には、複数のセキュリティメカニズムを装備できます。サポートされるメカニズムは、それぞれ `libtcl.cfg` ファイル内の SECURITY セクションにリストされています。
- メッセージの暗号化は、深刻なパフォーマンスの低下をともなう、負荷の高い処理です。通常は、特定のコネクションに対してだけ **msg_confidentiality** を “required” に設定してください。代わりに、**msg_integrity** などの負荷の低いセキュリティ機能を選択します。

パーミッション

create connection には、“sa” パーミッションが必要です。

参照：

- admin show_connection_profiles (74 ページ)

- alter connection (134 ページ)
- create alternate connection (269 ページ)
- create connection using profile (294 ページ)
- configure connection (238 ページ)
- create error class (308 ページ)
- create function string class (334 ページ)
- create logical connection (338 ページ)
- alter route (213 ページ)
- drop connection (407 ページ)
- resume connection (436 ページ)
- rs_classes (772 ページ)
- rs_profdetail (811 ページ)
- rs_profile (812 ページ)
- rs_systext (836 ページ)
- suspend connection (452 ページ)

create connection using profile

create connection using profile では、あらかじめ定義された情報を使用して、Replication Server および Adaptive Server 以外のデータベース間のコネクションを設定し、必要に応じて RSSD および指定した *data_server.database* を修正します。Adaptive Server へのコネクションを確立する方法については、**create connection** を参照してください。

構文

```
create connection to data_server.database
using profile connection_profile;version
set username [to] user
[other_create_connection_options]
[display_only]
```

パラメータ

- **data_server** – 複写システムに追加するデータベースを持つデータサーバです。
- **database** – 複写システムに追加するデータベースです。
- **connection_profile** – コネクションの設定、RSSD の修正、およびレプリケートデータベースオブジェクトの作成に使用する接続プロファイルを示します。
- **version** – 使用する接続プロファイルのバージョンを指定します。

- **user –** データベースの Replication Server メンテナンスユーザのログイン名です。Replication Server は、複写データを管理するのにこのログイン名を使用します。ネットワークベースセキュリティを有効に設定していない場合には、ユーザ名を指定する必要があります。
- **other_create_connection_options –** プロファイルで指定されない接続オプションの設定 (パスワードの設定など)、またはプロファイルで指定されているオプションの上書き (Replication Server に用意されているファンクション文字列クラスを上書きするカスタムファンクション文字列クラスの指定など) を行うには、他の **create connection** オプションを使用します。他の **create connection** オプションのリストについては、**create connection** を参照してください。
- **display_only – display_only** は、**using profile** 句とともに使用し、実行されるコマンド、およびそのコマンドを実行するサーバの名前を表示します。**display_only** を使用した結果については、クライアントログおよび Replication Server ログを参照してください。

例

- **例 1 –** Oracle レプリケートデータベースに対するコネクションを作成します。

```
create connection to oracle.instance
using profile rs_ase_to_oracle;standard
set username to ora_maint
set password to ora_maint_pwd
```

- **例 2 –** プライマリデータベースでもある Microsoft SQL Server レプリケートデータベースに対するコネクションを作成します。 この例では、コマンドにより、接続プロファイルによって提供されるエラークラス設定が *my_msss_error_class* エラークラスに置き換えられます。

```
create connection to msss_server.msss_db
using profile rs_ase_to_msss;standard
set username to msss_maint
set password to msss_maint_pwd
set error class to my_msss_error_class
with log transfer on
```

- **例 3 –** プロファイルの特定のバージョン v9_1 を使用して、DB2 レプリケートデータベースに対するコネクションを作成します。この例では、接続プロファイルによって提供されているコマンドバッチのサイズが、このコマンドにより新しい値 16384 で上書きされます。

```
create connection to db2.subsys
using profile rs_ase_to_db2;v9_1
set username to db2_maint
set password to db2_maint_pwd
set dsi_cmd_batch_size to '16384'
```

- **例 4 – display_only** オプションを使用して、特定のプロファイルを使用した場合に実行されるコマンドを表示します。コマンドと画面に表示されるコマンド出力は、Replication Server のログにも書き込まれます。

```
create connection to oracle.instance
using profile rs_ase_to_oracle;standard
set username to ora_maint
set password to ora_maint_pwd
display_only

go
```

```
Display only using Connection Profile rs_ase_to_oracle;standard.

Command(s) intended for: prs01
create connection to oracle.instance
set error class to rs_oracle_error_class
set function string class to rs_oracle_function_class
set username to ora_maint
set password to ********
set batch to off

Command(s) intended for 'edsprs01.edbprs01':
delete from rs_translation where classid = 0x0000000001000007 and
source_dtid = 0x000000000000000c

Command(s) intended for 'edsprs01.edbprs01':
insert rs_translation (prsid, classid, type, source_dtid,
target_dtid,
target_length, target_status, rowtype)
values (0, 0x0000000001000007, 'D', 0x000000000000000c,
0x0000000000010200,
19, 0, 0)

Command(s) intended for 'edsprs01.edbprs01':
delete from rs_translation where classid = 0x0000000001000007  and
source_dtid = 0x000000000000000d

Command(s) intended for 'edsprs01.edbprs01':
insert rs_translation (prsid, classid, type, source_dtid,
target_dtid,
target_length, target_status, rowtype)
values (0, 0x0000000001000007, 'D', 0x000000000000000d,
0x0000000000010200,
19, 0, 0)

Command(s) intended for 'edsprs01.edbprs01':
delete from rs_translation where classid = 0x0000000001000007 and
source_dtid = 0x0000000000000001

Command(s) intended for 'edsprs01.edbprs01':
insert rs_translation (prsid, classid, type, source_dtid,
target_dtid,
target_length, target_status, rowtype)
values (0, 0x0000000001000007, 'D', 0x0000000000000001,
```

```
0x0000000000010202,
0, 0, 0)
Command(s) intended for 'edsprs01.edbprs01':
delete from rs_translation where classid = 0x0000000001000007 and
source_dtid = 0x0000000000000013

Command(s) intended for 'edsprs01.edbprs01':
insert rs_translation (prsid, classid, type, source_dtid,
target_dtid,
target_length, target_status, rowtype)
values (0, 0x0000000001000007, 'D', 0x0000000000000013,
0x0000000000010202,
0, 0, 0)

Command(s) intended for 'edsprs01.edbprs01':
delete from rs_translation where classid = 0x0000000001000007 and
source_dtid = 0x000000000000000E

Command(s) intended for 'edsprs01.edbprs01':
insert rs_translation (prsid, classid, type, source_dtid,
target_dtid,
target_length, target_status, rowtype)
values (0, 0x0000000001000007, 'D', 0x000000000000000E,
0x0000000000010205,
136, 0, 0)

Command(s) intended for 'edsprs01.edbprs01':
delete from rs_translation where classid = 0x0000000001000007 and
source_dtid = 0x000000000000000F

Command(s) intended for 'edsprs01.edbprs01':
insert rs_translation (prsid, classid, type, source_dtid,
target_dtid,
target_length, target_status, rowtype)
values (0, 0x0000000001000007, 'D', 0x000000000000000f,
0x0000000000010205,
136, 0, 0)

Command(s) intended for 'edsprs01.edbprs01':
delete from rs_translation where classid = 0x0000000001000007 and
source_dtid = 0x000000000000001b

Command(s) intended for 'edsprs01.edbprs01':
insert rs_translation (prsid, classid, type, source_dtid,
target_dtid,
target_length, target_status, rowtype)
values (0, 0x0000000001000007, 'D', 0x000000000000001b,
0x0000000000010201,
9, 0, 0)

Command(s) intended for 'edsprs01.edbprs01':
delete from rs_translation where classid = 0x0000000001000007  and
source_dtid = 0x000000000000001c

Command(s) intended for 'edsprs01.edbprs01':
insert rs_translation (prsid, classid, type, source_dtid,
```

```
target_dtid,
target_length, target_status, rowtype)
values (0, 0x0000000001000007, 'D', 0x000000000000001c,
0x0000000000010200,
19, 0, 0)

Command(s) intended for 'oracle.instance':
drop table rs_info

Command(s) intended for 'oracle.instance':
commit

Command(s) intended for 'oracle.instance':
create table rs_info (rskey varchar2 (20), rsval varchar2 (20))

Command(s) intended for 'oracle.instance':
commit

Command(s) intended for 'oracle.instance':
insert into rs_info values ('charset_name', 'iso_1')

Command(s) intended for 'oracle.instance':
insert into rs_info values ('sortorder_name', 'bin_iso_1')

Command(s) intended for 'oracle.instance':
commit

Command(s) intended for 'oracle.instance':
drop public synonym rs_lastcommit

Command(s) intended for 'oracle.instance':
commit

Command(s) intended for 'oracle.instance':
drop table rs_lastcommit

Command(s) intended for 'oracle.instance':
commit

Command(s) intended for 'oracle.instance':
create table rs_lastcommit(origin  number(8),origin_qid char(72),
secondary_qid char(72),origin_time  date,
dest_commit_time date)

Command(s) intended for 'oracle.instance':
commit

Command(s) intended for 'oracle.instance':
grant all on rs_lastcommit to public

Command(s) intended for 'oracle.instance':
commit

Command(s) intended for 'oracle.instance':
create public synonym rs_lastcommit for rs_lastcommit
```

```
Command(s) intended for 'oracle.instance':
commit

Command(s) intended for 'oracle.instance':
CREATE OR REPLACE PROCEDURE
RS_UPDATE_SEQUENCE(SequenceName VARCHAR2, SequenceValue NUMBER,
Increment NUMBER)
AS CurrentID NUMBER; LastID NUMBER; SeqCursor INTEGER; SQLStmt
VARCHAR2(1024);
Result NUMBER;
BEGIN
SQLStmt := 'SELECT ' || SequenceName || '.NEXTVAL FROM DUAL';
SeqCursor := DBMS_SQL.OPEN_CURSOR;
DBMS_SQL.PARSE(SeqCursor,SQLStmt,DBMS_SQL.NATIVE);
DBMS_SQL.DEFINE_COLUMN(SeqCursor, 1, LastID);
Result := DBMS_SQL.EXECUTE_AND_FETCH(SeqCursor);
DBMS_SQL.COLUMN_VALUE(SeqCursor,1,CurrentID);
LOOP
IF ( Increment < 0 ) THEN EXIT WHEN CurrentID <= SequenceValue;
EXIT WHEN CurrentID > LastID;
ELSE EXIT WHEN CurrentID >= SequenceValue;
EXIT WHEN CurrentID < LastID;
END IF;
LastID := CurrentID;
Result := DBMS_SQL.EXECUTE_AND_FETCH(SeqCursor);
DBMS_SQL.COLUMN_VALUE(SeqCursor,1,CurrentID);
END
LOOP;
DBMS_SQL.CLOSE_CURSOR(SeqCursor);
END;

Command(s) intended for 'oracle.instance':
grant all on RS_UPDATE_SEQUENCE to public

Command(s) intended for 'oracle.instance':
DROP sequence rs_ticket_seq

Command(s) intended for 'oracle.instance':
CREATE sequence rs_ticket_seq

Command(s) intended for 'oracle.instance':
Drop table rs_ticket_history

Command(s) intended for 'oracle.instance':
CREATE TABLE rs_ticket_history(cnt numeric(8,0, h1 varchar(10,
h2 varchar(10), h3 varchar(10), h4 varchar(50), pdb varchar(30),
prs varchar(30), rrs varchar(30), rdb varchar(30), pdb_t date,
exec_t date, dist_t date, rsi_t date, dsi_t date,
rdb_t date default current_date, exec_b int, rsi_b int, dsi_tnx
int,
dsi_cmd int, ticket varchar(1024))

Command(s) intended for 'oracle.instance':
create unique index rs_ticket_idx on rs_ticket_history(cnt)

Command(s) intended for 'oracle.instance':
```

```
create or replace trigger rs_ticket_tri
before insert on rs_ticket_history
for each row
begin
if :new.cnt is null then
select rs_ticket_seq.nextval into :new.cnt  from dual;
end if;
end rs_ticket_tri;Command(s) intended for 'oracle.instance':
grant all on rs_ticket_history to public

Command(s) intended for 'oracle.instance':
commit
```

使用法

- 接続プロファイルでは、ファンクション文字列クラスとエラークラスを指定します。 また、コマンドをバッチ処理するかどうかなどの他の接続オプションや、使用するコマンドセパレータを指定することもできます。 接続設定の他にも、接続プロファイルでは、RSSD にインストールするクラスレベル変換や、レプリケートデータベースに作成される `rs_lastcommit` テーブルなどのオブジェクトを指定できます。
- 接続プロファイルを使用してコネクションを作成するときに、システムテーブルサービス (STS: System Table Services) キャッシュがリフレッシュされるため、Replication Server を再起動する必要はありません。
- **set username** 句は、必ず **using profile** 句のすぐ後に指定します。
- **admin show_connection_profiles** を使用して、Replication Server で定義されている各プロファイルの接続プロファイル名、バージョン、コメントをリストします。
- レプリケート Adaptive Server データベースへの接続の作成に使用できるのは **ase_to_ase** プロファイルのみです。

参照：

- admin show_connection_profiles (74 ページ)
- create connection (287 ページ)

create database replication definition

データベースまたはデータベースオブジェクトを複写するための複写定義を作成します。

構文

```
create database replication definition db_repdef
      with primary at server_name.db
      [not replicate DDL] |
```

```
      [replicate DDL [{{with | without} auto_update_table_list} |
      {{with | without} auto_extend_table_list}]]
      [[not] replicate functions setcont]
      [[not] replicate transactions setcont]
      [[not] replicate system procedures setcont]
     [[not] replicate tables [[setcont [except setcont]] | in files
('file_path')]
       [[not] replicate {SQLDML | DML_options} [in table_list]]
       [user username password pass]]
setcont  ::= [[in] ([owner1.]name1[, [owner2.]name2 [, ... ]])]
```

パラメータ

- **db_repdef –** データベース複写定義の名前です。
- **server_name.db –** プライマリサーバとデータベースの組み合わせの名前です。例：*TOKYO.dbase*.
- **[not replicate DDL] –** サブスクライブするデータベースに DLL を送信しないよう Replication Server に指示します。
- **[replicate DDL [{{with | without} auto_update_table_list} | {{with | without} auto_extend_table_list}]] –** Replication Server に対し、サブスクライブするデータベースに DDL を送信するよう指示します。また、テーブルリストを更新または拡張するかどうかを指示します。 オプションなしで **replicate DDL** を指定すると、DDL はレプリケートデータベースに送信されますが、テーブルは複写パスに追加されません。
- **replicate DDL with auto_update_table_list –** DDL コマンドをレプリケートデータベースに送信します。 テーブルで **pdb_automark_tables** が true の場合、DDL コマンドの **drop table** または **rename table** が検出されると、そのテーブルはテーブルリストで自動的に更新されます。
- **replicate DDL without auto_update_table_list –** DDL コマンドをレプリケートデータベースに送信しますが、DDL コマンドの **drop table** または **rename table** が検出されても、テーブルリスト内のテーブルを更新しません。
- **replicate DDL with auto_extend_table_list –** DDL コマンドをレプリケートデータベースに送信します。 テーブルで **pdb_automark_tables** が true の場合、DDL コマンドの **create table** が検出されると、そのテーブルは自動的にテーブルリストに追加されます。
- **replicate DDL without auto_extend_table_list –** DDL コマンドをレプリケートデータベースに送信しますが、DDL コマンドの **create table** が検出されても、テーブルをテーブルリストに追加しません。

  注意： replicate DDL コマンドに **auto_extend_table_list** オプションを指定する場合は、システムプロシージャまたは SQLDML オペレーションを同時にレプリケートしないようにしてください。
- **[[not] replicate functions setcont] –** レプリケートデータベースにファンクションを送信するかどうかを指定します。 ファンクションを複写するとき、ファン

クションごとに最大 1 個の句を使用できます。 句を省略すると、Replication Sever によりすべてのファンクションが複写されます。

- **[[not] replicate transactions setcont] –** レプリケートデータベースにトランザクションを送信するかどうかを指定します。 トランザクションを複写するとき、トランザクションごとに最大 1 個の句を使用できます。 句を省略すると、Replication Sever によりすべてのトランザクションが複写されます。
- **[[not] replicate system procedures setcont] –** レプリケートデータベースにシステムプロシージャを送信するかどうかを指定します。 システムプロシージャを複写するとき、システムプロシージャごとに最大 1 個の句を使用できます。 句を省略すると、Replication Sever はシステムプロシージャを複写しません。
- **[[not] replicate tables [[setcont [except setcont]] –** レプリケートデータベースにテーブルを送信するかどうかを指定します。 テーブルを複写するとき、テーブルごとに最大 1 個の句を使用できます。 句を省略すると、Replication Sever によりすべてのテーブルが複写されます。

  例外リストはテーブルリストとのみ使用できます。 例外リストを使用すると、データベース複写定義に定義されたテーブルリストをさらに詳細に調整できます。例外リストはテーブルリストに優先します。
- **[[not] replicate tables in files ('file_path')] –** *file_path* で指定されたテーブルをレプリケートデータベースに送信するかどうかを指定します。

  file_path には、対象テーブルリストまたは除外テーブルリストが記述されています。 一度に 1 つのファイルのみを指定できます。*file_path* には絶対パスを指定する必要があります。

注意： プライマリ Replication Server を起動するユーザにこのファイルの読み取りパーミッションが必要です。

テーブル名の書式は、ファイルでもテーブルリストでも同じです。 テーブルリストでは次の書式でテーブル名を指定できます。

- *ownername.tablename*
- *tablename* (テーブル名は dbo.tablename として格納されます)
- **.tablename*
- *ownername.**
- '*x*y'.'a*b' (文字列にワイルドカードが埋め込まれています)

注意： **create database replication definition** を発行すると、部分ワイルドカードはテーブルリストや例外リストを含むすべてのリストで展開されます。 単純ワイルドカードは、例外リストで展開されてからシステムテーブルに格納されます。 ワイルドカードの展開が必要な場合は、**user** と **password** の値を指定する必要があります。

対象テーブルリストまたは除外テーブルリストを使用するときは、次の次のガイドラインに従ってください。

- ファイル内ではテーブル名のデリミタとして改行文字を使用します。
- # で始まる行はコメントとして無視されます。
- 所有者名またはテーブル名の前の空白はトランケートされます。
- 所有者名の最大長は 30 文字です。

例外リストはテーブルリストとのみ使用できます。 例外リストを使用すると、データベース複写定義に定義されたテーブルリストをさらに詳細に調整できます。例外リストはテーブルリストに優先します。

- **[not] replicate {SQLDML | DML_options} [in table_list] –** SQL 文を、*in table_list* に定義されているテーブルに複写するかどうかを Replication Server に伝えます。
- **SQLDML –** 次の DML オペレーションです。
  - U - **update**
  - D - **delete**
  - I - **insert select**
  - S - **select into**
- **DML_options –** 次の DML オペレーションの任意の組み合わせです。
  - U - **update**
  - D - **delete**
  - I - **insert select**
  - S - **select into**

  データベースの複写モードを **UDIS** の任意の組み合わせに設定すると、RepAgent は、個々のログレコードと Replication Server が SQL 文を作成するために必要な情報の両方を送信します。
- **[user *username* password *pass*] –** プライマリ Adaptive Server database または Replication Agent への接続用のユーザ ID とパスワード。プライマリテーブルから選択します。

  テーブル名にワイルドカードを使用する場合は、**username** と **password** の値を指定する必要があります。

  注意： user と **password** の値は一度だけ使用され、RSSD には保存されません。
- **owner –** テーブルの所有者またはトランザクションを実行するユーザです。

  注意： create database replication definition を実行するとき、ファンクションまたはシステムプロシージャの所有者を指定しないでください。

  owner は、一重引用符で囲まれた 1 つのスペース、またはアスタリスクで置き換えることができます。

  - スペース (' ') - 所有者がいないことを示します。
  - アスタリスク (*) - すべての所有者を表します。たとえば、**.publisher* は、所有者に関係なく、*publisher* という名前のすべてのテーブルを表します。
- **name –** テーブル、ファンクション、トランザクション、またはシステムプロシージャの名前です。

  name は、一重引用符で囲まれた 1 つのスペース、またはアスタリスクで置き換えることができます。

  - スペース (' ') - 名前がないことを示します。たとえば、*maintuser.' '* はメンテナンスユーザのすべての名前のないトランザクションを表します。
  - アスタリスク (*) - すべての名前を表します。 たとえば、*robert.** は、*robert* が所有するすべてのテーブル (またはトランザクション) を表します。

例

- **例 1 –** データベース複写定義 *rep_1B* を作成します。このデータベース複写定義では、テーブル *employee* と *employee_address* だけを複写することを指定しています。

```
create database replication definition rep_1B
  with primary at PDS.pdb
  replicate tables in (employee, employee_address)
```

- **例 2 –** データベース複写定義 **rep_2** を作成します。この例では、データベース *my_db* が複写され、DDL も複写されますが、システムプロシージャは複写されません。

```
create database replication definition rep_2
  with primary at dsA.my_db
  replicate DDL

  not replicate system procedures
```

- **例 3 – insert**、**update**、**delete**、および **select into** コマンドを *pdb1* データベースのすべてのテーブルから複写します。 すべてのトランザクションとファンクションは複写されますが、DDL とシステムプロシージャは複写されません。

```
create database replication definition rep_3
with primary at ds3.pdb1
replicate SQLDML
```

  この例の結果は、前の例と同じです。

```
create database replication definition rep_3
with primary at ds3.pdb1
replicate 'UDSI'
```

- **例 4 –** すべてのテーブルの **select into** 文を除外します。2 つ目の句 **not replicate 'U' in (T)** は、テーブル *T* での update をフィルタします。

```
create database replication definition dbrepdef
with primary at ds1.pdb1
```

```
not replicate 'S'
not replicate 'U' in (T)
go
```

- **例 5 –** replicate 'UD' 句を使用して、すべてのテーブルで **update** 文と **delete** 文を有効にします。

```
create database replication definition dbrepdef_UD
with primary at ds2.pdb1
replicate 'UD'
go
```

- **例 6 –** 複数の句を使用して、同じ定義で 1 つのテーブルを複数回指定できます。ただし、**U**、**D**、**I**、および **S** はそれぞれ、定義ごとに一度しか使用できません。

```
create database replication definition dbrepdef
with primary at ds2.pdb1
replicate tables in (tb1,tb2)
replicate 'U' in (tb1)
replicate 'I' in (tb1,tb2)
go
```

- **例 7 –** データベース内のテーブル *T* 以外のすべてのテーブルに対し、すべてのユーザストアドプロシージャ、システムプロシージャ、および DML を複写する複写定義です。 テーブル *T* の場合は、この複写定義により、**delete** コマンド以外のすべてのコマンドが複写されます。

```
create database replication definition repdef_7
with primary at ds3.pdb1
replicate functions
replicate system procedures
replicate 'IUS' /* replicate 'IUS' DML for all tables, including
*/
/* table 'T' */
not replicate 'D' in (T)/* not replicate 'D' DML for table T, but
*/
/* replicate 'D' for all other tables */
```

- **例 8 –** データベース複写定義 *dbrepdef* を作成します。 DDL が複写され、テーブルがプライマリで複写対象としてマーク付けされている場合は、create table DDL コマンドが検出されると、テーブルが自動的に *table_list* に追加されます。 このコマンドは、テーブル *USER1.TABLE1* のデータと *TABLE2* のすべてのデータも複写します。ただし、*USER2* が所有する *TABLE2* は除きます。

```
create database replication definition dbrepdef
with primary at ds1.pdb1
replicate DDL
with auto_extend_table_list
replicate tables in (USER1.TABLE1, *.TABLE2)  except in
(USER2.TABLE2)
```

- **例 9 –** データベース複写定義 *db_repdef* を作成します。 DDL が複写されますが、新しい DDL コマンドが検出されても *table_list* は更新されません。 このコマンドは、ファイル `/sap/user/tablelist.txt` で指定されたテーブルのデータ

も複写します。これには、テーブル *USER1.TABLE1* と、*TABLE2* の全データが含まれます。

たとえば、`tablelist.txt` には次のように記述されているとします。

```
================
#user tables:
USER1.TABLE1
*.TABLE2
================
```

```
create database replication definition db_repdef
with primary at ds1.pdb1
replicate DDL
replicate tables in files ('/sap/user/table_list.txt')
```

使用法

- **create database replication definition** を使用すると、テーブル、ファンクション、トランザクション、システムプロシージャについて、そのすべて、一部の例外を含むすべて、または一部のみを、プライマリデータベースから複写できます。
- **create database replication definition** は単独で使用するか、テーブル複写定義やファンクション複写定義と組み合わせて使用します。
- データベース複写定義だけを使用した場合 (つまり、テーブル複写定義またはファンクション複写定義を使用しない場合)、Replication Server はデータを変換できません。ただし、最少カラムの複写は実行できます。このデータレプリケーションの動作は、デフォルトのウォームスタンバイの動作と同様です。テーブルレベルの複写定義を使用しないで暗号化カラムを複写するデータベース複写定義では、INIT_VECTOR NULL と PAD NULL を使用して暗号化カラムの暗号化キーを定義します。データベースのテーブルに暗号化カラムが含まれており、その暗号化キーがランダム埋め込み (デフォルト) または初期化ベクトルを使用して作成されている場合は、データベースの一貫性を確保するために、テーブルレベルの複写定義が必要となります。
- データベース複写定義はグローバルオブジェクトです。定義元の Replication Server からのルートを持つすべての Replication Server に複写されます。
- データベース複写定義は、要求ファンクションの複写には影響しません。
- テーブルとファンクションのサブスクリプションが存在する場合、テーブルとファンクションのフィルタは実装されません。
- Replication Server は、ファンクションとシステムプロシージャの所有者情報は処理しません。
- **auto_update_table_list** オプションを使用している場合に **drop table** DDL コマンドが検出されると、テーブルは対象リストから削除されるか、除外リストに追加されます。
- **auto_update_table_list** オプションを使用している場合に **rename table** DDL コマンドが検出されると、テーブルリストのテーブルの名前が変更されます。 元

のテーブルがすでに複写されている場合は、名前を変更したテーブルも複写されます。 それ以外の場合は、名前を変更したテーブルは複写されません。

- **auto_extend_table_list** オプションを使用している場合に **create table** DDL コマンドによりテーブルが自動的にマーク付けされていることが示されているとき、テーブルはレプリケートデータベースで作成され、テーブルデータの複写は、新しく作成されたテーブルに対して行われます。 **create table** DDL コマンドが、テーブルが自動的にマーク付けされていることを示していない場合は、テーブルはレプリケートデータベースに作成されます。 ただし、テーブルデータの複写は、新しく作成したテーブルに対しては行われません。

所有者情報

- Replication Server は、データベース複写定義で提供される所有者情報を常に使用します。
- テーブルが **sp_reptostandby** でマーク付けされている場合、テーブル複写定義で提供される所有者情報は使用されません。
- テーブルが **sp_setreptable** によってマーク付けされている場合 (**owner_on** 句を指定)、Replication Server はテーブル複写定義に指定されている所有者情報だけを使用します。

SQL 文の複写

- SQL 文を MSA 環境で複写するには、複写定義に **replicate SQLDML** 句を含める必要があります。
- **create database replication** 定義では、複数の replicate 句を使用できます。ただし、**alter database replication** 定義では、1 つの句しか使用できません。
- 複写定義でフィルタを指定しない場合、デフォルトは **not replicate** 句です。 SQLDML フィルタを変更するには、**alter database replication definition** を適用します。 **replicate** 句では、1 つまたは複数の SQLDML フィルタを指定できます。
- テーブルに対して **send standby** 句が指定されたテーブル複写定義が定義されている場合、そのテーブル複写定義の SQL 複写設定は、そのテーブルのデータベース複写定義で定義されている設定より優先されます。

参照：

- alter database replication definition (177 ページ)
- drop database replication definition (408 ページ)

create error class

エラークラスを作成します。

構文

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

パラメータ

- **replication server –** 新しいエラークラスが Replication Server エラークラスであり、データサーバのエラークラスではないことを示します。
- **error_class –** 新しいエラークラスの名前です。名前は複写システム内でユニークにし、識別子の規則に従わなければなりません。

  注意： Replication Server エラークラスとデータサーバのエラークラスを同じ名前にすることはできません。
- **set template to template_error_class –** この句を使用して、別のエラークラスに基づいてエラークラスを作成します。**create error class** により、テンプレートのエラークラスのエラーアクションが新しいエラークラスにコピーされます。

例

- **例 1 –** この例では、*pubs2_db_err_class* という新しいエラークラスを作成します。

  ```
  create error class pubs2_db_err_class
  ```
- **例 2 – my_error_class** エラークラスを **rs_oracle_error_class** に基づいて作成します。

  ```
  create error class my_error_class set template to
  rs_oracle_error_class
  ```
- **例 3 – pubs2_rs_err_class** という名前の新しい Replication Server エラークラスを作成します。

  ```
  create replication server error class
  pubs2_rs_err_class
  ```
- **例 4 – my_rs_err_class** Replication Server エラークラスを、デフォルトの Replication Server エラークラスである **rs_repserver_error_class** に基づいて作成します。

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

使用法

- **create error class** は、エラークラスを作成するときに使用します。エラークラスは、データベースに割り当てられたエラーアクションをグループ化するために使用される名前です。
- このコマンドには、次の要件があります。
  - エラークラスを作成した Replication Server から、そのエラークラスを使用するデータサーバを管理する Replication Server へのルートが存在する必要があります。
  - *rs_sqlserver_error_class* は Adaptive Server データベースに用意されているデフォルトのエラークラスであり、*rs_repserver_error_class* は Replication Server に用意されているデフォルトのエラークラスです。最初は、この 2 つのエラークラスにはプライマリサイトがありません。デフォルトのエラーアクションを変更するには、プライマリサイトでこれらのエラークラスを作成する必要があります。
  - **create error class** を使用した後、**rs_init_erroractions** ストアドプロシージャを使用してエラークラスを初期化します。
- **create connection** または **alter connection** を使用して、データベースにエラークラスを関連付けます。各データベースには 1 つのエラークラスがあります。エラークラスは、複数のデータベースに関連付けることができます。
- Replication Server は、新しいエラークラスを、条件を満たしているサイトへ複写システムを介して分配します。複写システムで通常の遅延時間が発生するため、変更内容がレプリケートサイトにすぐに反映されるわけではありません。

エラーアクションの割り当て

- データサーバの特定のエラーに対する Replication Server の応答を変更するには、**assign action** を使用します。エラークラスを作成した Replication Server で、アクションが割り当てられます。

エラークラスの削除

- エラークラスとそのクラスに対応するすべてのアクションを削除するには、**drop error class** を使用します。

Adaptive Server 以外のエラークラス

- **create connection** および **alter connection** コマンドを使用して、Adaptive Server 以外のエラークラスを Adaptive Server 以外のレプリケートデータベースの特定のコネクションに割り当てることができます。
- Replication Server は、ASE 以外のレプリケートサーバへのコネクションを確立するときに、コネクションで ASE 以外のレプリケートサーバからネイティブエラーコードが返されるオプションが有効になっているかどうかを検証します。オプションが有効になっていない場合、Replication Server は、コネクショ

ンは機能しているが、エラーアクションのマッピングが正確でない可能性があることを示す警告メッセージをログに記録します。
Enterprise Connect™ Data Access (ECDA) Option for ODBC でレプリケートサーバ用のオプションを設定するには、Replication Server Options のマニュアルで「**ReturnNativeError**」を参照してください。

- Adaptive Server 以外のエラークラスのリストについては、「エラークラスとファンクションクラス」の表を参照してください。Adaptive Server 以外の複写のエラークラスの詳細については、『Replication Server 管理ガイド 第 1 巻』を参照してください。

パーミッション

create error class には、“sa” パーミッションが必要です。

参照：

- alter connection (134 ページ)
- alter error class (182 ページ)
- assign action (226 ページ)
- create connection (287 ページ)
- drop error class (409 ページ)
- move primary (432 ページ)
- rs_init_erroractions (734 ページ)

create function

ユーザ定義ファンクションを作成します。

注意： ファンクション複写定義を作成すると、ユーザ定義ファンクションが自動的に作成されます。詳細については、**create applied function replication definition** および **create request function replication definition** を参照してください。

アプリケーションでテーブル複写定義に対応する非同期プロシージャの配信を使用する場合、ユーザ定義ファンクションを作成することが必要になる場合があります。詳細については、『Replication Server 管理ガイド 第 2 巻』を参照してください。

構文

```
create function replication_definition.function
([@param_name datatype [, @param_name datatype]...])
```

パラメータ

- **replication_definition –** ファンクション用の複写定義の名前です。同じテーブルでは、すべての複写定義に対してユーザ定義ファンクションを 1 つだけ作成できます。同じテーブルに対して複数の複写定義がある場合は、いずれか 1 つの名前を指定できます。ただし、それぞれの複写定義は、ユーザ定義ファンクションに対して自身のファンクション文字列を持ちます。
- **function –** ファンクションの名前です。名前は複写定義に対してユニークであり、識別子の規則に従う必要があります。「Replication Server システムファンクション」にリストされているシステムファンクションの名前と、“rs_” で始まるすべてのファンクション名は、予約されています。
- **@param_name –** ユーザ定義ファンクションの引数の名前です。各パラメータ名の前には必ず @ 記号を付け、識別子の規則に従わなければなりません。パラメータの値は、ファンクションの実行時に提供されます。
- **datatype –** パラメータのデータ型です。データ型によっては、データ型名の後に長さをカッコで囲んで指定する必要があります。データ型とその構文の説明は、「データ型」を参照してください。データ型として、*text*、*unitext*、*rawobject*、または *image* を指定することはできません。

例

- **例 1 –** *publishers_rep* 複写定義に、4 つのパラメータを持つ *newpublishers* というユーザ定義ファンクションを作成します。

```
create function publishers_rep.newpublishers
 (@pub_id char(4), @pub_name varchar(40),
 @city varchar(20), @state char(2))
```

使用法

- **create function** は、ユーザ定義ファンクションを作成するときに使用します。
- **create function** は、複写定義を作成した Replication Server で実行します。
- ユーザ定義ファンクションは、非同期プロシージャの配信に使用できます。非同期プロシージャの詳細については、『Replication Server 管理ガイド 第 2 巻』を参照してください。
- パラメータのリストは必ずカッコ **()** で囲んでください。これは、ファンクションにパラメータを指定しないでファンクションを定義する場合も必要です。
- ユーザ定義ファンクションを使用する 3 種類のシステム提供ファンクション文字列クラスと、これらのクラスから継承した各派生クラスに対して、Replication Server はユーザ定義ファンクションのデフォルトのファンクション文字列を生成します。

- *rs_sqlserver_function_class* と、ユーザが作成したファンクション文字列クラスでは、**create function string** を使用してファンクション文字列をカスタマイズできます。
- ユーザ定義ファンクションを使用する、ユーザが作成した各基本ファンクション文字列クラスと、これらのクラスから継承した各派生クラスでは、**create function string** を使用してファンクション文字列を作成します。ファンクション文字列では、レプリケートデータサーバに適した言語を使用して、ストアドプロシージャまたは RPC を呼び出す必要があります。
- ファンクション文字列クラス、ファンクション文字列、ファンクションの概要については、『Replication Server 管理ガイド 第 2 巻』を参照してください。
- Replication Server は、複写システムを介して、新しいユーザ定義ファンクションを条件を満たすサイトに分配します。複写システムで通常の遅延時間が発生するため、変更内容がレプリケートサイトにすぐに反映されるわけではありません。
- ある複写定義に対して 1 つのユーザ定義ファンクションを作成すると、このユーザ定義ファンクションは、プライマリテーブル内のすべての複写定義に対して作成されます。

パーミッション

create function には、“create object” パーミッションが必要です。

参照：

- create applied function replication definition (274 ページ)
- create function string (318 ページ)
- create request function replication definition (362 ページ)
- drop function (410 ページ)

create function replication definition

複写するストアドプロシージャのファンクション複写定義とユーザ定義ファンクションを作成します。

注意： **create function replication definition** コマンドと **alter function replication definition** コマンドは、今後廃止される予定です。 これらの代わりに、次のコマンドを使用することをお奨めします。

- **create applied function replication definition** と **alter applied function replication definition**

- **create request function replication definition** と **alter request function replication definition**。

構文

```
create function replication definition
function_rep_def
with primary at data_server.database
[deliver as 'proc_name']
([@param_name datatype [, @param_name datatype]...])
[searchable parameters (@param_name
 [, @param_name]...)]
[send standby {all | replication definition}
parameters]
```

パラメータ

- **function_rep_def** – ファンクション複写定義の名前です。名前は識別子の規則に従う必要があります。
- **with primary at** – データサーバとプライマリデータを格納するデータベースを指定します。
- **data_server** – プライマリデータのあるデータサーバの名前です。プライマリデータベースがウォームスタンバイアプリケーションの一部である場合、*data_server* は論理データサーバ名になります。
- **database** – プライマリデータのあるデータベースの名前です。プライマリデータベースがウォームスタンバイアプリケーションの一部である場合、*database* は論理データベース名になります。
- **deliver as** – 複写ファンクションを配信するデータベースで実行するストアドプロシージャの名前を指定します。*proc_name* は最大 200 文字の文字列です。この句を指定しない場合、ファンクションはファンクション複写定義と同じ名前のストアドプロシージャとして配信されます。
- **@param_name** – ファンクションからのパラメータ名です。同じパラメータ名を、1 つの句の中で 2 回以上指定することはできません。パラメータとそのデータ型の指定は必須ではありませんが、パラメータを指定するかどうかに関係なく、この句はカッコ **()** で囲んでください。
- **datatype** – ファンクションのパラメータのデータ型です。データ型とその構文のリストについては、「データ型」を参照してください。Adaptive Server のストアドプロシージャとファンクション複写定義には、*text*、*unitext*、*rawobject*、*image* の各データ型のパラメータを含めることはできません。
- **searchable parameters – where** 句 (**define subscription**、**create subscription**、または **create article**) で使用できるパラメータのリストを指定します。この句を含める場合は、パラメータ名をカッコ **()** で囲んでください。

- **send standby** – ウォームスタンバイアプリケーションで、スタンバイデータベースにファンクションのすべてのパラメータを送信するか (**send standby all parameters**)、複写定義で指定されたパラメータだけを送信するか (**send standby replication definition parameters**) を指定します。デフォルトは、**send standby all parameters** です。

例

- **例 1** – 同じ名前のファンクションとストアドプロシージャに対して、*titles_frep* というファンクション複写定義を作成します。プライマリデータは、LDS データサーバの *pubs2* データベースにあります。適用ファンクションには、このようなファンクション複写定義を使用します。

```
create function replication definition titles_frep
 with primary at LDS.pubs2
 (@title_id varchar(6), @title varchar(80),
  @type char(12), @pub_id char(4),
  @price money, @advance money,
  @total_sales int)
  searchable parameters (@title_id, @title)
```

- **例 2** – 前の例と同様に、*titles_frep* というファンクションとストアドプロシージャに対して、同じ名前のファンクション複写定義を作成します。この例の場合、送信先データベースで呼び出されるストアドプロシージャは、*upd_titles* です。要求ファンクションには、このようなファンクション複写定義を使用します。

```
create function replication definition titles_frep
 with primary at LDS.pubs2
 deliver as 'upd_titles'
 (@title_id varchar(6), @title varchar(80),
  @type char(12), @pub_id char(4),
  @price money, @advance money,
  @total_sales int)
  searchable parameters (@title_id, @title)
```

使用法

- **create function replication definition** は、複写するストアドプロシージャを記述するときに使用します。複写ストアドプロシージャの概要については、『Replication Server 管理ガイド 第 1 巻』を参照してください。
- **create function replication definition** は、プライマリデータが格納されているデータベースを管理する Replication Server で実行します。
- 複写ストアドプロシージャごとに、1 つのファンクション複写定義を作成できます。
- このコマンドを実行する前に、次のことを確認してください。

  - ファンクション複写定義の名前が、複写システム内でユニークであること。**create function replication definition** を使用するときに、Replication Server がこの条件を常に要求するわけではありません。
  - Replication Server からプライマリデータが格納されているデータベースへのコネクションが存在していること。詳細については、「**create connection**」を参照してください。**rs_init** を使用してコネクションを作成することもできます。使用しているプラットフォーム用の『Replication Server インストールガイド』と『Replication Server 設定ガイド』を参照してください。
  - ファンクション複写定義に対して指定する名前、パラメータ、データ型が、関連するストアドプロシージャの名前、パラメータ、データ型と一致していること。複写したいパラメータだけを指定できます。
- テーブル複写定義に対応する複写ストアドプロシージャとは異なり、ファンクション複写定義に対応するストアドプロシージャでは、テーブルを更新する必要はありません。そのため、複写データに関連しないトランザクションを複写できます。ストアドプロシージャの詳細については、「RSSD ストアドプロシージャ」を参照してください。
  複写ストアドプロシージャの 2 つのタイプについては、「**sp_setrepproc**」を参照してください。
- Replication Server は、新しいファンクション複写定義を、条件を満たしているサイトへ複写システムを介して分配します。複写システムで通常の遅延時間が発生するため、変更内容がレプリケートサイトにすぐに反映されるわけではありません。

ユーザ定義ファンクションとファンクション文字列

- ファンクション複写定義を作成する場合、Replication Server は、対応するユーザ定義ファンクションを自動的に作成します。
- このファンクション複写定義と対応するユーザ定義ファンクションを使用する、システム提供ファンクション文字列クラスと、これらのクラスから継承した各派生クラスに対して、Replication Server はユーザ定義ファンクションのデフォルトのファンクション文字列を生成します。
- *rs_sqlserver_function_class* と、ユーザが作成したファンクション文字列クラスでは、**create function string** を使用してファンクション文字列をカスタマイズできます。
- ユーザ定義ファンクションを使用する、ユーザが作成した各基本ファンクション文字列クラスと、これらのクラスから継承した各派生クラスでは、**create function string** を使用してファンクション文字列を作成します。ファンクション文字列では、レプリケートデータサーバに適した言語を使用して、ストアドプロシージャまたは RPC を呼び出す必要があります。
- ファンクション文字列クラス、ファンクション文字列、ファンクションの概要については、『Replication Server 管理ガイド 第 2 巻』を参照してください。

with primary at 句

- **with primaryat** 句は、プライマリデータを格納するデータサーバとデータベースを指定するときに使用します。呼び出されるストアドプロシージャを含むデータベースである必要はありません。
  適用ファンクション (プライマリからレプリケートへのファンクションの複写) と要求ファンクション (レプリケートからプライマリへのファンクションの複写) の場合は、プライマリデータを管理する Replication Server でファンクション複写定義を作成し、**with primary at** 句を使用してプライマリデータベースを指定してください。

deliver as 句

- オプションの **deliver as** 句は、複写ファンクションを配信する送信先データベースで実行するストアドプロシージャの名前を指定するために使用します。ファンクション複写定義を作成または変更するときにこの句を指定しない場合、そのファンクションはファンクション複写定義と同じ名前のストアドプロシージャとして配信されます。
  ウォームスタンバイデータベースのストアドプロシージャの名前は、アクティブデータベース内での名前と同じなので、**deliver as** 句は無視されます。
  通常、**deliver as** 句は、要求ファンクションの配信に使用します。つまり、ファンクションがレプリケート Replication Server からプライマリ Replication Server に複写されるときに使用します。この場合、複写されるファンクションの名前は、実行されるストアドプロシージャの名前と同じにはなりません。
  このメソッドは、ストアドプロシージャの「往復」複写で使用します。往復複写では、要求ファンクションの送信先であるプライマリ Replication Server で適用ファンクションが実行され、次に、送信元のレプリケート Replication Server で、この適用ファンクションのサブスクリプションが作成されます。
  詳細については、『Replication Server 管理ガイド 第 1 巻』を参照してください。

HDS パラメータのファンクション複写定義

- パラメータ値のデータ型を変更するファンクション複写定義は作成できませんが、HDS データ型定義を使用して適用ファンクション複写定義のパラメータを宣言できます。このようなパラメータは、その後クラスレベル変換の対象となります。HDS の詳細については、『Replication Server 管理ガイド 第 1 巻』を参照してください。
- Replication Server は、パラメータ値を要求ファンクション用に変換しません。ただし、ファンクション文字列のマップ時に、宣言したデータ型のパラメータ値に対して定義されたデリミタを使用して SQL を生成することに注意してください。

ファンクション複写定義の変更

- パラメータまたはサーチャブルパラメータを既存のファンクション複写定義に追加するには、**alter function replication definition** を使用します。また、送信先

データベースへ複写ファンクションを配信するとき、新しいストアドプロシージャ名を指定できます。
- ファンクション複写定義内のパラメータを削除したり名前を変更したりする必要がある場合は、ファンクション複写定義 (適用ファンクションのみ) へのサブスクリプションをすべて削除する必要があります。その後、ファンクション複写定義を削除して、再度作成してください。

ファンクション複写定義のサブスクリプションの作成

- ファンクション複写定義に対してサブスクリプションを作成するには、**create subscription** を使用するときに **without materialization** 句を指定するか、または **define subscription** とバルクマテリアライゼーション含むその他のコマンドを使用します。

ファンクション複写定義とテーブル複写定義

- 適用ファンクションによってストアドプロシージャを複写する場合は、複写ストアドプロシージャが影響を与える同じテーブルのテーブル複写定義とサブスクリプションを作成することをおすすめします。これにより、テーブルに影響する通常のトランザクションだけでなく、ストアドプロシージャの実行も確実に複写できるようになります。
  複写済みとしてマーク付けされたストアドプロシージャ内の DML は、テーブル複写によって複写されません。テーブルのサブスクリプションを作成している場合でも、ストアドプロシージャのサブスクリプションを作成する必要があります。
- 同じテーブルの 2 種類の複写定義の両方を使用する場合は、テーブル複写定義のサブスクリプションを使用してテーブルデータをマテリアライズします。**create subscription** を使用するときに、**without materialization** 句を指定して、ファンクション複写定義のサブスクリプションを作成します。

パーミッション

create function replication definition には、“create object” パーミッションが必要です。

参照：
- alter function replication definition (185 ページ)
- alter function string (188 ページ)
- create connection (287 ページ)
- create function string (318 ページ)
- define subscription (395 ページ)
- drop function replication definition (411 ページ)
- sp_setrepproc (673 ページ)

create function string

ファンクション文字列クラスにファンクション文字列を追加します。Replication Server は、ファンクション文字列を使用してデータサーバに対する命令を生成します。

構文

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

パラメータ

- **replication_definition –** ファンクションが実行される複写定義の名前です。複写定義スコープを持つファンクションに対してのみ使用します。

  ファンクションは、ファンクション文字列クラススコープ、複写定義スコープ、またはターゲットスコープを持ちます。

  トランザクション制御を指示するファンクションは、ファンクション文字列クラススコープを持ちます。ユーザ定義ファンクションとデータを修正するファンクションは、複写定義スコープを持ちます。

  スタンバイテーブルまたはレプリケートテーブルあるいはストアドプロシージャに対して作成されたファンクション文字列は、ターゲットスコープファンクション文字列になります。
- **[owner.]table –** ファンクション文字列のテーブル所有者とターゲットテーブルを指定します。
- **stored_procedure –** ファンクション文字列のターゲットストアドプロシージャを指定します。
- **function –** ファンクションの名前です。システムファンクションの名前は、「Replication Server システムファンクション」に掲載されている名前でなければ

なりません。ユーザ定義ファンクションの名前は、既存のユーザ定義ファンクションと一致する必要があります。

- **function_string –** ファンクション文字列名は、**rs_get_textptr**、**rs_textptr_init**、および **rs_writetext** の各ファンクションをカスタマイズするときには必須ですが、その他のファンクションでは任意です。**rs_get_textptr**、**rs_textptr_init**、**rs_writetext** では、複写定義の *text*、*unitext*、または *image* の各カラムにファンクション文字列が必要です。指定するファンクション文字列名は、次の要件を満たす必要があります。

  - 複写定義の *text*、*unitext*、または *image* カラム名である。
  - 識別子の規則に従っている。
  - ファンクションのスコープ内でユニークである。

  Replication Server は、エラーメッセージの生成時にもこのファンクション文字列名を使用します。
- **function_class –** 複写定義スコープファンクション文字列について、ファンクション文字列が関連付けられているファンクションクラスを指定します。
- **data_server.database –** ターゲットテーブルまたはストアドプロシージャについて、ターゲットスコープファンクション文字列を作成するスタンバイデータベースまたはレプリケートデータベースを指定します。

  『Replication Server 管理ガイド 第 2 巻』の「ファンクション文字列の作成」を参照してください。
- **with overwrite –** ファンクション文字列がすでに存在する場合、このオプションはファンクション文字列を削除し、代わりに **alter function string** を使用したようにファンクション文字列を再作成します。**with overwrite** オプションは、必ず create function string と組み合わせて使用します。
- **scan –** 入力テンプレートの先頭に付く文字です。
- **input_template –** 一重引用符で囲まれた文字列で、**rs_select** または **rs_select_with_lock** ファンクション文字列と **where** 句 (**create subscription** コマンド内) を関連付けるために Replication Server がスキャンします。入力テンプレート文字列は SQL の **select** 文として書き込まれ、サブスクリプションの **where** 句内のリテラル値の代わりに、ユーザ定義の変数を使用します。
- **output –** 出力テンプレートの先頭に付く文字です。
- **language –** Client/Server Interfaces の言語インタフェースを使用してデータサーバに出力テンプレートコマンドを送信するように、Replication Server に指示します。
- **lang_output_template –** データサーバに対する指示を組み込む文字列で、一重引用符で囲まれます。言語出力テンプレート文字列には、データサーバに送信される前に、その文字列がランタイム値と置き換えられる埋め込み変数が含まれている場合があります。

- **rpc –** Client/Server Interfaces のリモートプロシージャコール (RPC) インタフェースを使用するように Replication Server に指示する出力テンプレートです。Replication Server は文字列を解析し、リモートプロシージャコールを構築して、データサーバに送信します。

  RPC 出力テンプレートには、次のキーワードとオプションがあります。

  procedure - 実行するリモートプロシージャの名前です。これは、Adaptive Server のストアドプロシージャ、Open Server ゲートウェイ RPC ハンドラによって処理されるプロシージャ、または Open Server ゲートウェイのレジスタードプロシージャのいずれかになります。ゲートウェイプログラムでの RPC の処理については、『Open Server Server-Library/C リファレンスマニュアル』を参照してください。

  @*param_name* - プロシージャによって定義されているプロシージャの引数の名前です。@*param_name* = *value* の形式で使用すると、パラーメータは任意の順序で提供できます。パラメータ名を省略する場合は、リモートプロシージャで定義されている順序でパラメータ値を指定する必要があります。

  constant - 割り当てるパラメータのデータ型を持つリテラル値です。

  ?*variable*!*mod*?－ *variable* は、ランタイム値のプレースホルダです。ここには、カラム名、システム定義変数名、ユーザ定義ファンクションのパラメータ名、または入力テンプレートで定義されている変数名を指定できます。変数は、割り当てるパラメータと同じデータ型が指定されている値を参照する必要があります。システム定義変数のリストについては、「システム定義の変数」を参照してください。

  変数名の *mod* の部分は、変数が表すデータ型を示します。変数の変更子はすべての変数に必要であり、次の表に示すいずれかでなければなりません。

表 32 : ファンクション文字列変数の変更子

変更子	説明
new、*new_raw*	挿入または更新するローのカラムの新しい値への参照。
old、*old_raw*	更新または削除するローのカラムの既存値への参照。
user、*user_raw*	**rs_select** ファンクション文字列または **rs_select_with_lock** ファンクション文字列の入力テンプレートに定義されている変数への参照
sys、*sys_raw*	システム定義変数への参照。
param、*param_raw*	ファンクションパラメータへの参照。

変更子	説明
text_status	*text_status* 値 (*text*、*unitext*、または *image* データの) への参照。有効な値は次のとおりです。 • 0x000 - NULL 値を含むテキストフィールド。テキストポインタは初期化されていない。 • 0x0002 - テキストポインタは初期化されている。 • 0x0004 - 実テキストデータが続く。 • 0x0008 - テキストデータが複写されていないので、テキストデータは続かない。 • 0x0010 - テキストデータは複写されていないが、NULL 値を含む。

注意： ユーザ定義ファンクションのファンクション文字列は、new 変更子または old 変更子を使用しない場合があります。

- **writetext –** Client-Library™ 関数の **ct_send_data** を使用して、*text*、*unitext*、または *image* カラム値を更新するように Replication Server に指示します。このオプションは、**rs_writetext** ファンクションにのみ適用されます。

  次のオプションは、レプリケートデータベースの *text*、*unitext*、または *image* カラムのロギング動作を指定するために、**writetext** 出力テンプレートで使用されます。

  use primary log - プライマリデータベースにロギングオプションが指定されている場合、レプリケートデータベースでデータのログを記録する。

  with log - レプリケートデータベースのトランザクションログにデータのログを記録する。

  no log - レプリケートデータベースのトランザクションログにデータのログを取らない。
- **none –** すべての関数に適用され、Replication Server がレプリケートデータベースでの実行を省くことができるファンクション文字列の指定を柔軟に行うことができます。
  - **rs_writetext** ファンクションの場合 - *text*、*unitext*、または *image* カラム値を複写しないように Replication Server に指示します。
  - **rs_writetext** ファンクション以外の場合 - レプリケートデータベース上でコマンドを実行しないように Replication Server に指示します。

例

- **例 1 – rs_begin** ファンクションのファンクション文字列を作成します。

```
create function string rs_begin
 for sqlserver2_function_class
```

```
 output language
 'begin transaction'
```

- **例 2 –** セミコロンで区切られた 2 つのコマンドを含む **rs_commit** ファンクションのファンクション文字列を作成します。このファンクション文字列は、*rs_lastcommit* システムテーブルを更新し、トランザクションをコミットする Adaptive Server ストアドプロシージャを実行します。

```
create function string rs_commit
 for sqlserver2_function_class
 output language
 'execute sqlrs_update_lastcommit
   @origin = ?rs_origin!sys?,
   @origin_qid = ?rs_origin_qid!sys?,
   @secondary_qid = ?rs_secondary_qid!sys?;
   commit transaction'
```

- **例 3 –** 例 3 と例 4 では、*titles* テーブルの複写定義と、*sqlserver2_function_class* の **rs_insert** ファンクション文字列を作成します。このファンクション文字列は、レプリケートデータベースの *titles* テーブルではなく、*titles_rs* テーブルにデータを挿入します。

```
create replication definition titles_rep
 with primary at LDS.pubs2
 (title_id varchar(6), title varchar(80),
  type char(12), pub_id char(4), advance money,
  total_sales int, notes varchar(200),
  pubdate datetime, contract bit, price money)
 primary key (title_id)
 searchable columns (price)
```

- **例 4 –** 例 3 と例 4 では、*titles* テーブルの複写定義と、*sqlserver2_function_class* の **rs_insert** ファンクション文字列を作成します。このファンクション文字列は、レプリケートデータベースの *titles* テーブルではなく、*titles_rs* テーブルにデータを挿入します。

```
create function string titles_rep.rs_insert
 for sqlserver2_function_class
 output language
 'insert titles_rs values (?title_id!new?,
   ?title!new?, ?type!new?, ?pub_id!new?,
   ?advance!new?, ?total_sales!new?, ?notes!new?,
   ?pubdate!new?, ?contract!new?, ?price!new?)'
```

- **例 5 –** 例 5 と例 6 では、ユーザ定義ファンクション **update_titles** と、*sqlserver2_function_class* の対応するファンクション文字列を作成します。このファンクション文字列は、**update_titles** という Adaptive Server ストアドプロシージャを実行します。

```
create function titles_rep.update_titles
 (@title_id varchar(6), title varchar(80),
  @price money)
```

- **例 6 –** 例 5 と例 6 では、ユーザ定義ファンクション **update_titles** と、*sqlserver2_function_class* の対応するファンクション文字列を作成します。このファンクション文字列は、**update_titles** という Adaptive Server ストアドプロシージャを実行します。

```
create function string titles_rep.update_titles
 for sqlserver2_function_class
 output rpc
 'execute update_titles
   @title_id = ?title_id!param?,
   @title = ?title!param?,
   @price = ?price!param?'
```

- **例 7 –** 例 7 の **rs_select** ファンクション文字列は、*title_id* カラムで指定された値を持つローを要求するサブスクリプションをマテリアライズするために使用します。例 8 と同様に、2 つのファンクション文字列は、**scan** 句で指定した入力テンプレートによって区別されます。

```
create function string
   titles_rep.rs_select;title_id_select
 for sqlserver2_function_class
 scan 'select * from titles
   where title_id = ?title_id!user?'
 output language
 'select * from titles
   where title_id = ?title_id!user?'
```

- **例 8 –** 例 8 の **rs_select** ファンクション文字列は、RPC ファンクション文字列の例です。このファンクション文字列は、*price* カラムの値が指定された範囲内にあるローを要求するサブスクリプションをマテリアライズするために使用します。

```
create function string
   titles_rep.rs_select;price_range_select
 for sqlserver2_function_class
 scan 'select * from titles
   where price > ?price_min!user?
   and price < ?price_max!user?'
 output rpc
   'execute titles_price_select
   ?price_min!user?, ?price_max!user?'
```

- **例 9 – upd_datetime** ストアドプロシージャのターゲットスコープファンクション文字列を、データベース `NY_DS.rdb1` について、作成します。

```
create function string upd_datetime.upd_datetime
     for database NY_DS.rdb1
     with overwrite
          output language
          'update datetime set
          row_num = ?row_num!param?,
          datecol = ?datecol!param?,
          timecol = ?timecol!param?,
          ndatecol = ?ndatecol!param?,
```

```
          ntimecol = ?ntimecol!param?,
          comment = ?comment!param?
          where
          row_num = ?row_num!param?'
```

- **例 10 –** dbo.datetime テーブルのターゲットスコープファンクション文字列を、NY_DS.rdb1 について作成します。

```
create function string dbo.datetime.rs_insert
     for database NY_DS.rdb1
     with overwrite
     output language
     'insert datetime values (
             ?row_num!new? ,
             ?datecol!new? ,
             ?timecol!new? ,
             ?ndatecol!new? ,
             ?ntimecol!new? ,
             ?comment!new?)
       update fn_monitor set insert_count = insert_count + 1'
```

- **例 11 –** dbo.tbl1.unitext_fd1 カラムについて、rs_writetext のカスタマイズされたファンクション文字列を作成します。

```
create function string dbo.tbl1.rs_writetext; unitext_fld1 for
NY_DS.rdb1
         output RPC
        'exec update_repl_unitext
                @p_key       = ?p_key!new?,
                @unitext_fld       = ?unitext_fld1!new?,
                @last_chunk         = ?rs_last_text_chunk!sys?'
```

- **例 12 –** dbo.tbl1 テーブルのターゲットスコープファンクション文字列を作成します。

```
create function string dbo.tbl1.rs_datarow_for_writetext
for NY_DS.rdb1
        output RPC
        'exec update_txtimg_stat
                @p_key       = ?p_key!new?,
                @txtfld_stat = ?unitext_fld1!text_status?'
```

- **例 13 – rs_insert** のカスタマイズされたファンクション文字列 (この文字列の dbo.authors テーブルは、NY_DS データサーバの rdb1 ターゲットデータベースにあります) を作成します。

```
create function string dbo.authors.rs_insert
     for database NY_DS.rdb1
     output language
     'insert authors values (
           ?au_id!new? ,
           ?au_lname!new? ,
           ?au_fname!new? ,
           ?phone!new? ,
           ?address!new? ,
           ?city!new? ,
```

```
            ?state!new? ,
            "00000" ,
            ?contract!new?)
      update fn_monitor set insert_count = insert_count + 1'
```

- **例 14 –** カスタマイズされたファンクション文字列 (この **upd_bits** ストアドプロシージャは、NY_DS データサーバの `rdb1` ターゲットデータベースにあります) を作成します。ストアドプロシージャのファンクションの名前はストアドプロシージャと同じになります。

```
create function string upd_bits.upd_bits
     for database NY_DS.rdb1
     with overwrite
     output language
     'exec upd_bits
           @firstbit = ?firstbit!param?,
           @secondbit = ?secondbit!param?,
     @commit = ?comment!param?'
```

使用法

- **create function string** は、ファンクション文字列をファンクション文字列クラスに追加するときに使用します。ファンクション文字列には、Replication Server がファンクションをデータベースのコマンドに変換するために必要なデータベース固有の命令が組み込まれます。
- ファンクション、ファンクション文字列、ファンクション文字列クラスの概要については、『Replication Server 管理ガイド 第 2 巻』を参照してください。
- ターゲットデータベース (スタンバイデータベースまたはレプリケートデータベース) を制御する Replication Server にあるターゲットスコープファンクション文字列に対し、**create function string** を実行します。
- **with overwrite** オプションは、必ず **create function string** と組み合わせて使用します。
- 複写定義スコープファンクション文字列はファンクションクラスに関連付けられ、ターゲットスコープファンクション文字列はターゲットデータベースに関連付けられます。
- ターゲットテーブルに所有者情報が含まれておらず、コマンドのファンクション文字列で **function class** オプションおよび **database** オプションを指定していない場合、Replication Server では、**for** キーワードの後の文字列の形式がファンクションクラスまたはデータベースの形式であるかどうかを確認して、ファンクション文字列が複写定義またはテーブルのどちらに関する文字列であるかについてのみ認識します。状況に応じて次のようになります。
  - 複写定義スコープファンクション文字列 - **rs_sqlserver_function_class** など、**for** キーワードの後の文字列形式にはデータベース名は含まれません。

    - ターゲットスコープファンクション文字列 - `NY_DS.rdb1` など、**for** キーワードの後の文字列形式にはデータサーバとデータベースの名前が含まれます。
- ターゲットスコープファンクション文字列は、複数の複写パスについて設定されている可能性があるコネクションに対してではなく、スタンバイデータベースまたはレプリケートデータベースに対してのみ作成できます。
- ウォームスタンバイ環境では、影響を受けるデータベースは物理データベースです。論理データベースについてターゲットスコープファンクション文字列を定義する場合、アクティブデータベースとスタンバイデータベースの両方に対してファンクション文字列コマンドを発行する必要があります。
- ファンクション名は、スタンバイストアドプロシージャまたは複写ストアドプロシージャのターゲットスコープファンクション文字列のストアドプロシージャ名と同じになります。
- スタンバイテーブルまたはレプリケートテーブルのターゲットスコープファンクション文字列では、有効なファンクションは、**rs_insert**、**rs_update**、**rs_delete**、**rs_truncate**、**rs_writetext**、**rs_datarow_for_writetext**、**rs_textptr_init**、および **rs_get_textptr** です。
- オブジェクトの複写定義が存在しない場合、またはオブジェクトのすべての複写定義がオブジェクトによって使用されない場合、Replication Server ではターゲットスコープファンクション文字列のみを使用します。
  『Replication Server 管理ガイド 第 1 巻』の「ウォームスタンバイ環境と Muliti-Site Availability 環境」を参照してください。
- クラススコープを持つファンクションのファンクション文字列は、ファンクション文字列クラスのプライマリサイトで作成または変更します。ファンクション文字列クラスのプライマリサイトの詳細については、「**create function string class**」を参照してください。
- ユーザ定義ファンクションなどの複写定義スコープを持つファンクションのファンクション文字列は、複写定義が作成されたサイトで作成または変更します。複写定義には、それぞれ固有のファンクション文字列セットがあります。
- Replication Server は、新しいファンクション文字列を、条件を満たしているサイトへ複写システムを介して分配します。複写システムで通常の遅延時間が発生するため、変更内容がレプリケートサイトにすぐに反映されるわけではありません。
- 一部のファンクション文字列は動的に生成されます。これらは RSSD に格納されません。

ファンクション文字列とファンクション文字列クラス

- ファンクションを使用する各システム提供ファンクション文字列クラスと、これらのクラスから継承した各派生クラスに対して、Replication Server はファクションのデフォルトのファンクション文字列を生成します。これは、システムファンクションとユーザ定義ファンクションの両方に言えることです。
  rs_dumpdb および **rs_dumptran** ファンクションには、デフォルトのファンク

ション文字列は提供されていません。コーディネートダンプを使用する場合にのみ、デフォルトのファンクション文字列を作成する必要があります。

- **alter function string** を使用して、*rs_sqlserver_function_class* のファンクション文字列をカスタマイズします。ユーザが作成したファンクション文字列クラスのファンクション文字列をカスタマイズする場合は、**create function string** を使用します。
- ファンクションを使用する、ユーザが作成した各基本ファンクション文字列クラスと、継承ファンクション文字列を上書きする各派生クラスでは、**create function string** を使用してファンクション文字列を作成します。
- **output** 句を省略すると、*rs_sqlserver_function_class* ファンクション文字列クラスまたは *rs_default_function_class* ファンクション文字列クラスのファンクション文字列を生成する場合と同様にファンクション文字列を生成するように、Replication Server に指示します。
- ユーザ定義ファンクションのデフォルトのファンクション文字列は、名前がファンクション名で、パラメータがファンクションパラメータであるストアドプロシージャを呼び出します。ストアドプロシージャは、RPC としてではなく、言語コマンドとして実行されます。

注意： ExpressConnect for Oracle は、text 型と image 型を処理するためのカスタムファンクション文字列の使用をサポートしていません。

『Replication Server 異機種間複写ガイド』の「ExpressConnect の設定」と「ファンクション文字列、エラークラス、ユーザ定義データ型」を参照してください。

ファンクション文字列と replicate minimal columns

- 複写定義に **replicate minimal columns** を指定した場合、通常 **rs_update**、**rs_delete**、**rs_get_textptr**、**rs_textptr_init**、または **rs_datarow_for_writetext** システムファンクションにデフォルト以外のファンクション文字列を作成できません。
  ただし、ファンクション文字列内で *rs_default_fs* システム変数を使用すると、**rs_update** ファンクションと **rs_delete** ファンクションのデフォルト以外のファンクション文字列を作成できます。この変数は、デフォルトのファンクション文字列の動作を表します。コマンドを追加して、ファンクション文字列の動作を拡張することもできます。
- 「**create replication definiton**」では、**replicate minimal columns** オプションの詳細について説明しています。

入力テンプレートと出力テンプレート

- ファンクションによっては、ファンクション文字列に入力テンプレートと出力テンプレートがあります。Replication Server は、テンプレートに変数値を代入してから、結果を処理するためにデータサーバに渡します。

- 入力テンプレートと出力テンプレートには、次の要件があります。
  - テンプレートのサイズは 64K に制限される。ファンクション文字列の入力テンプレートまたは出力テンプレート内の埋め込み変数にランタイム値を代入した結果が 64K を超えてはならない。
  - 入力テンプレートと言語、または RPC 出力テンプレートは、2 つの一重引用符 (') で区切る。
  - 入力テンプレートと出力テンプレート内の変数名は、疑問符 (?) で区切る。
  - 変数名とその変更子は感嘆符 (!) で区切る。
- ファンクション文字列を作成する場合は、次のようにします。
  - 文字データ型または日付/時刻データ型のデータ内またはデータを囲む 1 つのリテラル一重引用符を表すには、連続する 2 つの一重引用符 ('') を使用します。たとえば、次の文字列内の “Berkeley” のようになります。

    ```
    'insert authors
    (city, au_id, au_lname, au_fname)
    values (''Berkeley'', ?au_id!new?,
    ?au_lname!new?,
    ?au_fname!new?)'
    ```

  - 文字データ型のデータ内で 1 つの疑問符を表すには、連続する 2 つの疑問符 (??) を使用します。
  - 2 つの連続したセミコロン (;;) は、文字データ型のデータ内で 1 つのセミコロンを表すために使用します。
  - 引用符付き定数が含まれるカスタムファンクション文字列とともに引用符付き識別子を使用する場合は、引用符付き定数なしで、または **without materialization** 句を指定せずに **create subscription** を実行します。 それ以外の場合、サブスクリプションのマテリアライゼーション時に引用符付き定数が原因でクエリが失敗します。 レプリケートデータサーバは、引用符付き定数を定数ではなくカラムとして認識します。

入力テンプレート

- 入力テンプレートは、**rs_select** ファンクションと **rs_select_with_lock** ファンクションでのみ使用されます。これらのファンクションは、非バルクサブスクリプションマテリアライゼーションおよび **with purge** サブスクリプションマテリアライゼーション解除の実行時に使用されます。Replication Server は、サブスクリプションの where 句と入力テンプレートを照合し、使用するファンクション文字列を検出します。
- 入力テンプレートには、次の動作条件があります。
  - 入力テンプレートには、where 句内の定数から取得した値を持つユーザ定義変数だけを組み込む。ユーザ定義変数は、ファンクション文字列の出力テンプレート内でも参照できる。
  - *input_template* を省略すると、任意の **select** コマンドと一致させることができる。これにより、ファンクション文字列クラスのファンクション文字列

の *input_template* が **select** コマンドと一致しない場合に実行される、デフォルトのファンクション文字列を作成できる。

出力テンプレート

- 出力テンプレートにより、レプリケートデータサーバに送信されるコマンドのフォーマットが決まります。ほとんどの出力テンプレートでは、言語、RPC または none のフォーマットを使用できます。**rs_writetext** ファンクション文字列の出力テンプレートでは、RPC フォーマット、または他のフォーマットとして **writetext** または none を使用できます。これらのフォーマットについては、『Replication Server 管理ガイド 第 2 巻』を参照してください。
- Replication Server は、ファンクション文字列の出力テンプレートをデータサーバコマンドにマップするときに、Adaptive Server で必要とされるフォーマットを使用して変数をフォーマットします。Replication Server は、末尾に *_raw*(通常使用される変更子) が付かない変更子のデータ型を、次のように変更します。
  - 文字と日付／時刻の値に含まれる一重引用符にさらに一重引用符を 1 つ追加し、一重引用符の特殊な意味をエスケープする。
  - 文字と日付／時刻の値に一重引用符がない場合は、その前後に一重引用符を追加する。
  - 通貨データ型の値に適切な通貨記号 (英語の場合はドル記号) を追加する。
  - バイナリデータ型の値に “0x” プレフィクスを追加する。
  - 円記号 (¥) と改行文字を組み合わせたものを、文字値内の円記号と改行文字の既存のインスタンス間に追加する。Adaptive Server は、改行文字が後ろに付いた円記号をひと続きの文字とみなすため、元の文字をそのまま残し、追加された一対の文字を削除する。
    Replication Server は、末尾に *_raw* が付いた変更子に対して、このようなデータ型の変更は行いません。
    ファンクション文字列変数のフォーマットの表は、Replication Server での、末尾に *_raw* が付かない変更子の各データ型のフォーマット方法を要約したものです。

表 33 : ファンクション文字列変数のフォーマット

データ型	リテラルのフォーマット
`bigint、int、smallint、tinyint、rs_address`	整数値
`unsigned bigint、unsigned int、unsigned smallint、unsigned tinyint`	符号なし整数値

データ型	リテラルのフォーマット
decimal、numeric、identity	真数値 (10 進数)
float、real	10 進数
char、varchar	一重引用符文字で囲まれる。 一重引用符文字の任意のインスタンスに一重引用符文字を追加する。 円記号と改行文字のインスタンスを埋め込む。
unichar、univarchar	Unicode
money、smallmoney	適切な通貨記号 (英語の場合はドル記号) を追加する。
date、time、datetime、smalldatetime	一重引用符文字で囲まれる。 一重引用符文字の任意のインスタンスに一重引用符文字を追加する。
binary、timestamp、varbinary	0x で始まる。
bit	1 または 0。

- 出力テンプレートには、次の要件があります。
  - ファンクション文字列の出力テンプレート内の埋め込み変数にランタイム値を代入した結果が 64K を超えてはならない。
  - セミコロン (;) で区切ることによって、言語ファンクション文字列の出力テンプレートに複数のコマンドを入力できる。データベースがコマンドバッチを使用できるように設定されている場合 (デフォルトの設定) は、Replication Server は、このセミコロンをそのコネクションの DSI コマンドセパレータ文字に置き換えてから、単一のバッチ内のファンクション文字列としてデータサーバに送信する。セパレータ文字は、**dsi_cmd_separator** オプション (**alter connection** コマンド) に定義されている。
    セミコロンを、コマンドセパレータとして変換されないようにするには、2 つの連続したセミコロン (;;) を使用します。
    データベースへのコネクションがバッチを使用できるように設定されていない場合は、Replication Server は、ファンクション文字列内のコマンドを一度に 1 つずつデータサーバに送信します。データベースのバッチを有効または無効にするには、alter connection を使用します。

Replication Server のシステム定義変数の表に、ファンクション文字列の出力テンプレートで使用できるシステム定義変数を示します。これらの変数には、*sys* 変更子または *sys_raw* 変更子を使用します。

表 34 : Replication Server のシステム定義変数

システム変数	データ型	説明
rs_default_fs	`text`	ファンクションに対してデフォルトで生成されるファンクション文字列テキスト。
rs_deliver_as_name	`varchar`(200)	複写ファンクションを実行する場合は、送信先で呼び出されるプロシージャの名前。
rs_destination_db	`varchar`(30)	トランザクションが送信されたデータベースの名前。
rs_destination_ds	`varchar`(30)	トランザクションが送信されたデータサーバの名前。
rs_destination_ldb	`varchar`(30)	トランザクションが送信された論理データベースの名前。
rs_destination_lds	`varchar`(30)	トランザクションが送信された論理データサーバの名前。
rs_destination_ptype	`char`(1)	トランザクションが送信されたデータベースの物理コネクションタイプ (“A” はアクティブ、“S” はスタンバイを示す)。
rs_destination_user	`varchar`(30)	送信先でトランザクションを実行するユーザ。
rs_dump_dbname	`varchar`(30)	データベースダンプまたはトランザクションダンプが開始されるデータベースの名前。
rs_dump_label	`varchar`(30)	データベースダンプまたはトランザクションダンプのラベル情報。Adaptive Server では、この変数にはダンプが開始された時間を示す `datetime` 値が格納される。
rs_dump_status	`int`(4)	ダンプステータスインジケータ： • 0 - ダンプトランザクションコマンドに **with standby_access** パラメータが含まれないことを示す。 • 1 - ダンプトランザクションコマンドに **with standby_access** パラメータが含まれることを示す。
rs_dump_timestamp	`varbinary`(16)	データベースダンプまたはトランザクションダンプのタイムスタンプ。
rs_lorigin	`int`(4)	トランザクションが開始される論理データベースの ID。

システム変数	データ型	説明
rs_isolation_level	varchar(30)	データベースコネクションのトランザクションの独立性レベル。
rs_origin	int(4)	トランザクションが開始されるデータベースのID。
rs_origin_begin_time	datetime	オリジンでコマンドが適用された時間。 **注意：** ASE がまだユーザデータベースのリカバリを処理している間に **select getdate()** を実行した場合、**select getdate()** の戻り値が、rs_origin_begin_time の値と異なる可能性があります。
rs_origin_commit_time	datetime	オリジンでトランザクションがコミットされた時間。 **注意：** ASE がまだユーザデータベースのリカバリを処理している間に **select getdate()** を実行した場合、**select getdate()** の戻り値が、rs_origin_begin_time の値と異なる可能性があります。
rs_origin_db	varchar(30)	オリジンデータベースの名前。
rs_origin_ds	varchar(30)	オリジンデータサーバの名前。
rs_origin_ldb	varchar(30)	ウォームスタンバイアプリケーションの論理データベースの名前。
rs_origin_lds	varchar(30)	ウォームスタンバイアプリケーションの論理データサーバの名前。
rs_origin_qid	varbinary(36)	トランザクションにある最初のコマンドのオリジンキュー ID。
rs_origin_user	varchar(30)	オリジンでトランザクションを実行したユーザ。
rs_origin_xact_id	binary(120)	トランザクションに対してシステムが割り当てたユニーク ID。
rs_origin_xact_name	varchar(30)	オリジンにあるトランザクションに対してユーザが割り当てた名前。
rs_repl_objowner	varchar	複写オブジェクトの所有者
rs_secondary_qid	varbinary(36)	サブスクリプションマテリアライゼーションキューまたはマテリアライゼーション解除キューにあるトランザクションのキュー ID。

システム変数	データ型	説明
rs_last_text_chunk	int(4)	値が 0 の場合、text データの最後のまとまりではないことを示す。値が 1 の場合、text データの最後のまとまりであることを示す。
rs_writetext_log	int(4)	値が 0 の場合、rs_writetext が text、unitext、image データをプライマリデータベースのトランザクションログに記録し終わっていないことを示す。値が 1 の場合は、rs_writetext が text、unitext、image データをプライマリデータベースのトランザクションログに記録し終わっていることを示す。

ラージトランザクションのコミットが DSI キューから読み込まれる前に、並列 DSI を使用しないでこれらのトランザクションを処理すると、*rs_origin_commit_time* システム変数の値は、トランザクショングループの最後のトランザクションがプライマリサイトでコミットされた時間に設定されます。

ラージトランザクションのコミットが DSI キューから読み込まれる前に、並列 DSI を使用してこれらのトランザクションを処理すると、DSI スレッドがこれらのトランザクションの 1 つの処理を開始するときに、*rs_origin_commit_time* システム変数の値は *rs_origin_begin_time* システム変数の値に設定されます。

トランザクションのコミット文が読み込まれると、*rs_origin_commit_time* の値は実際のコミット時間に設定されます。そのため、**dsi_num_large_xact_threads** 設定パラメータが 0 より大きな値に設定されていると、*rs_origin_commit_time* の値は、rs_commit 以外のシステムファンクションにおいて、信頼できる値にはなりません。

システム変数と NULL 値

- 次のシステム変数は、NULL 値を持つことができます。
  - *rs_origin_ds*
  - *rs_origin_db*
  - *rs_origin_user*
  - *rs_origin_xact_name*
  - *rs_destination_db*
  - *rs_destination_user*
  - *rs_dump_dbname*
  - *rs_dump_label*

  システム変数に値がない場合、Replication Server は “NULL” という文字をファンクション文字列テンプレートにマップします。これは、生成される一部の文

で構文エラーの原因になることがあります。たとえば、*rs_origin_xact_name* に null 値が含まれていると、次のコマンドが生成されます。

```
begin transaction NULL
```

このエラーを防ぐには、出力テンプレートのファンクション文字列を次のように作成します。

```
'begin transaction t_?rs_origin_xact_name!sys_raw?'
```

rs_origin_xact_name システム変数が null の場合、トランザクション名は “t_NULL” になります。

ファンクション文字列の置き換え

- ファンクション文字列を置き換えるには、**alter function string** を使用するか、**overwrite** を指定して **create function string** を使用します。どちらの方法も、1 つのトランザクション内で **drop function string** と **create function string** を実行します。これは、一時的にファンクション文字列が見つからないことによって発生するエラーを防ぎます。

パーミッション

create function string には、“create object” パーミッションが必要です。

参照：

- alter function string (188 ページ)
- configure connection (238 ページ)
- create connection (287 ページ)
- create function string class (334 ページ)
- create subscription (376 ページ)
- define subscription (395 ページ)
- drop function string (412 ページ)

create function string class

ファンクション文字列クラスを作成します。

構文

```
create function string class function_class
    [set parent to parent_class]
```

パラメータ

- **function_class –** 作成するファンクション文字列クラスの名前です。名前は識別子の規則に従う必要があります。ファンクション文字列クラス名はグローバルなネームスペースを持つため、複写システム内でユニークである必要があります。
- **set parent to –** 新しい派生クラスの親クラスを指定します。
- **parent_class –** 新しい派生クラスの親クラスとして指定する既存のファンクション文字列クラスの名前です。**rs_sqlserver_function_class** を親クラスとして使用することはできません。

例

- **例 1 –** *sqlserver_derived_class* という派生ファンクション文字列クラスを作成します。このクラスは、システム提供クラス *rs_default_function_class* からファンクション文字列を継承します。

```
create function string class
 sqlserver_derived_class
 set parent to rs_default_function_class
```

- **例 2 –** *sqlserver2_function_class* というファンクション文字列クラスを作成します。このクラスは基本クラスになり、ファンクション文字列は継承しません。ただし、このクラスは派生クラスの親クラスとして指定できます。

```
create function string class sqlserver2_function_class
```

使用法

- **create function string class** はファンクション文字列クラスを作成するために使用します。ファンクション文字列クラスは、ファンクション文字列をデータベースごとにグループ化します。ファンクション文字列クラスは、そのメンバファンクション文字列とともに、データベースに対応付けられます。この対応付けは、**create connection** または **alter connection** コマンドを使用して行われます。
- **create function string class** の送信先の Replication Server は、新しく作成されたファンクション文字列クラスのプライマリ Replication Server になります。
- **set parent to** 句を使用して新しい派生クラスを作成し、新しいクラスがファンクション文字列を継承する親クラスを指定します。親クラスからファンクション文字列を継承しない新しい基本クラスを作成するには、この句を省略します。
- ファンクション文字列クラス、ファンクション文字列、ファンクションの概要については、『Replication Server 管理ガイド 第 2 巻』を参照してください。

- このコマンドを実行する前に、新しいファンクション文字列クラスの名前が複写システム内でユニークであることを確認してください。Replication Server は、名前の競合をすべて検出するわけではありません。
- Replication Server は、複写システムを介して、新しいファンクション文字列クラスを条件を満たすサイトに分配します。複写システムで通常の遅延時間が発生するため、変更内容がレプリケートサイトにすぐに反映されるわけではありません。
- クラス *rs_sqlserver_function_class* 内のファンクション文字列を修正するには、最初にクラスのプライマリサイトとなる Replication Server を選択してください。次に、そのサイトで *rs_sqlserver_function_class* に対して **create function string class** を実行します。
- ファンクション文字列クラスのプライマリサイトとして機能する Replication Server は、そのクラスを使用する他のすべての Replication Server へのルートを持つ必要があります。
- 派生クラスのプライマリサイトは、その親クラスのプライマリサイトと同じです。派生クラスは、その親クラスのプライマリサイトで作成しなければなりません。ただし、親クラスがシステム提供クラス *rs_default_function_class* または *rs_db2_function_class* である場合、派生クラスのプライマリサイトは、その派生クラスを作成した Replication Server になります。

システム提供ファンクション文字列クラス

- Replication Server には、3 種類のファンクション文字列クラスが用意されています。これらのクラスは次のように使用できます。
  - *rs_sqlserver_function_class* - このクラスには、デフォルトで生成された Adaptive Server のファンクション文字列が提供されます。*rs_sqlserver_function_class* と *rs_default_function_class* のデフォルトのファンクション文字列は同じです。このクラスは、**rs_init** を使用して複写システムに追加した Adaptive Server データベースにデフォルトで割り当てられます。このクラスのファンクション文字列はカスタマイズできます。*rs_sqlserver_function_class* は、親クラスとしても派生クラスとしても使用できません。
  - *rs_default_function_class* - このクラスには、デフォルトで生成された Adaptive Server のファンクション文字列が提供されます。*rs_sqlserver_function_class* と *rs_default_function_class* のデフォルトのファンクション文字列は同じです。このクラスのファンクション文字列はカスタマイズできません。このクラスは親クラスとして使用できますが、派生クラスになることはできません。
  - *rs_db2_function_class* - このクラスには、デフォルトで生成された DB2 固有のファンクション文字列が提供されます。このクラスは DB2 用にカスタマイズされた *rs_default_function_class* の派生クラスですが、このクラスのファ

ンクション文字列はカスタマイズできません。*rs_db2_function_class* は親クラスとして使用できますが、派生クラスになることはできません。

ファンクション文字列の継承の利点

- システム提供クラス *rs_default_function_class* または *rs_db2_function_class* から直接または間接的に継承した派生クラスを使用すると、新しいテーブル複写定義またはファンクション複写定義の場合でも、カスタマイズしたいファンクション文字列だけをカスタマイズし、他はすべて継承することができます。
  システム提供クラスから継承しないクラスを使用する場合は、親クラスまたは派生クラスのいずれかに、すべてのファンクション文字列を自分で作成する必要があります。また、新しいテーブルまたはファンクション複写定義を作成する場合は、必ず新しいファンクション文字列を追加してください。
- 今後、Replication Server を最新のリリースにアップグレードすると、システム提供クラス *rs_default_function_class* または *rs_db2_function_class* から直接または間接的に継承した派生クラスは、すべての新しいシステムファンクションのファンクション文字列定義を継承することになります。

ファンクション文字列クラスへのファンクション文字列の追加

- 親クラスからファンクション文字列を継承しないファンクション文字列クラスを作成したら、ファンクション文字列クラススコープを持つシステムファンクションのファンクション文字列を追加します。次に、複写定義スコープを持ち、新しいファンクション文字列クラスを使用するデータベースに複写されるシステムファンクションとユーザ定義ファンクションのファンクション文字列を追加します。
- ファンクション文字列クラスでファンクション文字列を作成またはカスタマイズするには、**create function string** を使用します。*rs_default_function_class* クラスまたは *rs_db2_function_class* クラスでは、ファンクション文字列は作成できません。

パーミッション

create function string class には、“sa” パーミッションが必要です。

参照：

- alter connection (134 ページ)
- alter function string class (190 ページ)
- create connection (287 ページ)
- create function (310 ページ)
- create function string class (334 ページ)
- move primary (432 ページ)

create logical connection

論理コネクションを作成します。Replication Server は、論理コネクションを使用してウォームスタンバイアプリケーションを管理します。

構文

```
create logical connection to data_server.database
[set logical_database_param [to] 'value'
[set logical_database_param [to] 'value']...]
```

パラメータ

- **data_server –** データサーバの名前です。このデータサーバは、実際のデータサーバである必要はありません。
- **database –** データベースの名前です。このデータベースは、実際のデータベースである必要はありません。
- **logical_database_param –** 論理コネクションに影響を与える設定パラメータの名前です。「表 19 : 論理コネクションに影響を与える設定パラメータ」に、**create logical connection** で設定できるパラメータを示します。

例

- **例 1 –** *LDS.logical_pubs2* という論理コネクションを作成します。

  ```
  create logical connection to LDS.logical_pubs2
  ```

- **例 2 –** 既存のコネクションの論理コネクションを作成します。たとえば、すでに存在しているデータベース *TOKYO_DS.pubs2* を、ウォームスタンバイアプリケーションでアクティブデータベースとして機能させる場合は、次のコマンドを入力します。

  ```
  create logical connection to TOKYO_DS.pubs2
  ```

使用法

- **create logical connection** は、ウォームスタンバイアプリケーションで使用される論理コネクションを作成します。ウォームスタンバイアプリケーションの設定と管理については、『Replication Server 管理ガイド 第 2 巻』を参照してください。
- 論理コネクションは、シンボリック *data_server.database* を指定するためのものです。データサーバとデータベースは実際のものである必要はありません。Replication Server がそれらを現在のアクティブデータベースにマップします。
- 既存のコネクションの論理コネクションを作成する場合、*data_server.database* は既存のコネクションのデータサーバ名とデータベース名を参照する必要があ

ります。もしくは、論理名をアクティブデータベース名やスタンバイデータベース名とは別の名前にすることをおすすめします。
- 複写定義とサブスクリプションには、論理コネクション名を使用します。
- 論理コネクションを作成した後に論理コネクションの物理アクティブデータベースや物理スタンバイデータベースを追加するには、**rs_init** を使用してください。

パーミッション

create logical connection には、“sa” パーミッションが必要です。

参照：
- alter logical connection (191 ページ)
- configure connection (238 ページ)
- configure logical connection (238 ページ)
- drop connection (407 ページ)
- drop logical connection (416 ページ)
- switch active (456 ページ)
- create alternate logical connection (272 ページ)

create partition

Replication Server でパーティションを使用できるようにします。パーティションには、ディスクパーティションまたはオペレーティングシステムファイルを使用できます。

構文

```
create partition logical_name
on 'physical_name' with size size
[starting at vstart]
```

パラメータ

- **logical_name –** パーティションの名前です。名前は識別子の規則に従う必要があります。この名前は、**drop partition** コマンドと **alter partition** コマンドでも使用されます。
- **physical_name –** パーティションの完全な指定 (フルパス) です。この名前は一重引用符で囲みます。
- **size –** パーティションのサイズ (メガバイト単位) です。指定できる最大サイズは 1TB です。

- **starting at vstart –** パーティションの先頭からのオフセットをメガバイト数 (*vstart*) で指定します。

例

- **例 1 –** `/dev/rsd0a` というデバイスに、*P1* という 20MB のパーティションを追加します。

```
create partition P1 on '/dev/rsd0a' with size 20
```

- **例 2 –** `/dev/rsd0a` というデバイスに、*P1* という 20MB のパーティションを追加します。ただし、1MB のオフセットが指定されているため、Replication Server で使用できるパーティション領域の合計は 19MB になります。

```
create partition P1 on '/dev/rsd0a' with size 20
 starting at 1
```

使用法

- Replication Server は、パーティションをステーブルメッセージキューに使用します。送信されるまでの間、データはメッセージキューに保持されます。
- パーティションで使用可能なディスク領域を増やすと、Replication Server がサポートできるルートやデータベースコネクションが増加し、障害が長引いた場合もメッセージのキューイングを続行できるようになります。
- パーティションの最大サイズは 1TB (約 1,000,000MB) です。
- オペレーティングシステムで使用するために、ディスクパーティションをマウントしないでください。また、スワップ領域や Adaptive Server のディスクデバイスなど、他の目的で使用することはできません。
- Replication Server にはパーティション全体を割り付けてください。パーティションの一部だけを Replication Server に割り付けても、残りの部分を他の目的に使用することはできません。**starting at***vstart* 句を指定する場合、Replication Server で使用できるパーティション領域は、パーティションの合計サイズからオフセットサイズを引いた残りになります。
- **starting at***vstart* 句は、ディスクミラーリング情報用に、パーティションの最初にスペースを作成します。
- パーティション用にオペレーティングシステムファイルを使用できます。 しかし、オペレーティングシステムのファイル I/O バッファリングにより、障害が発生したときにステーブルキューを完全にリカバリできない可能性があります。 この潜在的な問題を回避するには、パーティションでオペレーティングシステムファイルを使用するときに、**configure replication server** コマンドで **sqm_write_flush** を on または dio に設定します。

  create partition を使用して Microsoft Windows プラットフォーム上にディレクトリレベルのパーティションを作成します。

```
c:¥dir_x¥partX.dat, or d:¥dir_x¥partX.dat, or e:¥dir_x¥partX.dat
```

ここで、c:¥ はディスク名、dir_x はディレクトリ、partX.dat はパーティション名です。

注意： Microsoft Windows プラットフォームでは、ローデバイスを使用してパーティションを作成することはできません。

- Replication Server のディスクパーティションがローデバイスの場合、**rs_init** で [ディスクパーティション情報] ウィンドウの各項目に情報を入力するには、そのディスクパーティションが存在している必要があります。 事前にパーティションを定義していない場合は、『Replication Server 設定ガイド』の「ディスクパーティションの作成」の手順に従います。 ただし、ディスクパーティションがオペレーティングシステムファイルの場合は、ディスクパーティションが存在しなければ、Replication Server によって自動的にパーティションが作成されます。
- ディスクパーティションまたはオペレーティングシステムファイルは、“sybase” ユーザが所有するようにします。このユーザには、パーティションに対する読み込み／書き込みパーミッションが必要です。“sybase” 以外のユーザには、このパーティションに対する読み込み／書き込みパーミッションを付与しないでください。

パーミッション

create partition には、“sa” パーミッションが必要です。

参照：

- admin disk_space (57 ページ)
- drop partition (417 ページ)
- alter partition (196 ページ)

create publication

サブスクリプションを作成する 1 つ以上のレプリケートデータベースに 1 つのグループとして複写される、テーブルまたはストアドプロシージャのパブリケーションを作成します。

構文

```
create publication pub_name
    with primary at data_server.database
```

パラメータ

- **pub_name –** パブリケーションの名前です。識別子の規則に従い、指定したプライマリデータサーバとプライマリデータベースでユニークでなければなりません。
- **with primary at data_server.database –** プライマリデータのロケーションを指定します。プライマリデータベースがウォームスタンバイアプリケーションの一部である場合、*data_server.database* は論理データサーバと論理データベースの名前になります。

例

- **例 1 –** *pubs2* データベース内の複数のテーブルとストアドプロシージャのデータを複写するために使用できる、*pubs2_pub* というパブリケーションを作成します。

```
create publication pubs2_pub
 with primary at TOKYO_DS.pubs2
```

使用法

- **create publication** は、パブリケーションを作成するために使用します。パブリケーションは、データベース内の複数のテーブルまたはストアドプロシージャの複写を簡単に設定できるようにするオブジェクトです。パブリケーションを作成し、複写定義を指定するアーティクルを追加して、次にそのパブリケーションに単一のサブスクリプションを作成します。
- プライマリデータが格納されているデータベースを管理する Replication Server で、**create publication** コマンドを実行してください。
- 複写定義、アーティクル、パブリケーションの処理の詳細については、『Replication Server 管理ガイド 第 1 巻』を参照してください。
  パブリケーションのサブスクリプションを作成する方法の詳細については、『Replication Server 管理ガイド 第 1 巻』の「サブスクリプションの管理」を参照してください。
- パブリケーションのサブスクリプションが作成またはリフレッシュされるときのみ、Replication Server は、新しいパブリケーションについての情報をレプリケートサイトに分配します。

create publication を使用するための条件

- **create publication** を実行する前に、次の条件を確認してください。
  - 入力するパブリケーション名が、プライマリデータサーバとプライマリデータベースでユニークである。
  - Replication Server から、プライマリテーブルまたはストアドプロシージャが格納されているデータベースへのコネクションが存在している。

サブスクリプションを作成するためのパブリケーションの準備

- パブリケーションを作成したら、**create article** を使用してアーティクルを作成し、パブリケーションに割り当てます。アーティクルは、テーブル複写定義またはファンクション複写定義を指定します。またアーティクルには、サブスクリプションを作成するレプリケートサイトの必要性に応じて、オプションの **where** 句を指定できます。詳細については、「create article」を参照してください。
- レプリケートテーブルでは、1 つのプライマリオブジェクトの複数の複写定義に対してサブスクリプションを作成することはできないため、同じプライマリテーブルとレプリケートテーブルの異なる複写定義の複数のアーティクルをパブリケーションに含めることはできません。
- すべてのアーティクルが割り当てられたら、レプリケートサイトでそのパブリケーションに対してサブスクリプションを作成できるように、**validate publication** を使用してパブリケーションを確定化してください。 パブリケーションを確定化することによって、パブリケーションに 1 つ以上のアーティクルが含まれていることが確認され、そのパブリケーションには、サブスクリプションの作成準備ができていることを示すマークが付けられます。 詳細については、「**validate publication**」を参照してください。
- パブリケーションのステータスを確認するには、**check publication** を使用します。 このコマンドは、パブリケーションに含まれるアーティクルの数を表示し、パブリケーションが有効 (VALID) かどうかを示します。 詳細については、「**check publication**」を参照してください。

パブリケーションのサブスクリプションの作成

- パブリケーションが有効であれば、レプリケートデータベースへの複写を開始するために、パブリケーションのサブスクリプションを作成できます。すべての形式のサブスクリプションマテリアライゼーションがサポートされています。詳細については、「create subscription」または「define subscription」を参照してください。
- パブリケーションサブスクリプションを作成すると、Replication Server は、パブリケーションに組み込まれたアーティクルごとに、別々の基本サブスクリプションを作成します。各アーティクルサブスクリプションは、親パブリケーションサブスクリプションの名前を使用します。
- パブリケーションのサブスクリプションには、**where** 句を指定することはできません。その代わりに、パブリケーションに組み込まれた各アーティクル内に 1 つまたは複数の **where** 句を指定することによって、レプリケートサイトへのレプリケーションをカスタマイズできます。

テーブル複写定義のアーティクル

- パブリケーションにテーブル複写定義のアーティクルだけが含まれている場合は、**create subscription** を使用して、アトミックマテリアライゼーションまたは

ノンアトミックマテリアライゼーションを使用するパブリケーションのサブスクリプションを作成できます。詳細については、「create subscription」を参照してください。

- パブリケーションサブスクリプションに対して、バルクマテリアライゼーションも使用できます。
  - レプリケートデータベースでデータがすでに存在している場合は、**without materialization** 句を指定して **create subscription** コマンドを使用します。
  - サブスクリプションデータを手動で転送する必要がある場合は、**define subscription** とその他のバルクマテリアライゼーションコマンドを使用します。詳細については、「define subscription」を参照してください。

ファンクション複写定義のアーティクル

- パブリケーションにファンクション複写定義のアーティクルだけが含まれている場合は、パブリケーションサブスクリプションに対してバルクマテリアライゼーションを使用してください。
  - レプリケートデータベースでデータがすでに存在している場合は、**without materialization** 句を指定して **create subscription** コマンドを使用します。詳細については、「create subscription」を参照してください。
  - サブスクリプションデータを手動で転送する必要がある場合は、**define subscription**、**activate subscription**、および **validate subscription** を使用し、バルクマテリアライゼーションを使用してパブリケーションのサブスクリプションを作成します。詳細については、「define subscription」を参照してください。

テーブル複写定義とファンクション複写定義の両方のアーティクル

- パブリケーションにテーブル複写定義とファンクション複写定義の両方のアーティクルが含まれている場合は、それぞれのタイプの複写定義に異なるマテリアライゼーションメソッドが必要な場合でも、同じサブスクリプションコマンドを使用できます。
  サブスクリプションを作成するには、まず、ファンクション複写定義用のコンポーネントサブスクリプションなど、バルクマテリアライゼーションを必要とするコンポーネントサブスクリプションのために、レプリケートデータベースにデータを転送してください。次に、**create subscription** を使用して、パブリケーションのサブスクリプションを作成してください。
  - テーブル複写定義のアーティクルのサブスクリプションは、**without materialization** 句を使用しないかぎり、アトミックマテリアライゼーションまたはノンアトミックマテリアライゼーションを使用してマテリアライズされます。
  - ファンクション複写定義のアーティクルのサブスクリプションは、マテリアライゼーションを使用しないでマテリアライズされます。

ファンクション複写定義のストアドプロシージャが、テーブル複写定義もあるテーブルに作用する場合、データを別々に転送する必要はありません。

パブリケーションサブスクリプションのリフレッシュ

- 既存のパブリケーションに新しいアーティクルを追加したり、パブリケーションからアーティクルを削除したりする場合、そのパブリケーションは不確定化されます。既存のアーティクルの複写は引き続き影響を受けませんが、新しいアーティクルの複写を開始したり新しいパブリケーションサブスクリプションを作成したりするには、次のようにしてください。
  - パブリケーションへの変更が終了したら、そのパブリケーションを確定化する。
  - 次に、パブリケーションのサブスクリプションをリフレッシュする。
- アトミックマテリアライゼーションまたはノンアトミックマテリアライゼーションを使用する場合に、パブリケーションサブスクリプションをリフレッシュするには、次のようにしてください。
  - **create subscription** を使用してサブスクリプションを再作成する。詳細については、「**create subscription**」を参照してください。
- バルクマテリアライゼーションを使用する場合に、パブリケーションサブスクリプションをリフレッシュするには、次のようにしてください。
  - レプリケートデータベースでデータがすでに存在している場合は、**without materialization** 句を指定して **create subscription** コマンドを使用します。
  - **define subscription**、**activate subscription**、および **validate subscription** を使用してサブスクリプションを再作成し、必要に応じてサブスクリプションデータを手動で転送する。詳細については、「**define subscription**」を参照してください。

サブスクリプション、アーティクル、パブリケーションの削除

- パブリケーションのサブスクリプションを削除できます。また、必要に応じて、テーブル複写定義のアーティクルのコンポーネントサブスクリプションについて、サブスクリプションデータをパージすることもできます。詳細については、「**drop subscription**」を参照してください。
- サブスクリプションがない場合は、パブリケーションに含まれるアーティクルを削除できます。また、対応する複写定義が他の場所で使用されていない場合は、必要に応じて複写定義を削除することもできます。アーティクルを削除すると、パブリケーションは無効になります。詳細については、「**drop article**」を参照してください。
- パブリケーションのサブスクリプションがない場合は、そのパブリケーションを削除できます。パブリケーションを削除すると、そのアーティクルも削除されます。パブリケーションのアーティクルの複写定義が他の場所で使用されていない場合は、必要に応じてそれらの複写定義をすべて削除することもできます。詳細については、「**drop publication**」を参照してください。

ウォームスタンバイアプリケーションのパブリケーション

- ウォームスタンバイアプリケーションでは、スタンバイデータベースへの複写に使用される複写定義が、パブリケーションに含まれるアーティクルによって指定されている場合もあります。

パーミッション

create publication には、“create object” パーミッションが必要です。

参照：
- check publication (231 ページ)
- create article (280 ページ)
- create applied function replication definition (274 ページ)
- create replication definition (346 ページ)
- create request function replication definition (362 ページ)
- create subscription (376 ページ)
- define subscription (395 ページ)
- drop article (403 ページ)
- drop publication (418 ページ)
- drop subscription (424 ページ)
- validate subscription (525 ページ)

create replication definition

複写するテーブルの複写定義を作成します。

構文

```
create replication definition replication_definition
with primary at data_server.database
[with all tables named [table_owner.] 'table_name' [quoted] |
[with primary table named [table_owner.]'table_name']
with replicate table named [table_owner.]'table_name'] [quoted]]
(
column_name [as replicate_column_name] [datatype [null | not null]
[datatype [null |not null ]
[map to published_datatype]] [quoted]
[, column_name [as replicate_column_name]
[map to published_datatype]] [quoted]...)
[references [table_owner.]table_name [(column_name)]]
)

primary key (column_name [, column_name]...)
[searchable columns (column_name [, column_name]...)]
[send standby [{all | replication definition} columns]]
[replicate {minimal | all} columns]
```

```
[replicate {SQLDML ['off'] | 'options'}]
[replicate_if_changed (column_name [, column_name]...)]
[always_replicate (column_name [, column_name]...)]
[with dynamic sql | without dynamic sql]
```

パラメータ

- **replication_definition –** 複写定義の名前です。名前は識別子の規則に従う必要があります。テーブル名を指定しない限り、複写定義名はプライマリテーブルとレプリケートテーブルの両方の名前と見なされます。
- **with primary at data_server.database –** プライマリデータのロケーションを指定します。プライマリデータベースがウォームスタンバイアプリケーションの一部である場合、*data_server.database* は論理データサーバと論理データベースの名前になります。
- **with all tables named –** プライマリデータベースとレプリケートデータベースの両方のテーブル名を指定します。*table_name* は、最大 200 文字の文字列です。オプションの *table_owner* は、テーブル所有者を表します。実際のテーブル所有者が、複写定義内に指定されたテーブル所有者と一致しない場合には、データサーバのオペレーションが失敗する可能性があります。
- **quoted – quoted** パラメータでは、作成されるテーブル名またはカラム名を引用符付き識別子として指定します。レプリケートに引用符を必要とする各オブジェクトで、引用符付き句を使用します。
- **with primary table named –** プライマリデータベースでのテーブル名を指定します。*table_name* は、最大 200 文字までの文字列です。*table_owner* はオプションで、テーブルの所有者を示します。実際のテーブル所有者が、複写定義内に指定されたテーブル所有者と一致しない場合には、データサーバのオペレーションが失敗する可能性があります。

  プライマリテーブル名を指定し、レプリケートテーブル名を指定しない場合、複写定義名がレプリケートテーブル名と見なされます。
- **with replicate table named –** レプリケートデータベースでのテーブルの名前を指定します。*table_name* は、最大 200 文字までの文字列です。*table_owner* はオプションで、テーブルの所有者を示します。実際のテーブル所有者が、複写定義内に指定されたテーブル所有者と一致しない場合には、データサーバのオペレーションが失敗する可能性があります。

  レプリケートテーブル名を指定してプライマリテーブル名を指定しない場合は、複写定義名がプライマリテーブル名とみなされます。
- **column_name –** プライマリテーブルからのカラム名です。1 つの句の中で、カラム名を複数回使用することはできません。

  カラムとデータ型はそれぞれ、カッコ **( )** で囲んでください。

- **as replicate_column_name –** プライマリカラムからのデータのコピー先となるレプリケートテーブル内のカラム名を指定します。この句は、送信元カラムと送信先カラムの名前が異なる場合に使用します。
- **datatype –** プライマリテーブルのカラムのデータ型です。データ型とその構文のリストについては、「データ型」を参照してください。

  カラムレベルのデータ型変換を指定する場合は、*declared_datatype* として使用してください。宣言したデータ型は、Replication Server のネイティブデータ型か、プライマリデータ型のデータ型定義でなければなりません。

  同じテーブルに対して複数の異なる複写定義を作成する場合、カラムのデータ型は同じである必要がありますが、パブリッシュデータ型は異なってもかまいません。詳細については、『Replication Server 管理ガイド 第 1 巻』を参照してください。

  同じテーブルに対して作成された複写定義にこのカラムがすでに含まれている場合、データ型の指定は任意です。
- **null または not null –** *text*、*unitext*、*image*、または *rawobject* カラムにのみ適用されます。レプリケートテーブルで null 値を許可するかどうかを指定します。デフォルトの **not null** は、レプリケートテーブルが null 値を受け入れないことを示します。

  text、*unitext*、*image*、*rawobject* の各カラムの null ステータスは、同じプライマリテーブルのすべての複写定義と一致するとともに、実際のテーブル内の設定とも一致する必要があります。同じプライマリテーブルの既存の複写定義に *text*、*unitext*、*image*、*rawobject* カラムが含まれている場合、null ステータスの指定は任意です。

  テーブルの複写定義にカラムを含めた後に、カラムのこの設定を変更することはできません。値を変更するには、そのカラムを含むすべての複写定義を削除して再作成する必要があります。
- **map to published_datatype –** カラムレベルのデータ型変換を行った後、クラスレベル変換とレプリケートデータベースへの提示を行う前のカラムのデータ型を指定します。
- **references table owner.table name column name –** プライマリデータベースで参照制約を持つテーブルの名前を指定します。*table_name* は、最大 200 文字の文字列です。*table_owner* は、オプションで、テーブルの所有者を表します。*column name* はオプションです。実際のテーブル所有者が、複写定義内に指定されたテーブル所有者と一致しない場合には、データサーバのオペレーションが失敗する可能性があります。
- **primary key column_name –** テーブルのプライマリキーを構成するカラムを指定します。1 つの句の中で、カラム名を複数回使用することはできません。

text、*unitext*、*image*、*rawobject*、*rawobject in row*、または *rs_address* カラムは、プライマリキーの一部として含めることはできません。

- **searchable columns column_name – create subscription**、**define subscription**、または **create article** の **where** 句で使用できるカラムを指定します。1 つの句の中で、カラム名を複数回使用することはできません。

  text、*unitext*、*image*、*rawobject*、*rawobject in row* カラム、または暗号化カラムをサーチャブルカラムとして指定することはできません。
- **send standby –** ウォームスタンバイアプリケーションで、スタンバイデータベースに複写するときの複写定義の使用方法を指定します。この句とそのオプションの使用方法の詳細については、「複写定義とウォームスタンバイアプリケーション」を参照してください。
- **replicate minimal columns または replicate all columns –** すべてのトランザクションのすべての複写定義カラムを送信するか、レプリケートデータベースで更新オペレーションまたは削除オペレーションを実行する必要があるカラムだけを送信します。デフォルトでは、**すべてのカラムを複写します。**

注意： 複写定義に `[replicate {minimal | all} columns]` 句が含まれる場合、`[replicate {minimal | all} columns]` 句が常に `[replicate {SQLDML ['off'] | 'options'}]` 句の前にある必要があります。

- **replicate SQLDML [‘off’] –** 指定された DML オペレーションの SQL の複写を有効または無効にします。
- **replicate 'options' –** 次の DML オペレーションの任意の組み合わせを複写します。
  - U - **update**
  - D - **delete**
  - I - **insert select**
- **replicate_if_changed –** *text*、*unitext*、*image*、または *rawobject* カラムのデータが変更された場合にのみ、これらのカラムを複写します。
- **always_replicate –** *text*、*unitext*、*image*、および *rawobject* カラムを常に複写します。
- **with dynamic sql –** コマンドが条件を満たしており、使用できるキャッシュ領域が十分にある場合に、DSI で動的 SQL をテーブルに適用することを指定します。これは、デフォルト値です。

  動的 SQL を使用するためにコマンドが満たす必要のある条件については、『Replication Server 管理ガイド 第 2 巻』を参照してください。
- **without dynamic sql –** DSI で動的 SQL コマンドを使用できないことを指定します。

例

- **例 1 –** *authors* テーブルの *authors_rep* という複写定義を作成します。 *authors* テーブルのプライマリコピーは、LDS データサーバの *pubs2* データベース内にあります。 テーブルのコピーも、すべて *authors* という名前になっています。 削除オペレーションと更新オペレーションの対象となる最小限のカラムだけが複写されます。

```
create replication definition authors_rep
  with primary at LDS.pubs2
  with all tables named 'authors'
   (au_id varchar(11), au_lname varchar(40),
    au_fname varchar(20), phone char(12),
    address varchar(12), city varchar(20),
    state char(2), country varchar(12), postalcode
    char(10))
  primary key (au_id)
  searchable columns (au_id, au_lname)
  replicate minimal columns
```

- **例 2 –** *blurbs_rep* (この *blurbs* は *pubs2* データベース内の “emily” が所有します) という複写定義を作成します。*text* データ型を使用し、null 値を受け入れる *copy* カラムのデータが変更されると、このカラムのデータが複写されます。

```
create replication definition blurbs_rep
  with primary at TOKYO_DS.pubs2
  with all tables named emily.'blurbs'
    (au_id char(12), copy text null)
  primary key (au_id)
  replicate_if_changed (copy)
```

- **例 3 –** *pubs2* データベース内のプライマリテーブル *publishers* に対して、1 つ以上の複写定義がすでに存在する場合、このコマンドは *pubs_copy_rep* という追加の複写定義を作成します。この複写定義では、“joe” が所有者である *pubs_copy* という名前のレプリケートテーブルのサブスクリプションが作成されます。レプリケートテーブルの名前が *pubs_copy* であっても、所有者が “joe” でない場合、サブスクリプションの作成は失敗します。

```
create replication definition pubs_copy_rep
  with primary at TOKYO_DS.pubs2
  with primary table named 'publishers'
  with replicate table named joe.'pubs_copy'
    (pub_id, pub_name as pub_name_set)
  primary key (pub_id)
```

  プライマリテーブルの *pub_name* カラムのデータは、レプリケートテーブルの *pub_name_set* カラムに複写されますが、同じデータ型を共有する必要があります。既存の複写定義内にあるカラムに対して、データ型を指定する必要はありません。この例では、プライマリテーブルの *city* カラムと *state* カラムはレプリ

ケートテーブル *pubs_copy* には必要ないため、この複写定義から除外されています。

- **例 4 –** *authors* テーブルの変更されたすべてのカラムをスタンバイデータベースに複写する複写定義を作成します。この定義では MSA にも複写しますが、*au_id* カラムと *au_lname* カラムの変更された値だけが複写されます。*au_id* は、*authors* テーブルの更新と削除に使用されるキーです。

```
create replication definition authors_rep
  with primary at LDS.pubs2
  with all tables named 'authors'
    (au_id varchar(11), au_lname varchar(40))
  primary key (au_id)
  send standby
  replicate minimal columns
```

- **例 5 –** テーブル *foo* を作成します。ここで、*foo_col1* は引用符付き識別子です。

```
create replication definition repdef
   with primary at primaryDS.primaryDB
   with all tables named "foo"
   ("foo_col1" int quoted, "foo_col2" int)
   primary key ("foo_col1")
```

- **例 6 – update** 文と **delete** 文を複写するテーブル複写定義を作成します。

```
create replication definition repdef1
   with primary at ds3.pdb1
   with all tables named 'tb1'
   (id_col int, str_col char(40))
   primary key (id_col)
   replicate all columns
   replicate 'UD'
go
```

- **例 7 –** 参照関係を含むテーブル複写定義を作成します。

```
create replication definition doctors_rep
with primary at MED_DS.pubs2
with all tables named doctors
  (t1id     int,
   logid    int
   references doctors_main (logid),
   t1c1     VARCHAR(255),
   t1c2     VARCHAR(15))
 primary key (t1id)
 replicate minimal columns
```

使用法

- このコマンドは、テーブルのプライマリバージョンが格納されているデータベースを管理する Replication Server で実行してください。

- **rs_helprep** は、Replication Server バージョン 12.0 以前で使用できる複写定義を調べるために使用します。詳細については、「**rs_helprep**」を参照してください。
- 複写テーブルの定義と管理の概要と、複写定義、アーティクル、パブリケーションの処理の詳細については、『Replication Server 管理ガイド 第 1 巻』を参照してください。
- **create replication definition** コマンドを実行する前に、次のことを確認してください。
  - 入力する複写定義名が、複写システム内のすべての複写定義 (テーブルまたはファンクション) 間でユニークである。**create replication definition** を入力するときに、Replication Server がこの条件を常に要求するわけではありません。
  - Replication Server からプライマリテーブルが格納されているデータベースへのコネクションが存在する。詳細については、「**create connection**」を参照してください。**rs_init** を使用しても、データベースコネクションを作成できます。詳細については、使用しているプラットフォーム用の『Replication Server インストールガイド』と『Replication Server 設定ガイド』を参照してください。
  - 複数のバージョンの Replication Server (たとえば、バージョン 12.0 とバージョン 11.0.x) を使用し、同じプライマリテーブルに対して複数の複写定義を作成した場合は、複写システムの混合バージョンによる問題を確認する (たとえば、両方のバージョンで同じテーブルに対するカラム名が異なる場合)。詳細については、「複数の複写定義の作成」を参照してください。
- Replication Server は、新しい複写定義を、条件を満たしているサイトへ複写システムを介して分配します。複写システムで通常の遅延時間が発生するため、変更はすぐにはレプリケートサイトに反映されません。
- プライマリデータベースについて作成された複写定義は、複写定義を管理する Replication Server とプライマリデータベース間のすべてのプライマリコネクション、デフォルトコネクション、および代替コネクションに適用されます。したがって、プライマリデータベースへの最後のプライマリコネクションを削除する前に、プライマリデータベースのすべての複写定義を削除する必要があります。

  システムバージョン 1570 では、データベースに対してのみ複写定義とパブリケーションを作成できます。**with primary at** 句 (**create replication definition** コマンド) について指定する名前は、プライマリデータベースの名前である必要があります。
- LOB 圧縮データのサブスクリプションマテリアライゼーションのサポートは、複写定義でのカラムのデータ型の指定方法、また、Replication Server のバージョンによって異なります。『Replication Server 管理ガイド 第 1 巻』の「LOB 圧縮データのサブスクリプションマテリアライゼーション」を参照してください。

複写ステータス

- *text*、*unitext*、*image*、および *rawobject* カラムの複写ステータスは、Adaptive Server データベース内と複写定義内で同じである必要があります。
- 複写ステータスを変更するには、**alter replication definition** を使用します。
- 複写ステータスは、同じプライマリテーブルに対して作成したすべての複写定義において、一貫していなければなりません。
  - **alter replication definition** を使用して複写ステータスを変更すると、同じプライマリテーブルに対する他の複写定義の複写ステータスも変更されます。
  - 同じプライマリテーブルに対する別の複写定義内に、カラムがすでにリストされている場合、複写ステータスを指定する必要はありません。

複数の複写定義の作成

- 同じプライマリテーブルに対して複数の複写定義を作成し、それぞれの複写定義をカスタマイズできます。これによって、プライマリテーブルや他のレプリケートテーブルと異なる特性を持つレプリケートテーブルとなるようなサブスクリプションを作成できます。
  また、プライマリテーブルの記述に加えて、各複写定義では、カラム数を制限したり、レプリケートテーブルに対して異なるカラム名、または異なるテーブル名を指定したりできます。指定された特性に一致するレプリケートテーブルは、複写定義にサブスクリプションを作成することができます。また、レプリケートテーブルとプライマリテーブルが一致する場合でも、複数の複写定義を使用できます。
  この機能を使用すると、データベースの要件が異なる場合に、通常の複写用の複写定義と、スタンバイ用の別の複写定義を作成することもできます。詳細については、『Replication Server 管理ガイド 第 1 巻』を参照してください。
- レプリケートテーブルでは、プライマリテーブルごとに 1 つの複写定義のサブスクリプションしか作成できませんが、同じ複写定義のサブスクリプションは何回でも作成できます。
- 同じプライマリテーブルに対して複数の異なる複写定義を作成する場合は、*text*、*unitext*、*image* カラムに同じカラムデータ型と同じ null ステータスを使用する必要があります。
- 内部複写定義を使用して、スタンバイコネクションまたは MSA コネクションにテーブルを複写し、そのコネクションの動的 SQL が有効になっている場合は、テーブルの新しい複写定義で、プライマリデータベースのカラム順序と一致するカラム順序を定義します。このように定義しないと、既存の準備文が無効になることがあり、スタンバイコネクションまたは MSA コネクションの再起動が必要になる場合があります。

ファンクションとファンクション文字列

- Replication Server は、複写定義に対して **rs_insert**、**rs_delete**、**rs_update**、**rs_truncate**、**rs_select**、および **rs_select_with_lock** の各ファンクションを作成します。複写定義に *text*、*unitext*、*image*、または *rawobject* データが含まれている

場合は、**rs_datarow_for_writetext**、**rs_get_textptr**、**rs_textptr_init**、および **rs_writetext** の各ファンクションも作成します。

- システム提供ファンクション文字列クラスと、これらのクラスから継承した派生クラスでは、Replication Server がこれらのファンクションのデフォルトのファンクション文字列を生成します。一部のファンクション文字列は動的に生成されるため、RSSD 内には存在しません。その他のファンクション文字列クラスでは、すべてのファンクション文字列を作成してください。
- 各ファンクション文字列クラスでは、同じテーブルの複写定義ごとに、システムファンクションの独自のファンクション文字列セットがあります。
- ユーザ定義ファンクションの作成、削除、または変更を行うと、同じプライマリテーブルのすべての複写定義に対して、ユーザ定義ファンクションの作成、削除、または変更が行われます。
- 同じプライマリテーブルに対する異なる複写定義は、同じユーザ定義ファンクションを共有しますが、各ユーザ定義ファンクションには独自のファンクション文字列があります。テーブル複写定義に対応するメソッドを使用してストアドプロシージャを複写する場合は、**create function** を使用してユーザ定義ファンクションを作成します。

カラムとデータ型の指定

- 複写するカラムとデータ型を指定するときは、次のガイドラインに従ってください。
  - ユーザ定義データ型を持つカラムは、複写定義内で基本となるデータ型を使用して定義する必要があります。
  - *text*、*unitext*、*image*、または *rawobject* カラムの複写ステータス (**replicate_if_changed**、**always_replicate**) は、プライマリテーブルのすべての複写定義で同じであることが必要です。*text*、*unitext*、*image*、または *rawobject* カラムの複写ステータスを、**alter replication definition** を使用して変更すると、同じプライマリテーブルの他の複写定義でも該当のカラムの複写ステータスが変更されます。
    同じテーブルの複写定義に含まれる *text*、*unitext*、*image*、または *rawobject* カラムの複写ステータスを指定する必要はありません。
  - *numeric* データ型の宣言からは、長さと精度を省きます。Replication Server では、*numeric* データ型の値は、精度の影響を受けずに処理されます。

注意： **map to** オプションを使用して、大きな **varchar** をカラムあたりの文字数が少ない **varchar** に変換する場合、複写するデータが複写先カラムの文字長を超えないようにしてください。

たとえば、複写する項目が **varchar(25)** の制限を超えていなければ、**varchar(100)** を **varchar(25)** カラムにマップできます。制限を超えた場合は、エラーメッセージが表示されます。

- 複写定義に追加するカラムに `identity` カラムが含まれている場合、Transact-SQL の **identity_insert** オプションを使用して、テーブルに対するオペレーションを行うには、レプリケートデータベースでメンテナンスユーザがテーブルの所有者であるか、“dbo” または “dbo” のエイリアスであるか、sa_role のパーミッションを持っている必要があります。適切なパーミッションがない場合、`identity` カラムの複写を進めることができないときに Replication Server ログにエラーメッセージが表示されます。
  (1 つまたは複数の複写定義を持つ) プライマリテーブルに含めることができる `identity` カラムは 1 つだけです。 ただし、**map to** オプションを使用すると、1 つまたは複数の複写定義で複数のカラムを `identity` データ型としてパブリッシュできます。
- 複写定義のカラムリストに `timestamp` カラムが含まれており、レプリケートテーブルが Adaptive Server にある場合、メンテナンスユーザはレプリケートデータベースのテーブルの所有者 (または "dbo" か "dbo" のエイリアス) であることが必要です。
  1 つまたは複数の複写定義を持つプライマリテーブルに含めることができる `timestamp` カラムは 1 つだけです。 ただし、**map to** オプションを使用すると、1 つまたは複数の複写定義で複数のカラムを `timestamp` データ型としてパブリッシュできます。
- *rs_address* データ型では、独自のサブスクリプション解析手法を使用できます。 *rs_address* データ型のビットマップ (基本となる *int* データ型に基づく) を、サブスクリプションの **where** 句のビットマスクと比較して、ローをレプリケートすべきかどうかを判断できます。 このサブスクリプション解析メソッドを使用するには、最初に、*int* データ型のカラムを使用するテーブルを作成してください。 複写定義を作成するときに、これらのカラムをカラムリストに含めますが、データ型は *int* ではなく *rs_address* として宣言します。
  詳細については、「**create subscription**」を参照してください。また、『Replication Server 管理ガイド 第 1 巻』に記載してある *rs_address* データ型の使用方法の詳細も参照してください。

カラムレベル変換に使用するカラムとデータ型の指定

- *text*、*unitext*、*image*、または *rawobject* データ型は、基本データ型またはデータ型定義として使用することはできません。また、カラムレベル変換やクラスレベル変換の変換元または変換先として使用することもできません。
- *declared_datatype* は、Replication Server に配信される値のデータ型によって、次のように異なります。
  - Replication Agent が Replication Server のネイティブデータ型を配信する場合、*declared_datatype* はそのネイティブデータ型になる。

    - Replication Agent がその他のデータ型を配信する場合、*declared_datatype* はプライマリデータベースの元のデータ型のデータ型定義である必要がある。
- *published_datatype* は、カラムレベル変換を行った後、クラスレベル変換を行う前のデータ型です。*published_datatype* は、Replication Server のネイティブデータ型、またはターゲットデータベースのデータ型のデータ型定義である必要があります。
- 複数の複写定義で宣言されたカラムでは、各複写定義内で同じ *declared_datatype* を使用する必要があります。*published_datatype* は異なってもかまいません。

replicate minimal columns オプションの使用

- **replicate minimal columns** オプションを使用すると、DSI パフォーマンスの向上、メッセージオーバーヘッドの削減、キューサイズの削減が可能になります。また、このオプションは、実際には変更されていないカラムに設定されたトリガが原因で発生するアプリケーションの問題を防ぐうえでも役立ちます。

  注意： 複写定義に **replicate all columns** 句が含まれ、かつ **replicate minimal columns** コネクションプロパティが 'on' に設定されている場合、そのコネクションは最少数のカラムをレプリケートします。ターゲットデータベースにカラムをすべてレプリケートするには、DSI コネクションの **replicate minimal columns** 値を "off" に設定します。

  このオプションの機能の詳細については、『Replication Server 管理ガイド 第 2 巻』を参照してください。
- 最少数カラムのレプリケーションには、次の要件が適用されます。
    - 通常、**replicate minimal columns** は、**rs_update** および **rs_delete** ファンクションのデフォルトのファンクション文字列を使用する複写定義でのみ使用できる。**replicate minimal columns** を指定して、ファンクション文字列内で *rs_default_fs* システム変数を使用すると、デフォルトではない **rs_update** および **rs_delete** ファンクション文字列を、複写定義に対して作成できます。詳細については、「**create function string**」を参照してください。
    - オートコレクションは、**replicate minimal columns** オプションとともには使用できない。**replicate minimal columns** を設定する前に **set autocorrection on** を指定すると、削除または更新の各オペレーションについて情報メッセージがログに取られます。**replicate minimal columns** を先に指定した場合、複写定義に **set autocorrection on** を指定することはできません。
    - 複写定義に **replicate minimal columns** を指定している場合、ノンアトミックマテリアライゼーション (**create subscription** コマンド、**without holdlock** オプション) を使用して複写定義のサブスクリプションを作成することはできない。この場合、ノンアトミックマテリアライゼーションをシミュレートす

るバルクマテリアライゼーションオプションを使用します。詳細については、『Replication Server 管理ガイド 第 2 巻』を参照してください。

`text`、`unitext`、`image`、または `rawobject` データ型のレプリケーション

- 複写定義のプライマリキーには、テーブル内の 1 つのローをユニークに識別する 1 つまたは複数のカラムを含めます。
- **always_replicate** 句と **replicate_if_changed** 句を使用すると、*text*、*unitext*、*image*、*rawobject* の各カラムの複写ステータスを指定できます。このステータスは、Adaptive Server のシステムプロシージャ **sp_setreptable** と **sp_setrepcol** (またはいずれか)、または **sp_reptostandby** でも設定できます。複写ステータスは、Adaptive Server システムプロシージャ内とプライマリテーブルの複写定義内で同じである必要があります。複写ステータスに一貫性がないと、RepAgent が停止する場合があります。ステータスの設定方法と矛盾が発生した場合の解決方法については、『Replication Server 管理ガイド 第 1 巻』を参照してください。*text*、*unitext*、*image*、*rawobject* データをウォームスタンバイアプリケーションにレプリケートする方法の詳細については、複写定義とウォームスタンバイアプリケーションを参照してください。
- 複写定義の複写ステータスを **always_replicate** に指定する必要があるのは、**sp_setreptable** だけを使用してテーブルをマーク付けするときです。これは、**sp_setreptable** のデフォルトの複写ステータスが **always_replicate** だからです。テーブルの複写ステータスを **replicate_if_changed** に変更できます。そのためには、テーブルの複写定義の複写ステータスを **replicate_if_changed** に変更し、**sp_setrepcol** 複写ステータスを **replicate_if_changed** に設定した状態でテーブル内のすべてのカラムをマーク付けします。
- *text*、*unitext*、*image*、または *rawobject* データ型のレプリケーションには、次の要件が適用されます。
  - *text*、*unitext*、*image*、または *rawobject* カラムが **replicate_if_changed** カラムリストに含まれている場合、複写定義のオートコレクションを有効にしようとするとエラーが発生する。オートコレクションを有効にするには、複写定義の **always_replicate** リストに *text*、*unitext*、*image*、*rawobject* のすべてのカラムが含まれている必要がある。
  - *text*、*unitext*、*image*、または *rawobject* カラムの **replicate_if_changed** ステータスがプライマリテーブルでの更新オペレーションで変更されず、この更新によってローがサブスクリプションにマイグレートすると、レプリケートテーブルに挿入されたローで *text*、*unitext*、*image*、または *rawobject* データが消失することになる。ローがサブスクリプションにマイグレートし、*text*、*unitext*、*image*、または *rawobject* データが消失していると、Replication Server はエラーログに警告メッセージを表示する。この場合、**rs_subcmp** を実行して、レプリケートテーブルとプライマリテーブルのデータを調整する。

複写定義とウォームスタンバイアプリケーション

- Replication Server では、ウォームスタンバイアプリケーション内のスタンバイデータベースを保持するために、複写定義は必要ありません。複写定義を使用すると、スタンバイデータベースへの複写のパフォーマンスが向上する場合があります。この目的のためだけに、論理データベース内の各テーブルに複写定義を作成できます。
- 複写定義を使用してテーブルのトランザクションをスタンバイデータベースに複写する任意のオプションを指定して、**send standby** を使用します。複写定義のプライマリキーカラムと **replicate minimal columns** の設定は、スタンバイデータベースへの複写に使用されます。このメソッドのオプションには、次のものがあります。
  - **send standby** または **send standby all columns** を使用すると、テーブル内のすべてのカラムをスタンバイデータベースに複写できます。
  - 複写定義のカラムだけをスタンバイデータベースにレプリケートするには、**send standby replication definition columns** を使用します。
- スタンバイデータベースにレプリケートするときに、このテーブルの複写定義を一切使用しないことを示すには、**alter replication definition** 内で **send standby off** を使用します。

  プライマリテーブルに、スタンバイで使用するようにマーク付けされている複写定義がない場合は、すべてのカラムがスタンバイデータベースにレプリケートされます。また、そのテーブルのすべての複写定義のプライマリキーをすべて結合したものがプライマリキーに使用され、最少数のカラムがレプリケートされます。論理コネクションの **replicate_minimal_columns** 設定によって、更新と削除の対象として最少カラムを送信するか、すべてのカラムを送信するかが決まります。詳細は、「alter logical connection」と「alter replication definition」を参照してください。
- スタンバイデータベースへの複写に複写定義を使用することによって実現されるパフォーマンスの最適化の詳細については、『Replication Server 管理ガイド 第 2 巻』を参照してください。
- 複数の複写定義を持つプライマリテーブルに、スタンバイで使用するようにすでにマーク付けされている複写定義がある場合、**send standby** を指定して別の複写定義を作成または変更すると、最初の複写定義のマークが解除されます。
- 複写定義の複写ステータスを **replicate_if_changed** に指定する必要があるのは、**sp_reptostandby** だけを使用してデータベースにマーク付けしたときです。これは、**sp_reptostandby** のデフォルトの複写ステータスが **replicate_if_changed** だからです。**sp_reptostandby** だけを使用してデータベースにマーク付けした場合、*text*、*unitext*、*image*、*rawobject* カラムの複写ステータスを変更することはできません。
- **sp_reptostandby** を使用してデータベースにマーク付けし、**sp_setreptable** を使用してそのデータベース内のテーブルにマーク付けした場合は、複写定義の複写ステータスを **always_replicate** に指定してください。これは、デフォルトの複写ステータスが **always_replicate** だからです。テーブルの複写ステータスを

replicate_if_changed に変更できます。そのためには、テーブルの複写定義の複写ステータスを **replicate_if_changed** に変更し、**sp_setrepcol** 複写ステータスを **replicate_if_changed** に設定した状態でテーブル内のすべてのカラムをマーク付けします。

複写定義の変更

- **alter replication definition** は、既存の複写定義にカラムまたはサーチャブルカラムを追加したり、その複写定義の設定に別の変更を加えたりする場合に使用します。詳細は、「alter replication definition」を参照してください。
- 既存の複写定義内のプライマリカラムを削除したり名前を変更したりする必要がある場合は、複写定義に対するすべてのサブスクリプションを削除し、複写定義を削除してから再作成し、次にサブスクリプションを再作成します。

ストアドプロシージャの複写

- ストアドプロシージャの複写を有効にするには、**create applied function replication definition** または **create request function replication definition** を使用します。ストアドプロシージャの複写の概要については、『Replication Server 管理ガイド 第 1 巻』を参照してください。

計算カラムの複写

- **create replication definition** は、マテリアライズされた計算カラムのレプリケーションをサポートしています。マテリアライズされた計算カラムは、複写定義で基本データ型を使用して定義する必要があります。
- マテリアライズされた計算カラムは、通常カラムと同様にテーブルのページに値が格納された計算カラムです。マテリアライズされた計算カラムは、ベースカラムで挿入または更新が発生するたびに再評価されます。クエリでは再評価されません。
- 計算カラムには、仮想計算カラムまたは非マテリアライズ計算カラムと呼ばれる別のタイプがあります。この計算カラムの値は、テーブルまたはインデックスには格納されません。仮想計算カラムはクエリで参照された場合にのみ評価され、挿入オペレーションまたは更新オペレーションでは何も実行されません。
  仮想計算カラムのレプリケーションはサポートされていないため、複写定義に含めないようにしてください。

計算カラムの複写の詳細については、『Replication Server 管理ガイド 第 1 巻』を参照してください。

引用符付き識別子の使用

- レプリケートに引用符を必要とする各オブジェクトで、引用符付き句を使用します。**quoted** パラメータを使用して識別子をマーク付けし、複写定義にサブスクリプションを作成するレプリケートサーバの **dsi_quoted_identifiers** が on に設定されている場合、そのレプリケートサーバは引用符付き識別子としてマーク

付けされた識別子を受け取ります。**dsi_quoted_identifier** が off の場合、マーク付けは無視され、レプリケートサーバは引用符付き識別子を受け取りません。

- ウォームスタンバイデータベースおよび複写定義のサブスクライバへの複写時に、プライマリテーブル名は引用符付きとしてマーク付けされているが、レプリケートテーブル名はマーク付けされていない場合 (またはその逆)、Replication Server は、プライマリテーブル名とレプリケートテーブル名の両方を引用符付きとして送信します。
- 識別子での埋め込み二重引用符はサポートされません。
- Adaptive Server、SQL Anywhere、Microsoft SQL Server、Universal Database (UDB)、Oracle などのデータサーバでは、サポートされる長さ、特殊文字、および予約語に関して、引用符付き識別子は異なる方法で処理されます。異機種環境では、複写されている引用符付き識別子がプライマリデータサーバとレプリケートデータサーバの両方で有効であることを確認してください。
- 引用符付き識別子のレプリケーションを成功させるには、プライマリ Replication Server とレプリケートデータサーバに接続する Replication Server のバージョンを 15.2 以降にします。ただし、ルート上の中間 Replication Server は、古いバージョンでもかまいません。

SQL 文の複写

- **send standby** 句を含むテーブル複写定義を使用して、**replicate 'I'** 文を指定できます。**insert select** 文を SQL 文の複写として複写できるのは、ウォームスタンバイまたは MSA 環境のみです。**send standby** 句が含まれないテーブル複写定義では、**insert select** 文を複写できません。
- デフォルトでは、ウォームスタンバイアプリケーションは、DML コマンド (SQL 文のレプリケートをサポート) をレプリケートしません。SQL レプリケーションを使用するには、以下を行います。
  - **replicate SQLDML** 句と **send standby** 句を使用して、テーブル複写定義を作成する。
  - **WS_SQLDML_REPLICATION** パラメータを on に設定する。デフォルト値は **UDIS** です。ただし、**WS_SQLDML_REPLICATION** の優先度は SQL の複写のテーブル複写定義よりも低い。テーブル複写定義にテーブルの **send standby** 句が含まれている場合、その句によって、DML 文をレプリケートするかどうかが決定される。**WS_SQLDML_REPLICATION** パラメータの設定とは無関係。
- SQL 文の複写ではオートコレクションを実行できません。データサーバインタフェース (DSI) で、SQL 文の複写対象の DML コマンドが検出され、オートコレクションが on になっている場合、デフォルトで DSI がサスペンドされ、複写が停止されます。Replication Server でこのエラーを処理する方法を指定するには、エラー番号 5193 を使用して、**assign action** コマンドを使用します。

テーブルレベルのサブスクリプションが確定化されるまで、Replication Server は SQLDML を複写しません。

- 次の場合、SQL 文の複写はサポートされません。
  - レプリケートデータベースに、プライマリデータベースとは異なるテーブルスキーマがある。
  - Replication Server が、データまたはスキーマの変換を実行する必要がある。
  - サブスクリプションに **where** 句が含まれている。
  - update に 1 つ以上の *text* または *image* カラムが含まれている。

参照制約のあるテーブルの扱い

alter replication definition と **create replication definition** (**reference** 句あり) の両方で、Replication Server は以下のように動作します。

- **reference** 句をカラムプロパティとして扱う。各カラムはテーブルを 1 つだけ参照できる。
- **reference** 句内の *column_name* パラメータに指定したカラム名を処理しない。
- 循環参照になる参照制約を許可しない。たとえば、元の参照先テーブルは元の参照元テーブルへの参照制約を持つことはできない。

複写プロセスでは、RTL は次のようにロードします。

- 参照先テーブルへの挿入の後で複写定義で指定した参照元テーブルに挿入する。
- 複写定義で指定したテーブルでの削除の後で参照先テーブルを削除する。

場合によっては、両方のテーブルでの更新が競合によって失敗することがあります。RTL が複写処理のリトライをしないようにして、パフォーマンスの低下を防ぐには、以下を行います。

- 更新を削除と挿入に変換するように、**dsi_command_convert** を "u2di" に設定してレプリケーションの更新を停止する。
- **dsi_compile_enable** を off にして、影響を受けたテーブルがコンパイルされるのを避ける。

RTL は、カスタムファンクション文字列を持つテーブルと、コンパイルできない既存テーブルへの参照制約を持つテーブルをコンパイルできないため、それらのテーブルをマークアウトします。これらのテーブルにマークを付けることによって、RTL は参照制約エラーによって発生するトランザクションのリトライを避け、複写処理を最適化できます。

パーミッション

create replication definition には、"create object" パーミッションが必要です。

参照：

- alter function string (188 ページ)
- alter replication definition (199 ページ)
- configure logical connection (238 ページ)
- create connection (287 ページ)
- create applied function replication definition (274 ページ)
- create request function replication definition (362 ページ)
- create function string (318 ページ)
- create subscription (376 ページ)
- drop replication definition (419 ページ)
- rs_set_quoted_identifier (578 ページ)
- set (444 ページ)
- sp_setrepcol (664 ページ)
- sp_setreptable (675 ページ)
- rs_send_repserver_cmd (735 ページ)

create request function replication definition

複写するストアドプロシージャの要求ファンクション複写定義とユーザ定義ファンクションを作成します。要求ファンクションは、プライマリデータベースでストアドプロシージャを実行するユーザと同じユーザによってレプリケートデータベースで適用されます。

構文

```
create request function replication definition repdef_name
with primary at dataserver.database
with primary function named 'func_name'
with replicate function named 'func_name'
([@param_name datatype [, @param_name datatype]…])
[searchable parameters (@param_name [, @param_name]…)]
[send standby {all | replication definition} parameters]
```

パラメータ

- **repdef_name –** ファンクション複写定義の名前です。名前は識別子の規則に従う必要があります。
- **with primary at –** プライマリデータを格納するデータサーバとデータベースを指定します。

- **dataserver –** プライマリデータを格納するデータサーバの名前です。プライマリデータベースがウォームスタンバイアプリケーションの一部である場合、*dataserver* は論理データサーバ名になります。
- **database –** プライマリデータを格納するデータベースの名前です。プライマリデータベースがウォームスタンバイアプリケーションの一部である場合、*database* は論理データベース名になります。
- **with primary function named –** プライマリデータベースのストアドプロシージャ名を指定します。プライマリファンクション名を指定しない場合、Replication Server はプライマリファンクションの名前として複写定義名を使用します。プライマリファンクション名は、**with replicate function named** 句で指定したレプリケートファンクション名とは異なる必要があります。
- **'func_name' –** ファンクションの名前です。最大長は 255 文字です。
- **with replicate function named –** レプリケートデータベースで実行するストアドプロシージャの名前を指定します。レプリケートファンクション名を指定しない場合、Replication Server はレプリケートファンクションの名前として複写定義名を使用します。レプリケートファンクション名は、**with primary function named** 句で指定したプライマリファンクション名とは異なる必要があります。

  注意： プライマリストアドプロシージャはクライアントによって呼び出されるストアドプロシージャを指し、レプリケートストアドプロシージャはプライマリデータベースから複写され、レプリケート Replication Server によって呼び出されるストアドプロシージャを指します。

  要求ファンクションのこの動作は、Replication Server 15.0.1 以前の要求ファンクションの動作とは異なります。Replication Server 15.0.1 以前のバージョンにおける要求ファンクションの動作の詳細については、『Replication Server 管理ガイド 第 2 巻』を参照してください。

- **@param_name –** ファンクションからのパラメータ名です。1 つの句の中で同じパラメータ名を 2 回以上指定することはできません。パラメータとそのデータ型の指定は必須ではありませんが、パラメータを指定するかどうかに関係なく、この句はカッコで囲んでください。
- **datatype –** ファンクションのパラメータのデータ型です。データ型とその構文のリストについては、「データ型」を参照してください。Adaptive Server のストアドプロシージャとファンクション複写定義には、*text*、*unitext*、*rawobject*、*image* の各データ型のパラメータを含めることはできません。
- **searchable parameters – where** 句 (**define subscription**、**create subscription**、または **create article**) で使用できるパラメータのリストを指定します。この句を含める場合は、パラメータ名をカッコ **()** で囲んでください。
- **send standby –** ウォームスタンバイアプリケーションで、スタンバイデータベースにファンクションのすべてのパラメータを送信するか (**send standby all parameters**)、複写定義で指定されたパラメータだけを送信するか (**send standby**

replication definition parameters) を指定します。デフォルトは、**send standby all parameters** です。

例

- **例 1 – titles_frep** という要求ファンクション複写定義 (**upd_titles_prim** というファンクションの) を作成します。送信先データベースで呼び出されるストアドプロシージャは、**upd_titles** です。

```
create request function replication definition titles_frep
with primary at LDS.pubs2
with primary function named 'upd_titles_prim'
with replicate function named 'upd_titles'
(@title_id varchar(6), @title varchar(80), @type char(12), @pub_id
char(4),
@price money, @advance money, @total_sales int)
searchable parameters (@title_id, @title)
```

使用法

- **create request function replication definition** は、複写するストアドプロシージャを記述するときに使用します。適用ファンクション複写定義と要求ファンクション複写定義の違いは、適用ファンクション複写定義を使用して複写されたファンクションは、レプリケートサイトでメンテナンスユーザが実行するのに対し、要求ファンクション複写定義を使用して複写されたファンクションは、プライマリサイトでプライマリファンクションを実行するユーザと同じユーザがレプリケートサイトで実行する点です。複写ストアドプロシージャの概要については、『Replication Server 管理ガイド 第 1 巻』を参照してください。
- プライマリファンクションの要求ファンクション複写定義を作成する場合は、そのファンクションに次の 2 つの条件を満たす既存のファンクション複写定義がまだないことを確認してください。
  - **create function replication definition** コマンドを使用して作成されている。
  - そのファンクション複写定義が、Replication Server 15.0.1 以前のバージョンでサブスクリプションのない要求ファンクション複写に使用されている。

  上記の両方の条件に該当する場合、既存の要求ファンクション複写定義は無効になります。Replication Server 15.0.1 以前の要求ファンクション複写定義の詳細については、『Replication Server 管理ガイド 第 2 巻』を参照してください。
- **create request function replication definition** コマンドは、プライマリストアドプロシージャが格納されているデータベースを管理する Replication Server で実行します。
- **create request function replication definition** を実行する前に、次のことを確認してください。

    - ファンクション複写定義の名前が、複写システム内でユニークであること。Replication Server は、**create request function replication definition** の使用時に、この要件を常に適用できるわけではありません。
    - Replication Server からプライマリデータが格納されているデータベースへのコネクションが存在していること。**create connection** を参照してください。コネクションは、**rs_init** を使用して作成することもできます。使用しているプラットフォームの『Replication Server インストールガイド』と『Replication Server 設定ガイド』を参照してください。
    - ファンクション複写定義に指定した名前、パラメータ、データ型が、関連するストアドプロシージャの名前、パラメータ、データ型と一致していること。ファンクション複写定義で指定したパラメータだけが複写されます。
- Replication Server は、新しいファンクション複写定義を、条件を満たしているサイトへ複写システムを介して分配します。複写システムで通常の遅延時間が発生するため、変更内容がレプリケートサイトにすぐに反映されるわけではありません。

ユーザ定義ファンクションとファンクション文字列

- 要求ファンクション複写定義を作成すると、Replication Server は、対応するユーザ定義ファンクションを自動的に作成します。同様に、**rs_sqlserver_function_class** では、Replication Server はユーザ定義ファンクションのデフォルトのファンクション文字列を自動的に作成します。
- **rs_sqlserver_function_class** とユーザ定義ファンクション文字列クラスのファンクション文字列は、**create function string** を使用してカスタマイズできます。
- ユーザ定義ファンクションを使用する、ユーザが作成した各基本ファンクション文字列クラスと、これらのクラスから継承した各派生クラスでは、**create function string** を使用してファンクション文字列を作成します。ファンクション文字列では、レプリケートデータサーバに適した言語を使用して、ストアドプロシージャまたは RPC を呼び出す必要があります。
- ファンクション文字列クラス、ファンクション文字列、ファンクションの概要については、『Replication Server 管理ガイド 第 2 巻』を参照してください。

with primary at 句

with primary at 句は、プライマリデータサーバとプライマリデータベースを指定するときに使用します。プライマリデータベースは、呼び出されるプライマリストアドプロシージャを格納するデータベースです。

with replicate function named 句

with replicate function named 句は、複写ファンクションを配信する送信先データベースで実行するストアドプロシージャの名前を指定するときに使用します。ファンクション複写定義を作成または変更するときに **with replicate function named** を使用しない場合、ファンクションはファンクション複写定義と同じ名前のストアドプロシージャとして配信されます。ウォームスタンバイデータベースのスト

アドプロシージャは、アクティブデータベースのストアドプロシージャと同じ名前であるため、**with replicate function named** は無視されます。

往復複写では、データベースは別のデータベースにデータ変更要求を送信し、そのデータ変更を要求側のデータベースに複写できます。適用ファンクション複写定義と要求ファンクション複写定義の両方を使用して、往復複写を設定する方法の詳細については、『Replication Server 管理ガイド 第 1 巻』を参照してください。

HDS パラメータの要求ファンクション複写定義

パラメータ値のデータ型を変更するファンクション複写定義は作成できませんが、HDS データ型定義を使用して要求ファンクション複写定義のパラメータを宣言できます。宣言したパラメータは、クラスレベル変換の対象となります。

HDS の詳細については、『Replication Server 管理ガイド 第 1 巻』を参照してください。

ファンクション複写定義の変更

- パラメータまたはサーチャブルパラメータを既存の要求ファンクション複写定義に追加するには、**alter request function replication definition** を使用します。ファンクションに別のレプリケート名を指定することもできます。
- ファンクション複写定義内のパラメータを削除したり名前を変更したりするには、ファンクション複写定義のすべてのサブスクリプションを削除します。サブスクリプションを削除したら、ファンクション複写定義を削除して再作成します。

ファンクション複写定義のサブスクリプションの作成

要求ファンクション複写定義のサブスクリプションを作成するには、**without materialization** 句を指定した **create subscription** を使用するか、**define subscription** と、バルクマテリアライゼーションを含むその他のコマンドを使用します。

複数の複写定義の作成

- 1 つのプライマリファンクションに対して複数の要求ファンクション複写定義を作成し、それぞれ異なるレプリケートファンクションによってサブスクリプションを作成できるように各複写定義をカスタマイズできます。詳細については、『Replication Server 管理ガイド 第 1 巻』を参照してください。
- 1 つのプライマリファンクションに対して作成された各要求ファンクション複写定義では、同じ名前とデータ型の同じパラメータを使用する必要があります。
- 要求ファンクション複写定義のサブスクリプションを作成できるのは、バージョン 15.1 の Replication Server だけです。
- 1 つのプライマリファンクションは、適用ファンクション複写定義または要求ファンクション複写定義を持つことができますが、この両方を持つことはできません。**create function replication definition** コマンドを使用して作成されたファ

ンクション複写定義は、ファンクションが作成されたプライマリ Replication Server では適用ファンクションと見なされます。

- ウォームスタンバイデータベースでは、ストアドプロシージャはアクティブデータベースと同じ名前であるため、**with replicate function named** 句は無視されます。要求ファンクション複写定義のいずれかが **send standby replication definition parameters** 句を指定して作成されている場合、ファンクション複写定義で指定されたパラメータがスタンバイデータベースに配信されます。それ以外の場合は、プライマリファンクションのすべてのパラメータが配信されます。
- MSA 環境では、**send standby** 句を指定して作成したプライマリファンクションのファンクション複写定義が存在しない場合、レプリケートデータベースに配信されるファンクションには、プライマリファンクションと同じ名前が使用され、プライマリファンクションのすべてのパラメータが含まれます。それ以外の場合は、レプリケートデータベースに配信されるファンクションには、ファンクション複写定義の **with replicate function named** 句で指定された名前が使用され、同じファンクション複写定義で指定されたパラメータが含まれます。

パーミッション

create request function replication definition には、“create object” パーミッションが必要です。

参照：

- alter applied function replication definition (129 ページ)
- alter function string (188 ページ)
- alter request function replication definition (210 ページ)
- create applied function replication definition (274 ページ)
- create connection (287 ページ)
- create function string (318 ページ)
- define subscription (395 ページ)
- drop function replication definition (411 ページ)
- sp_setrepproc (673 ページ)
- rs_send_repserver_cmd (735 ページ)

create route

現在の Replication Server からリモート Replication Server へのコネクションに使用するルートを指定します。

構文

```
create route to dest_replication_server {
 set next site [to] thru_replication_server  |
 with primary at dataserver.database |
 [set username [to] user]
 [set password [to] passwd]
 [set route_param to 'value'
 [set route_param to 'value']... ]
 [set security_param to 'value'
 [set security_param to 'value']... ]}
```

パラメータ

- **dest_replication_server –** 送信先 Replication Server の名前です。
- **thru_replication_server –** 送信先 Replication Server へのメッセージが通過する中間 Replication Server の名前です。間接ルートを作成するときに指定します。
- **with primary –** 専用ルートを作成するプライマリデータベースからのコネクションを指定します。

  注意： 2 つの Replication Server 間に直接ルートがある場合、作成できるのはこれらサーバ間の 1 本の専用ルートだけです。Replication Server 間に間接ルートしかない場合、専用ルートは作成できません。
- **user –** 送信先 Replication Server へのログインに使用する Replication Server ログイン名です。これは、RSI ユーザスレッドが使用するログイン名です。ユーザ名が入力されない場合、Replication Server は、Replication Server の起動時に **-S** オプションを使用して入力されたプリンシパルユーザ名を使用します。
- **passwd –** このログイン名に使用するパスワードです。パスワードを指定しないと、Replication Server は null 値を使用します。
- **route_param –** ルートに影響するパラメータです。パラメータと値のリストについては、「ルートに影響を与える設定パラメータ」表を参照してください。
- **value –** パラメータの値を持つ文字列です。
- **security_param –** セキュリティパラメータの名前を指定します。表 35 : ネットワークベースセキュリティに影響を与えるパラメータを参照して、**create route** を使用して設定できるセキュリティパラメータのリストと説明を確認してください。

表 35 : ネットワークベースセキュリティに影響を与えるパラメータ

security_param	*値*
msg_confidentiality	Replication Server が暗号化データを送受信するかどうかを示す。"required" に設定すると、送信データが暗号化される。"not required" に設定すると、Replication Server は、送信されてくる暗号化データも非暗号化データも受け入れる。 デフォルト値は not_required
msg_integrity	データの改ざんに対するチェックを行うかどうかを示す。 デフォルト値は not_required
msg_origin_check	データの送信元を確認するかどうかを示す。 デフォルト値は not_required
msg_replay_detection	データが読み取られたり傍受されていないことを確認するために、データをチェックするかどうかを示す。 デフォルト値は not_required
msg_sequence_check	データに対して傍受のチェックを行うかどうかを示す。 デフォルト値は not_required
mutual_auth	コネクションが確立される前に、リモートサーバに身元の証明を要求する。 デフォルト値は not_required
security_mechanism	この経路で使用できるサードパーティのセキュリティメカニズムの名前。 デフォルト値は `libtcl.cfg` の SECURITY セクションで最初にリストされているメカニズム
unified_login	Replication Server がリモートデータサーバにログインし、受信ログインを受け入れる方法を示す。値は、次のとおり。 • required - リモートサーバには、必ずクレデンシャルを使用してログインする。 • not_required - リモートサーバには、必ずパスワードを使用してログインする。 デフォルト値は not_required

security_param	*値*
use_security_services	Replication Server に、セキュリティサービスを使用するかどうかを指示する。**use_security_services** が "off" の場合、セキュリティ機能は無効となる。 **注意：** このパラメータは **configure replication server** によってのみ設定できます。

例

- **例 1 –** このコマンドを TOKYO_RS Replication Server で入力すると、TOKYO_RS から SYDNEY_RS への直接ルートが作成されます。TOKYO_RS は、ログイン名 "sydney_rsi" とパスワード "sydney_rsi_ps" を使用して、このルート経由で SYDNEY_RS にログインできます。

```
create route to SYDNEY_RS
 set username sydney_rsi
 set password sydney_rsi_ps
```

- **例 2 –** TOKYO_RS でこのコマンドを入力すると、TOKYO_RS から中間 Replication Server である MANILA_RS を経由した SYDNEY_RS への間接ルートが作成されます。TOKYO_RS から MANILA_RS への直接ルートと、MANILA_RS から SYDNEY_RS への直接ルートがすでに存在している必要があります。

```
create route to SYDNEY_RS
 set next site MANILA_RS
```

- **例 3 –** このコマンドは、例 1 と同様の直接ルートを作成します。ただし、ネットワークベースセキュリティが有効になっている場合、TOKYO_RS はクレデンシャルを使用して SYDNEY_RS にログインする必要があります。

```
create route to SYDNEY_RS
 set unified_login 'required'
```

- **例 4 –** `NY_DS.pdb1` プライマリコネクションのために、RS_NY プライマリ Replication Server と RS_LON レプリケート Replication Server との間の専用ルートを作成するには、RS_NY で次のように入力します。

```
create route to RS_LON
 with primary at NY_DS.pdb1
 go
```

  特定のコネクションのために専用ルートを作成すると、そのコネクションから送信先 Replication Server へのトランザクションはすべて、その専用ルートを通るようになります。

使用法

- **create route** は、現在の Replication Server からリモート Replication Server への直接ルートまたは間接ルートを作成するときに使用します。
- ルートを作成する前に、全体のルート指定スキームを決定しておいてください。ルートの作成と管理については、『Replication Server 管理ガイド 第 1 巻』を参照してください。
- Replication Server は、複数のルートが同じ送信元 Replication Server から分岐し、同じ中間 Replication Server または送信先 Replication Server に収束するルート指定スキームはサポートしていません。
- Replication Server は、新しいルートに関する情報を、条件を満たしているサイトへ複写システムを介して分配します。複写システムで通常の遅延時間が発生するため、変更内容がレプリケートサイトにすぐに反映されるわけではありません。
- Replication Server が Embedded RSSD (ERSSD) で設定されている場合は、両方の Replication Servers が 15.0 以降であればルートを作成できます。作成しているルートが現在のサイトを始点とする最初のルートである場合は、ログ転送が開始され、Replication Agent が自動的に起動します。

  Replication Agent の名前を変更するには、次のように入力します。

```
configure replication server
set erssd_ra to 'value'
```

直接ルート

- 現在の Replication Server から送信先 Replication Server への直接ルートを設定するには、RSI ユーザ名とパスワードを指定し、**next site** 句を **create route** から省きます。
- 直接ルートを作成する前に、送信先 Replication Server にログイン名とパスワードを作成します。ログイン名とパスワードの設定には、**rs_init** を使用できます。デフォルトのユーザ名は *RS_name*_rsi"、デフォルトのパスワードは *RS_name*_rsi_ps です。

  送信先 Replication Server に存在しないユーザ名とパスワードを指定してルートを作成する場合は、その送信先でユーザ名とパスワードを追加または変更してください。

間接ルート

- Replication Server のメッセージに間接ルートを設定するには、**next site** 句を **create route** に指定します。たとえば、ニューヨークで発信され、ヨーロッパのすべてのサイトに配信されるメッセージは、間接ルートに沿ってロンドンのサイトを経由するようにルートを指定できます。間接ルートを使用すると、ルートの一部を経由して渡されるメッセージ量を削減できます。

- 間接ルートを作成する前に、まず送信元 Replication Server から中間 Replication Server への直接ルートと、中間 Replication Server から送信先 Replication Server への直接ルートを作成してください。
- 1 つのルートには、中間 Replication Server をいくつでも指定できますが、中間 Replication Server を追加するごとにプライマリサイトとレプリケートサイト間の遅延時間が長くなるため、中間サイト数は制限してください。

専用ルート

専用ルートを作成できるのは、以下の条件が満たされた場合だけです。

- プライマリ Replication Server から送信先 Replication Server への共有ルートが存在し、この共有ルートが直接ルートである。Replication Server 間に間接ルートしかない場合、専用ルートは作成できません。
- 共有ルートが有効であり、サスペンドされていない。
- 共有ルートのバージョンが 1570 以降である。

『Replication Server 管理ガイド 第 2 巻』>「パフォーマンスチューニング」>「Multi-Path Replication」の「専用ルート」を参照してください。

ルートと RSSD テーブル

- 直接ルートの作成時に指定する RSI ユーザ名とパスワードは、送信先 Replication Server の RSSD 内にある *rs_users* システムテーブルに追加されます。また、ユーザ名とパスワードは送信元 Replication Server の RSSD 内にある *rs_maintusers* システムテーブルにも追加されます。
- ルートを作成すると、送信元 Replication Server は、送信元 RSSD のプライマリユーザのログイン名とパスワードを送信先 Replication Server に送信します。送信先 Replication Server は、このログインを使用して、送信元 Replication Server の RSSD システムテーブルのいくつかにサブスクリプションを作成します。このプライマリユーザのログイン名は、通常、"*source_RSSD_name*_prim" と命名され、送信先 Replication Server で *rs_users* システムテーブル内に格納されます。

ネットワークベースセキュリティのパラメータ

- ルートの両端では、同じセキュリティメカニズムとセキュリティ機能を備えた互換性のある SCL (Security Control Layer) ドライバを使用してください。各サーバについて、セキュリティ機能の選択と設定を行うのは複写システム管理者の仕事です。Replication Server は、リモートサーバとのコネクションを確立しようとする前に、そのサーバのセキュリティ機能の問い合わせは行いません。
- **create route** は、現在の Replication Server がターゲット Relication Server にログインする方法と、メッセージを安全に転送する方法に影響を与えるネットワークベースのセキュリティ設定を指定します。
- **unified_login** を "required" に設定すると、sa ユーザだけがクレデンシャルなしで Replication Server にログインできます。セキュリティメカニズムに問題が発生

した場合でも、"sa" ユーザはパスワードを使用して Replication Server にログインし、**unified_login** を無効にできます。

- Replication Server には、複数のセキュリティメカニズムを装備できます。サポートされるメカニズムは、それぞれ `libtcl.cfg` ファイル内の SECURITY セクションにリストされています。
- メッセージの暗号化は、深刻なパフォーマンスの低下をともなう、負荷の高い処理です。通常は、特定のルートに対してだけ **msg_confidentiality** を "on" に設定してください。代わりに、**msg_integrity** などの負荷の低いセキュリティ機能を選択します。

パーミッション

create route には、"sa" パーミッションが必要です。

参照：

- alter connection (134 ページ)
- alter route (213 ページ)
- configure replication server (238 ページ)
- create connection (287 ページ)
- drop connection (407 ページ)
- drop route (421 ページ)

create schedule

シェルコマンドを指定時刻に実行するスケジュールを作成します。

構文

```
create schedule sched_name as 'sched_time'
[set {on | off}] for exec 'command'
```

パラメータ

- **sched_name** – 指定されたスケジュールの名前。以下の条件を満たす必要があります。
  - 名前は識別子の規則に従う必要がある。
  - ユニークでなければならない。
  - 長さは 1 ～ 30 バイト。
- **sched_time** – '*command*' コマンドを実行する時刻と日付。時刻と日付のパラメータ間にスペースを 1 つ挿入して区切る、制限された UNIX cron 形式の日付と時刻を示します。

```
[mm] [HH] [DOM] [MON] [DOW]
```

時刻と日付の各パラメータは、以下のとおりです。

パラメータ	説明	値
mm	正時から経過した分数。	0 - 59。有効値をすべて含めるには、“*” を使用します。
HH	24 時間表記での時間。	0 - 23。有効値をすべて含めるには、“*” を使用します。
DOM	日	1 - 31。日をすべて含めるには、“*” を使用します。
MON	月	1 - 12。月をすべて含めるには、“*” を使用します。
DOW	曜日	0 - 6 で、0 = 日曜日。曜日をすべて含めるには、“*” を使用します。

- 有効値をすべて指定するには、アスタリスク “*” を使用します。たとえば、“17 20 * * *” は毎日のスケジュールで午後 8:17 を意味します。
- 範囲外の値を区切るには、カンマ “,” を使用します。たとえば、“17 20 1,15 * *” は毎月 1 日と 15 日の午後 8:17 を表します。ただし、1 と 15 は *DOM* パラメータの値です。
- 値の範囲 (両端を含む) を指定するには、ハイフン “-” を使用します。たとえば、“17 20 * * 1-5” は月曜日から金曜日の午後 8:17 を表します。ただし、"1-5" は *DOW* パラメータの値の範囲です。
- *DOM*、*MON*、または *DOW* パラメータの場合、*DOM* と *DOW* の両方のパラメータを使用して日を指定できます。Replication Server は、文字列で指定されたスケジュールすべてに従います。たとえば、“0 12 16 * 1” は毎週月曜日の午後 12:00 および毎月 16 日の午後 12:00 を表します。

- **set [on | off}** – スケジュール作成時に、有効と無効を指定します。デフォルトでは、スケジュールは on です。
- **command** – シェルコマンド (スクリプトなど) は、実行可能ファイルか、指定スケジュールで実行されるバッチファイルです。一重引用符で囲みます。

  シェルコマンドには、以下の特徴があります。

  - UNIX では `$SYBASE/$SYBASE_REP/sched`、Windows では `%SYBASE%¥%SYBASE_REP%¥sched` に存在しなければならない。
  - シェルコマンド内にスペースで区切ったパラメータを複数含めることができる。

- Windows では、**create schedule** はシェルコマンドまたはバッチファイル内の最後のパラメータで指定されたコマンドを実行します。さらに、**stdout** を **create schedule** コマンドラインのファイルに含めてください。

シェルコマンド名には、以下の特徴があります。

- ASCII 英数字文字だけを使用できる (A ～ Z、a ～ z、および 0 ～ 9)
- “.”、“-”、および “_” 文字を使用できる。
- “¥” と “/” 文字を使用できない。
- Windows 上で実行する場合、`.bat` サフィックスを含める必要がある。たとえば、名前は Windows では `suspend_conn.bat`、UNIX では `suspend_conn.sh` としてください。

例

- **例 1 –** SYDNEY_DS データサーバの *pubs2* データベースへのコネクションを毎週月曜日の午後 12:00 と毎月 16 日の午後 12:00 にサスペンドする “schedule1” を Windows で作成します。

  1. テキストファイル `sql.txt` を作成し、これにスケジュールする実際の Replication Server コマンドラインを含めます。たとえば、`sql.txt` に以下を含めることができます。

     ```
     suspend connection to SYDNEY_DS.pubs2
     go
     ```

  2. Windows で、バッチファイル `suspend_conn.bat` を作成し、これに **isql** および `sql.txt` のコマンドラインを実行するための該当するパラメータを含めます。たとえば、`suspend_conn.bat` に以下を含めることができます。

     `%SYBASE%¥OCS-15_0¥bin¥isql.exe` **-Usa-P-S** SYDNEY_DS **-I**`%SYBASE%¥sql.ini` **-i**`%SYBASE%¥REP-15_5¥sched¥sql.txt`

  3. スケジュール “schedule1” を作成します。

     ```
     create schedule schedule1 as '0 12 16 * 1' for exec
     'test.bat > c:¥temp¥test.out'
     go
     ```

- **例 2 –** UNIX で、`suspend_conn.sh` スクリプトを実行するための “schedule2” を作成します。このスケジュールは、SYDNEY_DS データサーバの *pubs2* データベースへのコネクションを毎日午後 8:17 にサスペンドします。

  ```
  create schedule schedule2 as '17 20 * * *' for exec
  'suspend_conn.sh'
  ```

- **例 3 –** `resume_conn.sh` スクリプトを実行するための “schedule2” を作成します。このスケジュールは、SYDNEY_DS データサーバの *pubs2* データベースへのコネクションを毎日午前 7:15 にレジュームします。

```
create schedule schedule2 as '15 7 * * *' for exec
'resume_conn.sh'
```

使用法

- レプリケーションタスクを実行する時刻をスケジュールします。たとえば、レプリケートデータベースがフリーズしており、プライマリデータベースからデータを受信していないときのレプリケートデータベースの特定のステータスに関するレポートを作成できます。
- レプリケーションが夜中の指定された期間にのみ行われるようにスケジュールを設定することによって、次の日の処理でレプリケートデータベースが変更されないようにし、レプリケートデータベースでフリーズされた前の日のデータに対してレポートが行われるようにすることができます。これは、レプリケートデータベースへのコネクションが 1 日の特定の時刻にサスペンドおよびレジュームされるようにスケジュールを設定することによって行うことができます。

パーミッション

create schedule には、"sa" パーミッションが必要です。

参照：

- admin schedule (69 ページ)
- alter schedule (221 ページ)
- drop schedule (423 ページ)

create subscription

サブスクリプションを作成して初期化し、サブスクリプションデータをマテリアライズします。サブスクリプションは、データベース複写定義、テーブル複写定義、ファンクション複写定義、またはパブリケーションに対して作成できます。

構文

```
create subscription sub_name
for {table_repdef | func_repdef | publication pub |
     database replication definition db_repdef }
     [ with primary at server_name.db ]
with replicate at data_server.database
[where {column_name | @param_name}
     {< | > | >= | <= | = | &} value
[and {column_name | @param_name}
     {< | > | >= | <= | = | &} value]...]
[without holdlock [direct_load [init replicate table with {create |
create_or_truncate | truncate | recreate}]
[user username password pass][num_of_selects selects]
```

```
[hold_resource_on_error]]| incrementally | without materialization]
[subscribe to truncate table]
[for new articles]
```

パラメータ

- **sub_name –** サブスクリプションの名前です。この名前は識別子の規則に従う必要があります。サブスクリプションの名前は、複写定義 (適用される場合) と、レプリケートデータサーバおよびデータベースに対してユニークでなければなりません。
- **for table_rep_def –** サブスクリプションの対象となるテーブル複写定義を指定します。
- **for function_rep_def –** サブスクリプションの対象となるファンクション複写定義の名前を指定します。
- **for publication pub_name –** サブスクリプションの対象となるパブリケーションを指定します。
- **for database replication definition db_repdef –** サブスクリプションの対象となるデータベース複写定義を指定します。
- **with primary at data_server.database –** プライマリデータのロケーションを指定します。 プライマリデータベースが論理コネクションを使用するウォームスタンバイアプリケーションの一部である場合、*data_server.database* は論理データサーバと論理データベースの名前になります。 Multi-Path Replication システムを設定している場合、句内に代替プライマリコネクション名を指定することもできます。
- **with replicate at data_server.database –** レプリケートデータのロケーションを指定します。 レプリケートデータベースがウォームスタンバイアプリケーションの一部である場合、*data_server.database* は論理データサーバと論理データベースの名前になります。 Multi-Path Replication システムを設定している場合、句内に代替レプリケートコネクション名を指定することもできます。 **direct_load** オプションを使用して作成したテーブルサブスクリプションの場合、これは物理データサーバの名前とデータベースの名前になります。論理接続または代替接続の名前は使用できません。
- **where –** サブスクリプションによって複写されるカラムまたはパラメータの値に対する基準を設定します。**where** 句を省略すると、すべてのローまたはパラメータが複写されます。

  where 句は、テーブル複写定義またはファンクション複写定義のサブスクリプションに指定できます。データベースまたはパブリケーションのサブスクリプションには、**where** 句は使用できません。

  where 句は、1 つ以上の単純比較で構成されます。単純比較では、次に示す関係演算子のいずれかを使用して、複写定義のサーチャブルカラムまたはサーチャブルパラメータがリテラル値と比較されます。**<**、**>**、**<=**、**>=**、**=**、または

& (**&** 演算子は、*rs_address* データ型のカラムまたはパラメータでのみサポートされます)。また、キーワード **and** を使用して、比較を結合できます。

式で使用されるカラム名またはパラメータ名は、テーブル複写定義の **searchable columns** リストまたはファンクション複写定義の **searchable parameters** リストに含まれている必要があります。

Java カラムはサブスクリプションの式では評価できません。このため、**where** 句には、*rawobject* または *rawobject in row* などのタイプの Java カラムを指定することはできません。

サブスクリプションの **where** 句の最大サイズは 255 文字です。

注意： 7 バイト未満のバイナリを整数に変換することはできません。 対処方法として、バイナリ値に 0 を埋め込んで 8 バイトにするか、バイナリ値ではなく整数値を使用してください。

- **column_name –** テーブル複写定義のサブスクリプションに使用するプライマリテーブルのカラム名です。
- **@param_name –** ファンクション複写定義に対するサブスクリプション用の複写ストアドプロシージャからのパラメータ名です。
- **value –** 指定したカラムまたはパラメータの値です。各種データ型の値の入力フォーマットについては、「データ型」を参照してください。

  式で使用されるカラム名またはパラメータ名は、複写定義の **searchable columns** リストまたは **searchable parameters** リストに含まれている必要があります。
- **without holdlock –** ノンアトミックマテリアライゼーションの場合、ホールドロックを使用しないでプライマリデータベースからデータを選択します。ローは、トランザクションごとに 1000 ローずつ挿入していく方法で、レプリケートデータベースに適用されます。 **direct_load** オプションを使用して作成したサブスクリプションについては、適用されるロー数は **mat_load_tran_size** 設定パラメータによって決まります。 詳細については、「ノンアトミックマテリアライゼーション」を参照してください。
- **init replicate table with create –** レプリケートデータベースでテーブルを作成します。 テーブルがすでに存在する場合、マテリアライゼーションは失敗します。
- **init replicate table with create_or_truncate –** レプリケートデータベースでテーブルを作成します。 テーブルがすでに存在する場合、Replication Server はトンランケーション後、既存のテーブルを使用します。
- **init replicate table with truncate –** レプリケートデータベースでテーブルをトランケートします。 テーブルが存在しない場合、マテリアライゼーションは失敗します。

- **init replicate table with recreate –** レプリケートデータベースでテーブルを削除して、テーブルを再作成します。

  注意： レプリケートテーブルがまだ存在してなくても、マテリアライゼーションは失敗しません。

- **incrementally –** サブスクリプションを初期化し、トランザクションごとに 1000 ローずつ挿入する方法でサブスクリプションデータを適用します。アトミックマテリアライゼーションの場合、プライマリデータベースでホールドロックが使用されます。
- **without materialization –** サブスクリプション用にデータをマテリアライズしません。このオプションは、プライマリデータベースにアクティビティがなく、レプリケートデータベース内にすでにデータが存在する場合に使用します。または、プライマリデータベース内でアクティビティをサスペンドしており、そのデータを手動でレプリケートデータベースに転送した場合に使用します。データベースサブスクリプションには、このオプションを指定する必要があります。
- **subscribe to truncate table –** テーブル複写定義、データベース複写定義、またはパブリケーションのサブスクリプションに対して、サブスクリプションを作成するレプリケートデータベースへの **truncate table** コマンドの複写を有効にします。

  このオプションは、特定のデータベースの同じレプリケートテーブルにデータを複写する他の既存のサブスクリプションと同じように設定してください。そうしないと、新しいサブスクリプションは拒否されます。
- **for new articles –** 既存のサブスクリプションをリフレッシュします。サブスクリプションをパブリケーションと照合し、サブスクリプションが作成されていないアーティクルに対してサブスクリプションを作成するように Replication Server に指示します。
- **direct_load –** 直接ロードマテリアライゼーションを有効化します。

  このオプションを使用すると、同じレプリケートテーブルに対して同時に他のサブスクリプションを作成できません。 このオプションは、物理データベース接続に対してのみ使用できます。代替接続や論理接続には使用できません。

  このオプションが使用できるのは、レプリケート SAP Replication Server のサイトバージョンとそのプライマリ SAP Replication Server に対するルートバージョンが 1571100 以降の場合のみです。
- **user *username* password *pass* –** プライマリ Adaptive Server database または Replication Agent への接続用の直接ロードマテリアライゼーションで使用されるユーザ ID とパスワード。プライマリテーブルから選択します。
- ***num_of_selects*selects –** direct_load オプションを指定して作成するサブスクリプションのマテリアライゼーションパフォーマンスを向上させるために、複数の選択スレッドを有効にします。 デフォルト値は 1 です。有効な値は 1 ～ 10 で

す。 このオプションを使用できるのは、**direct_load** オプションを指定する場合のみです。また、プライマリデータベースが DB2 UDB 9.7 以降か Oracle のどちらかであり、ファンクション文字列 **rs_select** がカスタマイズされていない場合にしか使用できません。 これらの条件を満たさない場合や、プライマリテーブルのロー数があまり多くない場合、この数値は 1 に下げられます。

- **hold_resource_on_error** – 直接ロードマテリアライゼーションで作成したサブスクリプションがエラーになった場合、サブスクリプションリソースを保持します。デフォルトでは、**direct_load** オプションで作成したサブスクリプションがエラーになってもリソースを保持しません。

例

- **例 1 –** *titles_sub* という名前のサブスクリプションを作成します。このサブスクリプションは、タイプが "business" であるカラムを持つ *titles* テーブルのローを、SYDNEY_DS というデータサーバの *pubs2* データベースにある *titles* テーブルに複写することを指定しています。

```
create subscription titles_sub
 for titles_rep
 with replicate at SYDNEY_DS.pubs2
 where type = 'business'
```

- **例 2 –** 10.00 ドル以上の価格が指定された *titles* テーブルのローを含む *titles_sub* というサブスクリプションを作成します。

```
create subscription titles_sub
 for titles_rep
 with replicate at SYDNEY_DS.pubs2
 where price >= $10.00
```

- **例 3 –** ファンクション複写定義 *myproc_rep* の *myproc_sub* というサブスクリプションを作成します。このコマンドを使用してファンクション複写定義のサブスクリプションを作成するには、データがレプリケートデータベースにすでに存在している必要があります。また、**without materialization** 句を使用します。

```
create subscription myproc_sub
 for myproc_rep
 with replicate at SYDNEY_DS.pubs2
 without materialization
```

- **例 4 –** パブリケーション *pubs2_pub* の *pubs2_sub* というサブスクリプションを作成します。

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

- **例 5 –** データベースサブスクリプション *pubs2_sub* (データベース複写定義 *pubs2_rep*) を作成します。

```
create subscription pubs2_sub
 for database replication definition pubs2_rep
   with primary at NEWYORK_DS.pubs2
   with replicate at TOKYO_DS.pubs2
 without materialization
 subscribe to truncate table
```

- **例 6 – sub_conn2** サブスクリプションを **repdef_conn2** 複写定義のために作成します。これは、NY_DS.rdb_conn2 代替レプリケートコネクション上にあります。

```
create subscription sub_conn2 for repdef_conn2
with replicate at NY_DS.rdb_conn2
without materialization
go
```

- **例 7 – sub_conn2** サブスクリプションを **repdef_conn2** 複写定義に対して作成します。これは、NY_DS.rdb がデフォルトのレプリケートコネクションである LON_DS プライマリデータサーバへの LON_DS.pdb_conn2 代替プライマリコネクション上にあります。

```
create subscription sub_conn2 for repdef_conn2
with primary at LON_DS.pdb_conn2
with replicate at NY_DS.rdb
without materialization
go
```

- **例 8 –** 10.00 ドル以上の価格が指定された *titles* テーブルのローを含む *titles_sub* というサブスクリプションを **direct load** オプションを指定して作成します。

```
create subscription titles_sub
 for titles_rep
 with replicate at SYDNEY_DS.pubs2
 where price >= $10.00
 without holdlock
 direct_load
```

  ユーザ ID とパスワードは明示的に指定されないので、サブスクリプションの作成に使用されたユーザ ID とパスワードが、プライマリデータベースまたは Replication Agent へのログイン、およびプライマリテーブルからのデータの選択に使用されます。 プライマリデータベースが Adaptive Server でない場合、Replication Server はユーザ ID が Replication Agent 管理ユーザのものであると想定します。

- **例 9 –** 売り上げが 5,000 コピーを超える *titles* テーブルのローを含む *titles_sub* というサブスクリプションを **direct load** オプションを指定して作成します。

```
create subscription titles_sub
 for titles_rep
 with replicate at SYDNEY_DS.pubs2
 where total_sales > 5000
 without holdlock
 direct_load
 user pubs2_owner password pubs2_owner_pwd
```

Replication Server は pubs2_owner ユーザ ID と pubs2_owner_pwd パスワードを使用して Adaptive Server プライマリデータベースまたは Replication Agent にログインし、**select** をプライマリデータベーステーブルに発行します。

- **例 10 – direct_load** オプションを指定し、最大 3 つの選択スレッドを指定して、**authors_sub** という名前のサブスクリプションを作成します。

```
create subscription authors_sub for authors_repdef
       with replicate at ost_replnxb10_01.rdb1
      without holdlock direct_load 3 selects
```

ユーザ ID とパスワードは明示的に指定されないので、サブスクリプションの作成に使用されたユーザ ID とパスワードが、プライマリデータベースまたは Replication Agent へのログイン、およびプライマリテーブルからのデータの選択に使用されます。 プライマリデータベースが Adaptive Server でない場合、Replication Server はユーザ ID が Replication Agent 管理ユーザのものであると想定します。

使用法

- ファンクション複写定義またはデータベース複写定義をサブスクライブするには、**create subscription** を使用する (**without materialization** 句を指定) か、**define subscription** やその他のバルクマテリアライゼーションコマンドを使用します。
- **create subscription** は、複写データを格納するデータベースの Replication Server で実行します。
- サブスクリプションの詳細とレプリケーションにおけるサブスクリプションの役割については、『Replication Server 管理ガイド 第 1 巻』を参照してください。
- Replication Server 15.5 以降では、テーブルサブスクリプションを含む複写定義を変更した場合、そのテーブルサブスクリプションを削除して再作成する必要はありません。
- 同じプライマリテーブルまたはデータベースに対して、複数の複写定義を作成できます。 同じ複写定義のサブスクリプションを複数回作成することはできますが、同じレプリケートテーブルまたはレプリケートデータベースの複数の複写定義に対してサブスクリプションを作成することはできません。
- Multi-path replication の場合、プライマリデータベースと Replication Server との間のすべてのプライマリコネクションが、すべての複写定義を共有しているので、どのプライマリコネクションがデータソースであり、どのレプリケートコネクションが複写先なのかをサブスクリプションで指定してください。『Replication Server 管理ガイド 第 2 巻』の「パフォーマンスチューニング」の「Multi-Path Replication」を参照してください。
- LOB 圧縮データのサブスクリプションマテリアライゼーションのサポートは、複写定義でのカラムのデータ型の指定方法、また、Replication Server のバージョンによって異なります。 『Replication Server 管理ガイド 第 1 巻』の「LOB

圧縮データのサブスクリプションマテリアライゼーション」を参照してください。

- 引用符付き定数が含まれるカスタムファンクション文字列とともに引用符付き識別子を使用する場合は、引用符付き定数なしで、または **without materialization** 句を指定せずに **create subscription** を実行します。 それ以外の場合、サブスクリプションマテリアライゼーション時に引用符付き定数が原因でクエリが失敗します。 レプリケートデータサーバは、引用符付き定数を定数ではなくカラムとして認識します。

データベース複写定義のサブスクリプションの作成

- データベースサブスクリプションを作成する場合、**where** 句を指定してデータのサブスクリプションを制限することはできません。すべてのデータがサブスクリプションの対象となります。
- データベースサブスクリプションでは、非マテリアライゼーションメソッドまたはバルクマテリアライゼーションメソッドだけを使用できます。ダンプとロードを使用したり、その他のバルクマテリアライゼーションメソッドを使用するには、**define subscription** コマンドを使用します。非マテリアライゼーションメソッドを使用するには、**create subscription** コマンドを使用します。
- 同じオリジンから複数のデータベース複写定義に対してサブスクリプションを作成することはできません。
- Replication Server のバージョンがプライマリ Replication Server のバージョンより低い場合は、プライマリ Replication Server によって制御されるプライマリデータベースのレプリケート Replication Server に対するデータベースサブスクリプションを作成できません。
- データベースサブスクリプションによってサブスクリプションが作成されたプライマリデータベースに対してテーブル複写定義を正常に作成するには、レプリケート Replication Server のバージョンはテーブル複写定義のバージョンと同じかそれ以上である必要があります。

パブリケーションのサブスクリプションの作成

- パブリケーションが有効であれば、レプリケートデータベースへの複写を開始するために、パブリケーションのサブスクリプションを作成できます。すべての形式のサブスクリプションマテリアライゼーションがサポートされています。
- パブリケーションサブスクリプションを作成すると、Replication Server は、パブリケーションに組み込まれたアーティクルごとに、別々の基本サブスクリプションを作成します。各アーティクルサブスクリプションは、親パブリケーションサブスクリプションの名前を使用します。
  - アトミックマテリアライゼーションまたはノンアトミックマテリアライゼーションを使用する場合、アーティクルサブスクリプションは、その

アーテイクルがパブリケーションに追加された順番で、一度に 1 つずつマテリアライズされます。
  - **create subscription** を使用する (**without materialization** 句を指定) と、すべてのアーティクルサブスクリプションが同時にアクティブ化および確定化されます。
- パブリケーションのサブスクリプションには、**where** 句を指定することはできません。その代わりに、パブリケーションに組み込まれた各アーティクル内に 1 つまたは複数の **where** 句を指定することによって、レプリケートサイトへのレプリケーションをカスタマイズできます。

HDS 変換の対象となるカラムの指定

- **where** 句を指定したサブスクリプションを作成する場合は、**where** 句の値が、宣言したデータ型のフォーマットで比較されなければなりません。
- クラスレベル変換またはカラムレベル変換の対象となるカラムを **where** 句で指定するサブスクリプションは、自動的にマテリアライゼーション解除することはできません。バルクマテリアライゼーションメソッドまたは非マテリアライゼーションメソッドを使用する必要があります。

truncate table のレプリケーション

- 最初のサブスクリプションを作成する場合、**subscribe to truncate table** オプションは指定しても指定しなくてもかまいません。同じテーブル内に複写する後続の各サブスクリプションは、最初のサブスクリプションの例に従う必要があります。最初のサブスクリプションに従わないと、サブスクリプションを作成しようとしたときに拒否されます。
- **sysadmin apply_truncate_table** を実行すると、特定のレプリケートテーブルの現在の "subscribe to truncate table" ステータスを変更できます。

create subscription を実行するための必要条件

- **create subscription** を実行する前に、後述するパーミッションの他に、次の条件を満たしているかどうかを確認してください。
  テーブル複写定義に対するサブスクリプションの場合
  - 複写するプライマリテーブルに対して複写定義が存在し、そのテーブルが **sp_setreptable** を使用して複写用にマーク付けされている。
  - **sp_reptostandby** を使用してマーク付けされたテーブルに対してサブスクリプションを作成する場合は、**rep_as_standby** 設定パラメータを使用してプライマリデータベース接続を設定し、**send_warm_standby_exacts** を使用して Replication Agent を設定しなければなりません。
  - 複写定義内で参照されるテーブルが、プライマリデータベースとレプリケートデータベースの両方に存在する。また、各テーブルには、複写定義内で指定されたカラムとデータ型が存在する。

このテーブルは、サブスクリプションを作成するユーザと管理するユーザから参照可能である必要があります。これを実現する最も簡単な方法は、データベース所有者にテーブルを作成させることです。

ファンクション複写定義に対するサブスクリプションの場合

- 複写するストアドプロシージャに対して複写定義が存在し、そのストアドプロシージャが **sp_setrepproc** を使用して複写用にマーク付けされている。
- ファンクション複写定義内で参照されるストアドプロシージャが、プライマリデータベースとレプリケートデータベースの両方に存在する。また、各ストアドプロシージャには、ファンクション複写定義内で定義されたパラメータとデータ型が存在する。

パブリケーションに対するサブスクリプションの場合

- 複写するプライマリテーブルまたはストアドプロシージャのアーティクルを含むパブリケーションが存在する。このアーティクルは、上記の条件を満たす複写定義を指定する。
- パブリケーションが有効である。

ウォームスタンバイアプリケーションの動作条件

- 次の動作条件は、ウォームスタンバイアプリケーション内でサブスクリプションを作成する場合に適用されます。
  - 送信先データベースがウォームスタンバイアプリケーションの一部である場合は、テーブルがアクティブデータベースとスタンバイデータベースの両方に存在しなければならない。両方のテーブルは、**sp_setreptable** または **sp_reptostandby** を使用して複写用にマーク付けされている必要があります。
  - 論理プライマリデータベースでは、Replication Server がスタンバイデータベースの追加処理を行っている間はサブスクリプションを作成できない。
  - **direct_load** オプションを指定して、ウォームスタンバイ接続に対してテーブルサブスクリプションを作成することはできません。

同じ名前の付いたテーブルの動作条件

- プライマリ Adaptive Server データベースに、複写テーブルと、それと同じ名前の付いた別のテーブルが含まれる場合、2 番目の (複写テーブルではない) テーブルの所有者は、カスタム **rs_select** または **rs_select_with_lock** ファンクション文字列を使用せずに複写テーブルへのサブスクリプションを作成することはできません。例：
  - *db.dbo.table1* という名前のプライマリテーブルに対する複写定義が存在し、さらに
  - データベースユーザ "jane" が *db.jane.table1* という名前のテーブルを所有している場合は、
  - Jane はデフォルトのファンクション文字列を使用して *db.dbo.table1* の複写定義へのサブスクリプションを作成できません。

アトミックマテリアライゼーション

- このコマンドを使用した、サブスクリプションのマテリアライゼーションを行うためのデフォルトメソッドは、アトミックマテリアライゼーションです。アトミックマテリアライゼーションは、プライマリテーブルをロックし、単一のアトミックオペレーションでネットワークを通じてサブスクリプションデータをコピーします。
- アトミックマテリアライゼーションの実行中は、プライマリデータベースで select トランザクションが完了するまで、レプリケートデータベースにローは表示されません。サブスクリプションが多数のローを指定すると、select トランザクションの実行時間が長くなり、これによってレープリケートサイトで遅延が生じます。

アトミックマテリアライゼーションを使用する場合の必要条件

- サブスクリプションマテリアライゼーションのアトミックメソッドを使用する場合は、次の条件が適用されます。
  - ユーザまたはデータベース所有者が、プライマリテーブルを所有していなければならない。または、ユーザが、プライマリデータベースでの **select** オペレーションにユーザ定義ファンクション文字列を使用しなければならない。
  - データベース所有者またはメンテナンスユーザが、レプリケートテーブルを所有していなければなりません。または、ユーザが、レプリケートデータベースでの **select** オペレーションにユーザ定義ファンクション文字列を使用しなければなりません。 レプリケートテーブルの所有者がプライマリテーブルの所有者と異なる場合は、別のファンクション文字列クラスを使用して、ユニークなファンクション文字列を作成する必要があります。プライマリデータベースは、Adaptive Server データベースであることが必要です。

without holdlock または incrementally オプションの使用

- **without holdlock** オプションまたは **incrementally** オプションは、サブスクリプションマテリアライゼーションのデフォルトのアトミックメソッドに代わるものです。これらのオプションを指定すると、Replication Server ではバッチ単位でローが適用されるため、データは一度に 1 つのバッチずつレプリケートデータベースに表示されます。
  この結果、マテリアライゼーション中は、レプリケートデータベースのクエリによってサブスクリプションの不完全なデータが返されることがあります。**check subscription** がサブスクリプションが有効であることを示すと、この一時的な状態は終了します。

incrementally オプション

- **incrementally** オプションは、アトミックマテリアライゼーションの一種です。このオプションを大きなサブスクリプションに対して使用すると、レプリケートデータベースで長時間トランザクションが実行されるのを防ぐことができま

す。サブスクリプションデータはレプリケートデータベースでアトミックに適応されないので、データは使用可能になっても、マテリアライゼーションが完了してサブスクリプションが確定化されるまでは完全ではありません。

- **incrementally** を使用すると、**select** は**ホールドロックを使用して**実行されるため、プライマリデータベースの順序一貫性が保持されます。レプリケートテーブルは、プライマリデータベースで以前に発生したステータスを渡します。
  すべての場合において、マテリアライゼーションが完了し、**check subscription** によってサブスクリプションが有効であると示されれば、レプリケートデータはプライマリデータベースと一貫性が保持されています。

ノンアトミックマテリアライゼーション

- **without holdlock** オプションは、ノンアトミックマテリアライゼーションを使用します。このオプションが指定されていると、ホールドロックを使用せずにプライマリデータベースからマテリアライゼーションローが選択されます。そのため、選択後にプライマリデータベースでローが更新されると、不整合が発生する可能性があります。不整合を修正するには、**set autocorrection on** を使用して、**without holdlock** を使用します。
- データがすでにレプリケートデータベースに存在する場合は、バルクマテリアライゼーションの代わりに、アトミックマテリアライゼーションまたはノンアトミックマテリアライゼーションを使用できます。

ノンアトミックマテリアライゼーションを使用する場合の必要条件

- サブスクリプションマテリアライゼーションのノンアトミックメソッドを使用する場合は、次の条件が適用されます。
  - プライマリデータベースから適用ファンクションを分配することによってデータを更新したり、交換関数を使用してデータを更新したりする場合は、**without holdlock** を使用しません。 たとえば、ストアドプロシージャが、あるカラムの以前の値を増やすことによってローを更新する場合、マテリアライゼーションが完了したときに値が不正確になる可能性があります。
  - ノンアトミックサブスクリプションの場合、**switch active** の実行時にノンアトミックサブスクリプションがマテリアライズしていると、SUSPECT とマーク付けされる。

注意： アトミックまたはノンアトミックマテリアライゼーション・メソッドのいずれかとともに **create subscription** を使用しており、かつ複写定義に引用符付き識別子がある場合、引用符付き識別子を使用できるようプライマリコネクションを変更してください。

直接ロードマテリアライゼーション

- **direct_load** オプションは、HANA DB へのデータのマテリアライズにのみ使用できます。

- **direct_load** を使用するには、適切な接続プロファイル (**rs_rs_to_msss_ra**、**rs_rs_to_oracle_ra**、または **rs_rs_to_udb_ra**) を使用して、Replication Server から Adaptive Server 以外のプライマリデータベースに接続する必要があります。
- **direct_load** オプションを使用すると、同じレプリケートテーブルに対して同時に他のサブスクリプションを作成できません。
- 他の自動マテリアライゼーションメソッドとは異なり、**select** コマンドから返されるとすぐにプライマリテーブルから選択されたデータがレプリケートデータベースに直接ロードされます。このときに非マテリアライゼーションキューが使用されます。
- **direct_load** オプションはテーブル複写定義へのサブスクリプション専用であり、**without holdlock** とともに使用します。 **without materialization** や **incrementally** とは併用できません。
- キャッチアップキューは、サブスクリプションマテリアライゼーションが進行中のプライマリテーブルに対する DML オペレーションを保持します。これにより、すでにレプリケート中のほかのテーブルに対するレプリケーションを中断する必要がなくなります。 プライマリテーブルから選択されたデータがレプリケートテーブルに適用された後、キャッチアップキュー内の DML オペレーションがレプリケートテーブルに適用されます。 キャッチアップキュー内のオペレーションに対するサブスクリプションアクティブ化マーカー、およびサブスクリプション検証マーカーの間で、オートコレクションが適用されます。

  注意： キャッチアップキューの DML 操作がレプリケートテーブルに適用されると、**insert** 操作のそれぞれが **delete** 操作とそれに続く **insert** 操作に変換されます。 更新でプライマリキーが変更されると、マテリアライゼーションは失敗します。

- **user** オプションと **password** オプションは、**direct_load** とのみ一緒に使用できます。 ユーザ ID には、プライマリ Adaptive Server データベースまたは Replication Agent にログインするためのパーミッション、およびマテリアライズ対象のプライマリデータベーステーブルに対する **select** パーミッションが必要です。 *username* と *pass* の値は一度だけ使用され、RSSD には保存されません。

  注意： プライマリデータベースが Adaptive Server でない場合、Replication Server はプライマリデータベースに接続し、Replication Agent を介して **select** を実行できます。 この場合、**create connection** コマンドで使用されるユーザ ID とパスワードは、Replication Agent への接続に必要なものであり、プライマリデータベースへの接続に必要なものではありません。

  このオプションに値を指定しない場合は、サブスクリプションの作成に使用されたユーザ ID とパスワードが、Adaptive Server データベースまたは Replication Agent へのログオン、およびプライマリテーブルからの選択に使用されます。

- *num_of_select***selects** オプションは、**direct_load** のみと組み合わせて使用され、マテリアライゼーションのパフォーマンスを高めるために複数の選択スレッドを使用可能にします。 このオプションのデフォルトは最小値の 1 であり、最大値は 10 です。 複数の選択スレッドは、ファンクション文字列 **rs_select** がカスタマイズされていない場合に、IBM DB2 および Oracle のプライマリデータベースでのみサポートされます。 これらの条件を満たさない場合や、プライマリテーブルのロー数があまり多くない場合、この数値は 1 に下げられます。
- 論理接続や代替接続に対しては **direct_load** オプションを使用できません。 複写定義でのプライマリ接続とサブスクリプションでのレプリケート接続は物理接続であることが必要です。
- プライマリデータベースが Adaptive Server でなく、**without materialization** オプションを指定せずにサブスクリプションが作成されている場合は、**direct load** オプションを使用する必要があります。
- **direct_load** オプションが使用できるのは、レプリケート Replication Server のサイトバージョンとルートバージョンが 1571100 以降の場合のみです。
- **direct_load** オプションを使用して作成されたサブスクリプションでは、ローフィルタリング、名前マッピング、カスタムファンクション文字列、およびデータ型マッピングを使用できます。
- プライマリデータベース接続のデータサーバ名が、**create subscription** の発行元である Replication Server の `interfaces` ファイルに存在している必要があります。

非マテリアライゼーション

without materialization 句は、非マテリアライゼーションメソッドを指定します。これは、サブスクリプションデータがレプリケートデータベースにすでに存在する場合にサブスクリプションを作成するための便利な方法です。

非マテリアライゼーションの必要条件

- サブスクリプションデータがレプリケートデータベースに、すでに存在している必要があります。
- プライマリデータベースとレプリケートデータベースが同期している必要があります。
- Replication Server のステーブルキュー内にこれ以上更新が発生しないように、プライマリデータベースでのアクティビティを停止しておく必要があります。

rs_address データ型の使用

- カラムまたはパラメータが特殊なデータ型 *rs_address* を使用する複写定義に対して、サブスクリプションを作成できます。 このデータ型では、独自のサブスクリプション解析メソッドを使用できます。この解析メソッドによって、*rs_address* データ型 (基本となるデータ型は *int* データ型) のビットマップがサブスクリプションの **where** 句のビットマスクと比較されます。 このビットマップ

の比較によって、レプリケートサイトが各ローのデータを受け取るかどうかがプライマリ Replication Server に通知されます。

- *rs_address* データ型のカラムまたはパラメータの場合にのみ、次のようにビットマップ比較演算子 **&** が **where** 句でサポートされます。

```
where rs_address_column1  & bitmask
[and rs_address_column2  & bitmask]
 [and other_search_conditions]
```

- 変更されたカラムが *rs_address* カラムだけである場合、変更されたビットがレプリケートデータベースでローを挿入または削除する必要があることを示していない限り、Replication Server はそのローを複写しません。

  このフィルタリングによって、レプリケートデータベース内の *rs_address* カラムは、プライマリデータベースの対応するカラムと同一にはならない場合があります。これにより、*rs_address* カラムを使用して送信先レプリケートデータベースを指定するアプリケーションが最適化されます。

rs_address データ型の動作

- *rs_address* カラムフィールド内の各ビットは、データのカテゴリ (「在庫」や「売り上げ」など) を表すことができます。サブスクリプションビットマスクでは、サブスクリプションを作成するサイトにレプリケートするデータのカテゴリごとに、対応するビットを "on" (1) に設定します。

  たとえば、在庫データに関心のある倉庫サイトのユーザは、サブスクリプションビットマップの在庫ビットを "on" に設定します。同じ倉庫ユーザが売り上げデータには関心がない場合は、そのビットを "off" (0) に設定します。サブスクリプションビットマスクと *rs_address* カラムの両方でビットが "on" に設定されると、そのビットを含むローがレプリケートされます。

基本となる `int` データ型 (**rs_address** 用) の 32 ビット制限

- 基本となる *int* データ型の 32 ビット制限によって、複数の *rs_address* カラムを使用してプライマリテーブルを構成する必要がある場合があります。**and** キーワードを使用すると、複数の *rs_address* カラムに対してビットマップの比較を実行する単一のサブスクリプションを作成できます。
  ただし、複数の *rs_address* カラムのいずれかに 1 つ以上のビットが設定されているときに、1 つのローに対するサブスクリプションを作成する場合は、サブスクリプションを個別に作成する必要があります。

rs_address での 32 ビットの 16 進数の使用

- コマンド構文で記述されているように、*rs_address* ではないカラムに検索条件を指定することもできます。その際、**and** キーワードと比較演算子 (`&` 以外) を使用します。検索条件を指定するのに **and** を使用した場合、*rs_address* ビットマップ比較が他の場合にはローをレプリケートするにもかかわらず、サブスク

リプションデータがレプリケートされなかったり、サブスクリプションがマイグレートアウトしたりすることがあります。

- *rs_address* カラムは、**where** 句内の 32 ビット整数値または 32 ビットの 16 進数と比較できます。16 進数を使用する場合は、必要に応じて各数字に 0 を埋め込み、8 桁の 16 進数値を作成します。

  警告！ *rs_address* カラムを、サブスクリプションの **where** 句内の 16 進数と比較する場合には、細心の注意が必要です。Adaptive Server と Replication Server では、16 進数値はバイナリ文字列として扱われます。バイナリ文字列は、バイトをコピーすることによって整数に変換されます。その結果のビットパターンは、異なるプラットフォームでは、異なる整数値として示されることがあります。

  たとえば、0x0000100 は、バイト 0 を最上位バイトとするプラットフォーム上では 65,536 と見なされ、バイト 0 を最下位バイトとするプラットフォーム上では 256 と見なされます。これらのバイト順の違いにより、16 進数を含むビットマップサブスクリプションは、マルチプラットフォーム複写システムでは動作しないことがあります。

- *rs_address* データ型と *int* データ型の詳細については、「データ型」を参照してください。『Replication Server 管理ガイド 第 1 巻』も参照してください。
- データ型間の変換の詳細については、『Adaptive Server Enterprise リファレンスマニュアル』と『Open Client/Server Common Libraries リファレンスマニュアル』を参照してください。

サブスクリプションのモニタ

- サブスクリプションをマテリアライズするとき、Replication Server はサブスクリプションの作成者のログイン名を使用してプライマリデータサーバにログインし、プライマリテーブルからローを選択します。**check subscription** を使用すると、マテリアライゼーションの進行状況をモニタできます。
- **create subscription** は、データのマテリアライゼーションが完了する前にプロンプトを戻します。レプリケート Replication Server で、**check subscription** が "VALID" をレポートすると、マテリアライゼーションが完了します。

パーミッション

create subscription を実行するには、次のログイン名とパーミッションが必要です。

- **user***username***password***pass* オプションを使用しない場合は、レプリケート Replication Server、プライマリ Replication Server、およびプライマリデータベースと同じログイン名とパスワードを使用する必要があります。

- このコマンドを入力するレプリケート Replication Server では、"create object" または "sa" パーミッションが必要です。
- プライマリ Replication Server では、"create object"、"primary subscribe"、または "sa" パーミッションが必要です。
- プライマリ Adaptive Server データベース内のプライマリテーブルに対しては、**select** パーミッションが必要です。
- プライマリ Araptive Server データベース内の **rs_marker** ストアドプロシージャに対しては、**execute** パーミッションが必要です。
- レプリケートデータベースのメンテナンスユーザには、レプリケートテーブルに対する **select**、**insert**、**update**、および **delete** の各パーミッションと、レプリケーションで使用されるファンクションに対する **execute** パーミッションが必要です。
- プライマリデータベースのメンテナンスユーザを **user***username* **password***pass* オプションで使用してサブスクリプションを作成することはできません。 メンテナンスユーザはデフォルトではレプリケーション時に無視されるので、メンテナンスユーザは使用しないことをおすすめします。
- **create subscription** コマンドで **direct_load** オプションとともに **user***username* **password***pass* を指定した場合は、この名前がプライマリ Adaptive Server データベースまたは Replication Agent へのログイン、およびプライマリテーブルからの選択に使用されます。 **create subscription** コマンドに *username* を指定しない場合は、Replication Server へのログインに使用した名前とパスワードが、プライマリ Adaptive Server データベースまたは Replication Agent への接続に使用されます。 プライマリ Adaptive Server データベースまたは Replication Agent へのログインに使用されるどの名前も、必要なパーミッションを持っていることを確認してください。
- サブスクリプションが **direct_load** オプションと明示的なユーザおよびパスワードを指定して作成された場合、そのユーザ ID とパスワードは 1 回使用され、`rs_users` には格納されません。 このユーザ ID とパスワードは、プライマリ Adaptive Server データベース (または Replication Agent) およびプライマリ Replication Server へのログオンに使用され、ここで説明したパーミッションを持っている必要があります。

参照：

- alter applied function replication definition (129 ページ)
- alter database replication definition (177 ページ)
- alter request function replication definition (210 ページ)
- check subscription (233 ページ)
- create alternate connection (269 ページ)
- create article (280 ページ)

- create database replication definition (300 ページ)
- create applied function replication definition (274 ページ)
- create function string (318 ページ)
- create publication (341 ページ)
- create replication definition (346 ページ)
- create request function replication definition (362 ページ)
- define subscription (395 ページ)
- drop subscription (424 ページ)
- set (444 ページ)
- sysadmin apply_truncate_table (458 ページ)
- 真数値 (整数) データ型 (26 ページ)

create user

Replication Server に新しいユーザログイン名を追加します。

構文

```
create user user
set password {new_password | null}
[set password_parameter to 'parameter_value']
```

パラメータ

- **user –** ログイン名。
- **new_password –** 新しいパスワード。
- **old_password –** *verify password* パラメータを使用する場合、現在のユーザパスワード。
- **password parameter –** 表 36 : パスワードパラメータを参照してください。
- **parameter_value –** パスワードパラメータの記述と値。

表 36 : パスワードパラメータ

password_parameter	説明と*値*
password_expiration	パスワードの有効期限が切れてから経過した日数。 • 0 - パスワードの有効期限は切れません (デフォルト)。 • 範囲 - 0 ～ 32,767。 **password_expiration** を **create user** で使用できます。 パスワードの有効期限が切れると、Replication Server はユーザアカウントをロックし、パスワードの有効期限が切れたことをユーザに通知します。ユーザはこのパスワードをリセットしないと、管理者がパスワードをリセットしないかぎり、いったん接続を解除された以降はログインできなくなります。新しいパスワードは、パスワード要件をすべて満たす必要があります。 **rs_init** が **connect source** パーミッションまたは ID ユーザで作成するユーザのパスワードには、有効期限はありません。そのようなパスワードは、Replication Server でユーザ全員に対して設定された **password_expiration** のどのような設定でもオーバーライドします。データベース、他の Replication Server、および Replication Agent では、**connect source** パーミッションがあるユーザ ID を使用します。 管理者は、レプリケーションエージェントまたは RSI 向けに作成されたユーザのパスワードの有効期限が切れないよう、配慮してください。

例

- **例 1 –** パスワードとして "EnnuI" を設定した新しいユーザログイン名 "louise" を作成します。

```
create user louise
set password EnnuI
```

- **例 2 –** jsmith という名前のユーザを作成します。初期パスワードは 1Buiopr89、パスワード有効期間は 90 日です。

```
create user jsmith
set password to 1Buiopr89
set password_expiration to '90'
```

使用法

- **create user** は、ユーザの新しいログイン名を作成します。
- ユーザは、**alter user** コマンドを使用して各自のパスワードを変更できます。
- ユーザログイン名とパスワードでは、大文字と小文字が区別されます。

- **password_expiration** は、管理者が **alter user** コマンドおよび **create user** コマンドとともに使用できる唯一のパラメータです。
- **create user** コマンドを使用して個別のユーザに対して指定されたパスワード設定は、**configure replication server** コマンドを使用して設定された値をオーバーライドします。
  「**configure replication server**」の「表 24 : パスワードパラメータ」表を参照してください。
- パスワード有効期間:
  - ユーザのパスワードを管理者またはそのユーザ本人が変更すると、Replication Server はパスワードが設定された日付を記録します。ユーザがログインすると、Replication Server はログイン日付とパスワード有効期限設定を比較します。パスワード有効期限設定が、そのユーザに対してまたはシステムレベルで設定されており、かつ Replication Server がパスワードの有効期限が切れたと判断した場合、Replication Server はユーザにパスワードを変更するよう通知し、ユーザアカウントをロックします。Replication Server がアカウントのロックを解除するのは、ユーザがパスワード要件をすべて満たす新しいパスワードを入力した場合だけです。ユーザが接続を解除してからパスワードを変更した場合は、管理者がパスワードをリセットしてください。
  - "connect source" 権限を持つユーザ (Replication Agent ユーザなど) については、**password_expiration** を 0 に設定することをおすすめします。これは、パスワードの有効期限が切れて、データの複写が妨げられることを防ぐためです。

パーミッション

create user には、"sa" パーミッションが必要です。

参照：

- alter user (224 ページ)
- drop user (429 ページ)
- grant (430 ページ)
- revoke (443 ページ)

define subscription

Replication Server システムテーブルにサブスクリプションを追加しますが、サブスクリプションのマテリアライゼーションまたはアクティブ化は行いません。サブスクリプションは、データベース複写定義、テーブル複写定義、ファンクション複写定義、またはパブリケーションに対して作成できます。このコマンドは、バ

ルクサブスクリプションマテリアライゼーションの処理、またはパブリケーションサブスクリプションのリフレッシュ処理を開始します。

構文

```
define subscription sub_name
for {table_rep_def | function_rep_def |
publication pub_name | database replication definition db_repdef
with primary at data_server.database} |
with replicate at data_server.database
[where {column_name | @param_name}
{< | > | >= | <= | = | &} value
[and {column_name | @param_name}
{< | > | >= | <= | = | &} value]...]
[subscribe to truncate table]
[for new articles]
[use dump marker]
```

パラメータ

- **sub_name –** サブスクリプションの名前です。この名前は識別子の規則に従う必要があります。サブスクリプションの名前は、複写定義 (適用される場合) と、レプリケートデータサーバおよびデータベースに対してユニークでなければなりません。
- **for table_rep_def –** サブスクリプションの対象となるテーブル複写定義を指定します。
- **for function_rep_def –** サブスクリプションの対象となるファンクション複写定義の名前を指定します。
- **for publication pub_name –** サブスクリプションの対象となるパブリケーションを指定します。
- **for database replication definition db_repdef –** サブスクリプションの対象となるデータベース複写定義を指定します。
- **with primary at data_server.database –** プライマリデータのロケーションを指定します。プライマリデータベースがウォームスタンバイアプリケーションの一部である場合、*data_server.database* は論理データサーバと論理データベースの名前になります。この句を使用するのは、パブリケーションのサブスクリプションの場合だけです。
- **with replicate at data_server.database –** レプリケートデータのロケーションを指定します。レプリケートデータベースがウォームスタンバイアプリケーションの一部である場合、*data_server.database* は論理データサーバと論理データベースの名前になります。
- **where –** サブスクリプションによって複写されるカラムまたはパラメータの値に対する基準を設定します。**where** 句を省略すると、すべてのローまたはパラメータが複写されます。

where 句は、テーブル複写定義またはファンクション複写定義のサブスクリプションに指定できます。パブリケーションのサブスクリプションには、**where** 句は使用できません。

where 句は、1 つ以上の単純比較で構成されます。単純比較では、次に示す関係演算子のいずれかを使用して、複写定義のサーチャブルカラムまたはサーチャブルパラメータがリテラル値と比較されます。**<**、**>**、**<=**、**>=**、**=**、または **&** (**&** 演算子は、*rs_address* データ型のカラムまたはパラメータでのみサポートされます)。また、キーワード **and** を使用して比較を結合できます。

式で使用されるカラム名またはパラメータ名は、テーブル複写定義の **searchable columns** リストまたはファンクション複写定義の **searchable parameters** リストに含まれている必要があります。

Java カラムはサブスクリプションの式では評価できません。このため、**where** 句には、*rawobject* または *rawobject in row* などのタイプの Java カラムを指定することはできません。

- **column_name –** テーブル複写定義のサブスクリプションに使用するプライマリテーブルのカラム名です。
- **@param_name –** ファンクション複写定義に対するサブスクリプション用の複写ストアドプロシージャからのパラメータ名です。
- **value –** 指定したカラムまたはパラメータの値です。
- **subscribe to truncate table –** テーブル複写定義またはパブリケーションのサブスクリプションに対して、サブスクリプションを作成するレプリケートデータベースへの **truncate table** コマンドの複写を有効にします。

  このオプションは、同じレプリケートテーブルにデータを複写する他の既存のサブスクリプションと同じように設定してください。そうしないと、新しいサブスクリプションは拒否されます。
- **for new articles –** 既存のサブスクリプションをリフレッシュします。サブスクリプションをパブリケーションと照合し、サブスクリプションが作成されていないアーティクルに対してサブスクリプションを作成するように Replication Server に指示します。
- **use dump marker –** レプリケートデータベースにトランザクションを適用するように Replication Server に指示します。**use dump marker** は、データベースサブスクリプションを自動的にアクティブ化および確定化します。このオプションを指定しない場合、ユーザはデータベースサブスクリプションを手動でアクティブ化および確定化する必要があります。

注意： **dump marker** は 1 つずつ使用してください。これは、**dump marker** では複数のデータベースサブスクリプションを定義できないからです。また、各サブスクリプションコマンドの間に **dump database** コマンドを入れる必要があり

ます。MSA 複写でプラットフォーム間の dump と load (XPDL) 機能を使用する場合、マテリアライズに **use dump marker** 句を使用しないでください。

例

- **例 1 –** *titles_sub* という名前のサブスクリプションを作成します。このサブスクリプションは、タイプが "business" であるカラムを持つ *titles* テーブルのローを、SYDNEY_DS というデータサーバの *pubs2* データベースにある *titles* テーブルに複写することを指定しています。

```
define subscription titles_sub
for titles_rep
with replicate at SYDNEY_DS.pubs2
where type = 'business'
```

- **例 2 –** 10.00 ドル以上の価格が指定された *titles* テーブルのローを含む *titles_sub* というサブスクリプションを作成します。

```
define subscription titles_sub
for titles_rep
with replicate at SYDNEY_DS.pubs2
where price >= $10.00
```

- **例 3 –** ファンクション複写定義 *myproc_rep* の *myproc_sub* というサブスクリプションを作成します。

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

- **例 4 –** パブリケーション *pubs2_pub* の *pubs2_sub* というサブスクリプションを作成します。

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

- **例 5 –** データベース複写定義 *pubs2_rep* のサブスクリプション *pubs2_sub* を作成します。

```
define subscription pubs2_sub
for database replication definition pubs2_rep
with primary at NEWYORK_DS.pubs2
with replicate at TOKYO_DS.pubs2
subscribe to truncate table
use dump marker
```

完全な複写システムに対するサブスクリプションの作成例については、『Replication Server デザインガイド』を参照してください。

使用法

- **define subscription** は、バルクマテリアライゼーションを使用して手動でサブスクリプションを作成するために使用します。バルクマテリアライゼーションを使用すると、サブスクリプションの作成とマテリアライゼーションが個別のステップで実行されるため、最初のデータは、プライマリデータベースから WAN を介して送信するのではなく、メディアからロードすることができます。
- 既存のサブスクリプションを持つパブリケーションに新しいアーティクルを追加した場合は、新しいアーティクルのサブスクリプションを作成するために、パブリケーションサブスクリプションをリフレッシュしてください。
- **activate subscription** はサブスクリプションをアクティブ化する場合に使用し、**validate subscription** はサブスクリプションを確定化する場合に使用します。
- 同じプライマリテーブルに対して、複数の複写定義を作成することはできますが、同じレプリケートテーブルの複数の複写定義に対して、サブスクリプションを作成することはできません。ただし、同じ複写定義に対して、サブスクリプション作成を複数回実行できます。

パブリケーションのサブスクリプションの作成

- パブリケーションが有効であれば、そのパブリケーションに対してサブスクリプションを作成し、レプリケートデーターベースへの複写を開始できます。すべての形式のサブスクリプションマテリアライゼーションがサポートされています。
- パブリケーションサブスクリプションに新しいアーティクルサブスクリプションを作成するには、**define subscription** を使用します。次に、**activate subscription** を使用して新しいアーティクルサブスクリプションのサブスクリプションデータを手動でロードし、**validate subscription** を使用して、パブリケーションサブスクリプションを確定化します。
- パブリケーションサブスクリプションを作成すると、Replication Server は、パブリケーションに組み込まれたアーティクルごとに、別々の基本サブスクリプションを作成します。各アーティクルサブスクリプションは、親パブリケーションサブスクリプションの名前を使用します。
- パブリケーションサブスクリプションをアクティブ化して確定化すると、そのすべてのアーティクルサブスクリプションも同時にアクティブ化および確定化されます。
- パブリケーションのサブスクリプションには、**where** 句を指定することはできません。その代わりに、パブリケーションに組み込まれた各アーティクル内に 1 つまたは複数の **where** 句を指定することによって、レプリケートサイトへのレプリケーションをカスタマイズできます。

データベース複写定義のサブスクリプションの作成

- データベースサブスクリプションを作成する場合、**where** 句を指定してデータのサブスクリプションを制限することはできません。すべてのデータがサブスクリプションの対象となります。
- データベースサブスクリプションでは、非マテリアライゼーションメソッドまたはバルクマテリアライゼーションメソッドだけを使用できます。ダンプとロードを使用したり、その他のバルクマテリアライゼーションメソッドを使用するには、**define subscription** コマンドを使用します。非マテリアライゼーションメソッドを使用するには、**create subscription** コマンドを使用します。
- 同じオリジンから複数のデータベース複写定義に対してサブスクリプションを作成することはできません。

truncate table の複写

- テーブルに対して最初のサブスクリプションを作成する場合、**subscribe to truncate table** オプションは指定しても指定しなくてもかまいません。同じテーブル内に情報をコピーする後続の各サブスクリプションは、最初のサブスクリプションの例に従う必要があります。最初のサブスクリプションに従わないと、サブスクリプションを作成しようとしたときに拒否されます。
- **sysadmin apply_truncate_status** を実行すると、特定のレプリケートテーブルの現在の "subscribe to truncate table" ステータスを表示または変更できます。

rs_address データ型の使用

「**create subscription**」を参照して、*rs_address* データ型を使用するカラムまたはパラメータの機能の詳細を確認してください。

define subscription を実行するための必要条件

このコマンドを実行する前に、後述するパーミッションの他に、次の条件を満たしているかどうかを確認してください。

- テーブル複写定義に対するサブスクリプションの場合
  - 複写するプライマリテーブルに対して複写定義が存在し、そのテーブルが **sp_setreptable** を使用して複写用にマーク付けされている。
  - 複写定義内で参照されるテーブルが、プライマリデータベースとレプリケートデータベースの両方に存在する。また、各テーブルには、複写定義内で指定されたカラムとデータ型が存在する。
    このテーブルは、サブスクリプションを作成するユーザと管理するユーザから参照可能である必要があります。これを実現する最も簡単な方法は、データベース所有者にテーブルを作成させることです。

  ファンクション複写定義に対するサブスクリプションの場合
  - 複写するストアドプロシージャに対して複写定義が存在し、そのストアドプロシージャが **sp_setrepproc** を使用して複写用にマーク付けされている。
  - ファンクション複写定義内で参照されるストアドプロシージャが、プライマリデータベースとレプリケートデータベースの両方に存在する。また、

各テーブルには、ファンクション複写定義内で定義されたパラメータとデータ型が存在する。

パブリケーションに対するサブスクリプションの場合

- 複写するプライマリテーブルまたはストアドプロシージャのアーティクルを含むパブリケーションが存在する。このアーティクルは、上記の条件を満たす複写定義を指定する。
- パブリケーションが有効である。

define subscription を使用したサブスクリプションの作成

- **define subscription** を使用すると、テーブル複写定義、ファンクション複写定義、またはパブリケーションのサブスクリプションを作成できます。
  - テーブル複写定義へのサブスクリプションの場合は、レプリケートデータが格納されるデータベースを管理する Replication Server で、 **define subscription** を入力します。
  - ファンクション複写定義へのサブスクリプションの場合は、適用ファンクションの配信を介して送信先ストアドプロシージャが実行されるデータベースを管理する Replication Server で、**define subscription** を入力します。
  - パブリケーションへのサブスクリプションの場合は、レプリケートデータが格納されるデータベース、または送信先ストアドプロシージャが実行されるデータベースを管理する Replication Server で、**define subscription** を入力します。
- テーブルサブスクリプションは、データベース内で、テーブルまたはテーブルから選択したローのレプリケートコピーを保持します。プライマリバージョンに加えられた変更は、このコピーにも適用されます。
- ファンクションサブスクリプションは、ファンクション複写定義に対応したユーザ定義ファンクションの呼び出しを複写します。複写ファンクションは、通常、パラメータを指定してデータを修正しますが、複写データを指定する必要はありません。
- パブリケーションサブスクリプションには、パブリケーションに含まれるアーティクルの基本サブスクリプションが含まれます。基本サブスクリプションは、アーティクル内の複写定義に従って、テーブルまたはユーザ定義ファンクションの呼び出しを複写します。
- サブスクリプションの詳細とレプリケーションにおけるサブスクリプションの役割については、『Replication Server 管理ガイド 第 1 巻』を参照してください。

サブスクリプションを作成するための代替コマンド

- テーブル複写定義、ファンクション複写定義、またはパブリケーションのサブスクリプションの作成、マテリアライズ、アクティブ化、確定化を 1 つの手順で行うには、**create subscription** を使用します。

パーミッション

define subscription を実行するには、次のログイン名とパーミッションが必要です。

- レプリケート Replication Server、プライマリ Replication Server、プライマリデータベースで、同じログイン名とパスワードが必要です。
- このコマンドを入力するレプリケート Replication Server では、"create object" または "sa" パーミッションが必要です。
- プライマリ Replication Server では、"create object"、"primary subscribe"、または "sa" パーミッションが必要です。

参照：

- alter applied function replication definition (129 ページ)
- alter request function replication definition (210 ページ)
- activate subscription (48 ページ)
- check subscription (233 ページ)
- create article (280 ページ)
- create function replication definition (312 ページ)
- create publication (341 ページ)
- create applied function replication definition (274 ページ)
- create request function replication definition (362 ページ)
- create subscription (376 ページ)
- drop subscription (424 ページ)
- sysadmin apply_truncate_table (458 ページ)
- validate subscription (525 ページ)

disconnect

サーバへのコネクションを終了します。

構文

```
{disconnect | disc} [all]
```

例

- **例 1 –** ost_replinuxvm_02 から ost_replinuxvm_03 へのコネクションを作成します。これにより、ost_replinuxvm_02 は ost_replinuxvm_03 から切断されます。

```
isql -Usa -P -S ost_replinuxvm_02
1> connect to ost_replinuxvm_03
2> go
```

```
Gateway connection to 'ost_replinuxvm_03' is created.
```

```
1> disc
2> go
```

```
Gateway connection to 'ost_replinuxvm_03' is dropped.
```

使用法

- **disconnect** により、コネクションスタックは一度に 1 つずつ終了します。すべてのコネクションを終了するには、**disconnect all** を使用します。
- Replication Server 15.1 以前では、**disconnect** コマンドの動作が異なります。これらのバージョンでは、**disconnect** コマンドは、ゲートウェイモードを終了し、最初の **connect** コマンドを発行した Replication Server に稼働中のサーバのステータスを返します。コネクションスタックに Replication Server バージョン 15.2 と 15.1 以前が含まれる場合に **disconnect** コマンドを発行すると、**show connection** コマンドや **show server** コマンドを実行したときに、想定した出力が表示されない可能性があります。

パーミッション

このコマンドは、すべてのユーザが実行できます。

参照：

- connect (266 ページ)
- show connection (449 ページ)
- show server (450 ページ)

drop article

アーティクルを削除し、必要に応じてその複写定義を削除します。

構文

```
drop article article_name
for pub_name
with primary at data_server.database
[drop_repdef]
```

パラメータ

- **article_name –** 削除するアーティクルの名前です。
- **for pub_name –** アーティクルの対象となるパブリケーションの名前を指定します。
- **with primary at data_server.database –** プライマリデータのロケーションを指定します。プライマリデータベースがウォームスタンバイアプリケーションの一部である場合、*data_server.database* は論理データサーバと論理データベースの名前になります。
- **drop_repdef –** オプションのキーワードであり、このキーワードを指定すると、アーティクルの複写定義が他の場所で使用されていなければ削除されます。

例

- **例 1 –** TOKYO_DS.*pubs2* データベースのパブリケーション *pubs2_pub* の *titles_art* というアーティクルを削除します。

```
drop article titles_art
 for pubs2_pub
 with primary at TOKYO_DS.pubs2
```

- **例 2 –** TOKYO_DS.*pubs2* データベースのパブリケーション *pubs2_pub* の *titles_art* というアーティクルを削除します。このコマンドは、アーティクルの複写定義が他の場所で使用されていなければ、その複写定義も削除します。

```
drop article titles_art
 for pubs2_pub
 with primary at TOKYO_DS.pubs2
 drop_repdef
```

使用法

- パブリケーションからアーティクルを削除するには、**drop article** を使用します。**drop article** は、プライマリデータが格納されているデータベースを管理する Replication Server で実行します。
- アーティクルについてのサブスクリプションがない場合は、そのアーティクルを削除できます。必要であれば、まずサブスクリプションを削除してください。
- アーティクルが他のアーティクルの一部ではなくサブスクリプションもない場合は、オプションでそのアーティクルの複写定義も削除できます。
- 削除されたアーティクルは、**create/define subscription** が実行された場合にのみ、レプリケートサイトから削除されます。

サブスクリプションがあるパブリケーションからのアーティクルの削除

- 既存のパブリケーションからアーティクルを削除した場合、パブリケーションは不確定化されます。**drop subscription for article** コマンドを使用して既存のす

べてのアーティクルサブスクリプションを削除してからでないと、アーティクルは削除できません。新しいパブリケーションサブスクリプションを作成するには、次のようにしてください。

- パブリケーションへの変更が終了したら、そのパブリケーションを確定化する。

「**create subscription**」と「**define subscription**」を参照して、パブリケーションサブスクリプションをリフレッシュする 2 つの方法の詳細を確認してください。

パーミッション

drop article には、"create object" パーミッションが必要です。

参照：

- check subscription (233 ページ)
- create article (280 ページ)
- create publication (341 ページ)
- create subscription (376 ページ)
- define subscription (395 ページ)
- drop function replication definition (411 ページ)
- drop publication (418 ページ)
- drop replication definition (419 ページ)
- drop subscription (424 ページ)

drop auto partition path

自動でサイズ変更可能なパーティションを Replication Server から削除します。

構文

```
drop auto partition path logical_name
```

パラメータ

- **logical_name** – 既存の自動でサイズ変更可能な Replication Server パーティションに対する論理パーティションパス名。 名前は識別子の規則に従う必要があります。 *logical_name* は、**create auto partition path** コマンドおよび **alter auto partition path** コマンドでも、自動でサイズ変更可能なパーティションの指定に使用します。

例

- **例 1 –** `auto_uxp` 論理パーティションパスに割り当てられた、自動でサイズ変更可能なパーティションと、`auto_uxp` に関連して自動生成されたすべてのパーティションファイルを Replication Server から削除します。

```
drop auto  partition path auto_uxp
```

使用法

- ディスク容量を至急確保する必要がある場合、**drop partition** を使用することで、自動で作成されたパーティションファイルを手動で削除できます。 または、**drop auto partition path** を使用して、自動でサイズ変更可能なパーティションの削除を行い、ディスク領域を適切に解放します。
- Replication Server で **drop auto partition path** を実行しても、自動でサイズ変更可能なパーティションがすぐに削除されるわけではありません。 このコマンドは、削除対象となる論理パーティションパスを "削除保留中" としてマークします。自動でサイズ変更可能な新規パーティションファイルは、このパスには作成されません。 Replication Server は、パス上のパーティションファイルが Replication Server によって削除されてから、自動でサイズ変更可能なパーティションを削除します。
- Replication Server のインストール時に初期パーティションが作成されます。 初期パーティションおよびそれ以降に手動で作成されるパーティションは、最小パーティションサイズ単位で構成され、自動でサイズ変更可能なパーティションをすべて削除した後でも、それは維持されます。 最小パーティションサイズのパーティションのいくつかは削除可能ですが、複写を続けるために十分なサイズのパーティションをいくつか残す必要があります。
- 障害が発生したパーティションのリカバリの詳細については、『Replication Server 管理ガイド第 2 巻』の「複写システムリカバリ」の「パーティションのロスまたは障害からのリカバリ」を参照してください。

パーミッション

drop auto partition path を使用するには、ディスクパーティションまたはオペレーティングシステムファイルは、“sybase” ユーザが所有するようにします。このユーザには、パーティションに対する読み込み/書き込みパーミッションが必要です。 “sybase” 以外のユーザには、このパーティションに対する読み込み/書き込みパーミッションを付与しないでください。

参照：

- admin auto_part_path (53 ページ)
- alter auto partition path (133 ページ)
- create auto partition path (284 ページ)

- rs_helppartition (714 ページ)
- drop partition (417 ページ)

drop connection

複写システムからデータベースを削除します。

構文

```
drop connection to data_server.database
```

パラメータ

- **data_server –** 複写システムから削除するデータベースのあるデータサーバの名前です。
- **database –** コネクションを削除するデータベースの名前です。

例

- **例 1 –** SYDNEY_DS データサーバにある *pubs2* データベースへのコネクションを削除します。

  ```
  drop connection to SYDNEY_DS.pubs2
  ```

使用法

- **drop connection** は、デフォルトコネクションと代替コネクションの Replication Server システムテーブルからデータベース接続情報を削除するときに使用します。 このコマンドは、システム内のどのデータベースからも複写データは削除しません。
- コネクションを削除する前に、次のことを実行してください。
  - データベースへデータを複写するサブスクリプションをすべて削除する。
  - プライマリデータベースに対するコネクションの場合は、データベースのテーブルに対する複写定義をすべて削除する。
- 同じ名前でデータベースへのコネクションを再作成する場合は、**sysadmin dropdb** を使用して、あらかじめコネクションを削除しておく必要があります。
- Replication Server は、削除されたデータベースコネクションに関する情報を、条件を満たしているサイトへ複写システムを介して分配します。複写システムで通常の遅延時間が発生するため、変更内容がレプリケートサイトにすぐに反映されるわけではありません。

パーミッション

drop connection には、"sa" パーミッションが必要です。

参照：

- admin show_connections (77 ページ)
- alter connection (134 ページ)
- create alternate connection (269 ページ)
- create connection (287 ページ)
- resume connection (436 ページ)
- suspend connection (452 ページ)
- sysadmin dropdb (466 ページ)

drop database replication definition

既存のデータベース複写定義を削除します。

構文

```
drop database replication definition db_repdef
   with primary at server_name.db
```

パラメータ

- **db_repdef –** データベース複写定義の名前です。
- **server_name.db –** プライマリサーバとデータベースの組み合わせの名前です。例：*TOKYO.dbase*.

例

- **例 1 –** データベース複写定義 *dbrep1* を削除します。

```
drop database replication definition dbrep1
  with primary at PDS.my_db
```

使用法

drop database replication definition は、指定したデータベース複写定義のデータベースサブスクリプションがない場合にのみ成功します。

参照：

- alter database replication definition (177 ページ)
- create database replication definition (300 ページ)

drop error class

エラークラスとそのクラスに対応するすべてのアクションを削除します。

構文

```
drop [replication server] error class error_class
```

パラメータ

- **replication server –** エラークラスが Replication Server エラークラスであり、データサーバのエラークラスではないことを示します。
- **error_class –** 削除するエラークラスの名前です。

例

- **例 1 –** Replication Server から *pubs2_db_err_class* エラークラスを削除します。また、*pubs2_db_err_class* エラークラスに割り当てられているエラーアクションも削除します。

  ```
  drop error class pubs2_db_err_class
  ```

- **例 2 –** Replication Server から *sydney_rs_err_class* Replication Server エラークラスを削除します。また、*sydney_rs_err_class* エラークラスに割り当てられているエラーアクションも削除します。

  ```
  drop replication server error class sydney_rs_err_class
  ```

使用法

- **drop error class** コマンドは、エラークラスを削除するときに使用します。エラークラスを削除すると、そのクラスに割り当てられているアクションもすべて削除されます。
- **drop error class** は、エラークラスを作成した Replication Server で実行してください。
- 次のものは削除できません。
  - *rs_sqlserver_error_class* エラークラス
  - *rs_repserver_error_class* エラークラス
  - データベースで使用中のエラークラス
- エラークラスのプライマリサイトを変更するには、**move primary of error class** コマンドを使用します。
- Replication Server は、削除されたクラスに関する情報を、条件を満たしているサイトへ複写システムを介して分配します。複写システムで通常の遅延時間が

発生するため、変更内容がレプリケートサイトにすぐに反映されるわけではありません。

パーミッション

drop error class には、"sa" パーミッションが必要です。

参照：
- assign action (226 ページ)
- alter error class (182 ページ)
- create connection (287 ページ)
- create error class (308 ページ)
- drop connection (407 ページ)
- move primary (432 ページ)

drop function

ユーザ定義ファンクションとそのファンクション文字列を削除します。

構文

```
drop function [replication_definition.]function
```

パラメータ

- **replication_definition** – ファンクションが作成されている複写定義の名前です。
- **function** – 削除するファンクションの名前です。

例

- **例 1** – *publishers_rep* 複写定義の *upd_publishers* ユーザ定義ファンクションを削除します。また、ファンクションに定義されているファンクション文字列もすべて削除します。

  ```
  drop function publishers_rep.upd_publishers
  ```

使用法

- **drop function** は、ファンクション名と、そのファンクションに対して作成されているすべてのファンクション文字列を削除するときに使用します。
- **drop function** は、複写定義を作成した Replication Server で実行してください。

- システムファンクションは削除できません。システムファンクションの詳細については、「Replication Server システムファンクション」を参照してください。
- Replication Server は、削除されたユーザ定義ファンクションに関する情報を、条件を満たしているサイトへ複写システムを介して分配します。複写システムで通常の遅延時間が発生するため、変更内容がレプリケートサイトにすぐに反映されるわけではありません。
- 複写定義の 1 つのユーザ定義ファンクションを削除すると、プライマリテーブル内のすべての複写定義からこのユーザ定義ファンクションが削除されます。
- 複写ファンクションに対して **drop function** を実行しないでください。代わりに **drop function replication definition** を使用します。

パーミッション

drop function には、"create object" パーミッションが必要です。

参照：
- create function (310 ページ)
- drop function string (412 ページ)
- move primary (432 ページ)

drop function replication definition

ファンクション複写定義とそのユーザ定義ファンクションを削除します。

構文

```
drop function replication definition function_rep_def
```

パラメータ

- **function_rep_def –** 削除するファンクション複写定義の名前です。

例

- **例 1 –** *titles_frep* というファンクション複写定義と、そのユーザ定義ファンクションおよびファンクション文字列を削除します。

  ```
  drop function replication definition titles_frep
  ```

使用法

- **drop function replication definition** は、ファンクション複写定義を削除するときに使用します。

- ファンクション複写定義を削除する前に、ファンクション複写定義のサブスクリプションをすべて削除しなければなりません。
- **drop function replication definition** は、ファンクション複写定義のプライマリ Replication Server で実行してください。
- このファンクション複写定義によって定義されたストアドプロシージャを削除したら、データベースで **sp_setrepproc** を実行して、プロシージャの複写ステータスを **'false'** に設定します。これにより、RepAgent はログエントリを Replication Server に転送しなくなります。
- Replication Server は、複写システムを介して、削除されたファンクション複写定義の情報を条件を満たすサイトに分配します。複写システムで通常の遅延時間が発生するため、変更内容がレプリケートサイトにすぐに反映されるわけではありません。

パーミッション

drop function replication definition には、"create object" パーミッションが必要です。

参照：

- alter applied function replication definition (129 ページ)
- alter request function replication definition (210 ページ)
- check subscription (233 ページ)
- create applied function replication definition (274 ページ)
- create request function replication definition (362 ページ)
- create subscription (376 ページ)
- define subscription (395 ページ)
- drop subscription (424 ページ)

drop function string

ファンクション文字列クラスからファンクション文字列を削除します。

構文

```
drop function string
 {replication_definition |
 [owner.] table |
 stored_procedure} .function[;function_string]
 for { [function class] function_class |
[database] data_server.database}
```

パラメータ

- **replication_definition –** ファンクションが実行されるテーブルまたはファンクション複写定義の名前です。
- **[owner.]table –** ファンクション文字列のテーブル所有者とターゲットテーブルを指定します。
- **stored_procedure –** ファンクション文字列のターゲットストアドプロシージャを指定します。
- **function –** ファンクション文字列が作成されているファンクションの名前です。
- **function_string –** 削除するファンクション文字列の名前です。デフォルトのファンクション文字列の名前は、ファンクションの名前と同じです。
- **function_class –** ファンクション文字列を削除するファンクション文字列クラスの名前です。
- **data_server.database –** ターゲットスコープファンクション文字列を削除するスタンバイデータベースまたはレプリケートデータベースを指定します。

例

- **例 1 – rs_insert** ファンクション (*publishers_rep* 複写定義用) に対するファンクション文字列を削除します。この複写定義は、派生クラス *sqlserver_derived_class* にあります。これで、**rs_insert** ファンクション文字列は、親クラスから継承されることになります。

```
drop function string
 publishers_rep.rs_insert
 for sqlserver_derived_class
```

- **例 2 – upd_publishers** ユーザ定義ファンクション (*publishers_rep* 複写定義用) のファンクション文字列を削除します。この複写定義は、*sqlserver2_function_class* ファンクション文字列クラスにあります。

```
drop function string
 publishers_rep.upd_publishers
 for sqlserver2_function_class
```

- **例 3 –** `dbo.authors` テーブルのカスタムファンクション文字列を削除します。このテーブルは、ターゲットデータベース `NY_DS.rdb1` にあります。

```
drop function string dbo.authors.rs_insert
for database NY_DS.rdb1
```

使用法

- 既存のファンクション文字列を新しいファンクション文字列に置き換えるには、**alter function string** を使用するか、**create function** を **overwrite** を指定して使用します。

警告！ ファンクション文字列を削除または再作成している間にトランザクションを実行すると、ファンクション文字列が失われたことを Replication Server が検出し、トランザクションは失敗します。

- ファンクションを削除すると、対応するファンクション文字列がすべてのファンクション文字列クラスから削除されます。
- 派生ファンクション文字列クラスからカスタマイズされたファンクション文字列を削除すると、そのクラスは親クラスからファンクション文字列を継承します。
- カスタマイズされたファンクション文字列を *rs_sqlserver_function_class* から削除すると、Replication Server は、カスタマイズされたファンクション文字列とデフォルトのファンクション文字列を削除します。カスタマイズされたファンクション文字列を、*rs_sqlserver_function_class* 内のファンクション用のデフォルトの文字列に復元するには、**alter function string** を使用し、**output** 句は省きます。
- Replication Server は、削除されたファンクション文字列に関する情報を、条件を満たしているサイトへ複写システムを介して分配します。複写システムで通常の遅延時間が発生するため、変更内容がレプリケートサイトにすぐに反映されるわけではありません。
- ターゲットデータベース (スタンバイデータベースまたはレプリケートデータベース) を制御する Replication Server にあるターゲットスコープファンクション文字列に対し、**drop function string** を実行します。

パーミッション

drop function string には、"create object" パーミッションが必要です。

参照：

- alter function string (188 ページ)
- create function (310 ページ)
- create function string (318 ページ)
- create function string class (334 ページ)
- drop function (410 ページ)

drop function string class

ファンクション文字列クラスを削除します。

構文

```
drop function string class function_class
```

パラメータ

- **function_class –** 削除するファンクション文字列クラスの名前です。

例

- **例 1 –** 派生ファンクション文字列クラス *sqlserver_derived_class* と、そのカスタマイズされたファンクション文字列をすべて削除します。

```
drop function string class
 sqlserver_derived_class
```

- **例 2 –** ファンクション文字列クラス *sqlserver2_function_class* と、そのファンクション文字列を削除します。

```
drop function string class
 sqlserver2_function_class
```

使用法

- **drop function string class** は、ファンクション文字列クラスを削除するときに使用します。ファンクション文字列クラスは、データベースのすべてのファンクション文字列をグループ化します。
- ファンクション文字列クラスを削除すると、対応するファンクション文字列もすべて削除され、クラスへのすべての参照が削除されます。
- データベースコネクションで使用中のファンクション文字列クラスは削除できません。
- *rs_sqlserver_function_class*、*rs_default_function_class*、*rs_db2_function_class* の 3 つのシステム提供クラスはいずれも削除できません。
- 派生したクラスの親クラスであるファンクション文字列クラスは削除できません。

パーミッション

drop function string class には、"sa" パーミッションが必要です。

参照：

- create function string class (334 ページ)
- drop function (410 ページ)
- drop function string (412 ページ)

drop logical connection

論理コネクションを削除します。論理コネクションは、ウォームスタンバイアプリケーションを管理するために使用します。

構文

```
drop logical connection to data_server.database
```

パラメータ

- **data_server – create logical connection** コマンドで指定された論理データサーバです。
- **database – create logical connection** コマンドで指定されたデータベースの名前です。

例

- **例 1** – LDS というデータサーバと *pubs2* というデータベースの論理コネクションを削除します。

  ```
  drop logical connection to LDS.pubs2
  ```

使用法

- ウォームスタンバイアプリケーションを削除する場合、論理コネクションを削除するには、このコマンドを使用します。
- 論理コネクションを削除する前に、スタンバイデータベースへのコネクションを削除しなければなりません。

パーミッション

drop logical connection には、"sa" パーミッションが必要です。

参照：

- create connection (287 ページ)
- create logical connection (338 ページ)
- drop connection (407 ページ)
- switch active (456 ページ)

drop partition

Replication Server からディスクパーティションを削除します。

構文

```
drop partition logical_name
```

パラメータ

- **logical_name – create partition** を使用して作成されたパーティションに割り当てられた名前です。

例

- **例 1 –** Replication Server から *P1* というパーティションを削除します。

  ```
  drop partition P1
  ```

使用法

- **drop partition** は、ディスクパーティションを削除するときに使用します。このコマンドは、まずパーティションに "pending drop" とマーク付けします。マーク付けされると、パーティションには新しいデータは書き込まれません。パーティションに格納されたすべてのデータが正常に配信されたら、パーティションは削除されます。

  注意： パーティションに格納されているデータの一部がまだ削除できる状態でない場合に **drop partition** を使用すると、予期しない結果になることがあります。たとえば、部分的に埋められたセグメントがパーティションキューに含まれる場合、セグメントが完全に埋められるまでそのキューは削除できません。しかし、パーティションは "pending drop" と指定されているため、セグメントを完全に埋めることができず、コマンドはパーティションの削除に失敗します。
- 障害が発生したパーティションのリカバリの詳細については、『Replication Server 管理ガイド 第 2 巻』を参照してください。

パーミッション

drop partition には、"sa" パーミッションが必要です。

参照：

- admin disk_space (57 ページ)

- alter partition (196 ページ)
- create partition (339 ページ)

drop publication

パブリケーションとそのすべてのアーティクルを削除し、必要に応じてそのアーティクルの複写定義を削除します。

構文

```
drop publication pub_name
with primary at data_server.database
[drop_repdef]
```

パラメータ

- **pub_name –** 削除するパブリケーションの名前です。
- **with primary at data_server.database –** プライマリデータのロケーションを指定します。プライマリデータベースがウォームスタンバイアプリケーションの一部である場合、*data_server.database* は論理データサーバと論理データベースの名前になります。
- **drop_repdef –** オプションのキーワードです。このキーワードを指定すると、パブリケーションのアーティクルの複写定義が削除されます (その複写定義が他の場所で使用されていない場合)。

例

- **例 1 –** プライマリデータベース TOKYO_DS.*pubs2* の *pubs2_pub* というパブリケーションを削除します。

  ```
  drop publication pubs2_pub
   with primary at TOKYO_DS.pubs2
  ```

- **例 2 –** プライマリデータベース TOKYO_DS.*pubs2* の *pubs2_pub* というパブリケーションを削除します。このコマンドは、パブリケーションのアーティクルの複写定義が他の場所で使用されていなければ、それらの複写定義もすべて削除します。

  ```
  drop publication pubs2_pub
   with primary at TOKYO_DS.pubs2
   drop_repdef
  ```

使用法

- **drop publication** は、パブリケーションを削除するときに使用します。**drop publication** は、プライマリデータが格納されているデータベースを管理する Replication Server で実行します。
- パブリケーションのサブスクリプションがない場合は、そのパブリケーションを削除できます。必要であれば、まずサブスクリプションを削除してください。
- パブリケーションを削除すると、そのアーティクルも削除されます。パブリケーションのアーティクルが他のアーティクルの一部ではなくサブスクリプションもない場合は、オプションでそのアーティクルの複写定義もすべて削除できます。
- レプリケートサイトでパブリケーションに対して **define/create subscription** または **check publication** を実行すると、削除されたパブリケーションがそのレプリケートサイトから削除されます。

パーミッション

drop publication には、"create object" パーミッションが必要です。

参照：
- check publication (231 ページ)
- create publication (341 ページ)
- drop article (403 ページ)
- drop function replication definition (411 ページ)
- drop replication definition (419 ページ)
- drop subscription (424 ページ)

drop replication definition

複写定義とそのファンクションを削除します。

構文

```
drop replication definition replication_definition
```

パラメータ

- **replication_definition –** 削除する複写定義の名前です。

例

- **例 1 –** *publishers_rep* という複写定義と、この複写定義のすべてのファンクション文字列を削除します。

```
drop replication definition publishers_rep
```

使用法

- **drop replication definition** は、複写定義を削除するときに使用します。複写定義を削除する前に、その複写定義に対するすべてのサブスクリプションを削除しておかなければなりません。
- **drop replication definition** は、複写定義のプライマリ Replication Server で実行してください。
- 削除された複写定義が、Adaptive Server に格納されているプライマリテーブルに対する最後の複写定義である場合は、複写定義を削除した後、データベースで **sp_setreptable** を実行してください。 テーブルの複写ステータスを false に設定し、そのテーブルに対して Adaptive Server が特別な複写レコードのログを出力しないようにします。
- 複数のバージョンの Replication Server (Replication Server バージョン 11.5 とバージョン 11.0.*x* など) を使用しているときに、同じプライマリテーブルの複数の複写定義を作成した場合、最初に作成した複写定義がマーク付けされ、11.0.*x* 以前の Replication Server に送信されます。この複写定義は、プライマリテーブル名とレプリケートテーブル名が同じで、プライマリカラム名とレプリケートカラム名も同じであり、テーブル所有者名は含まれていません。
  バージョン 11.0.*x* 以前の Replication Server に送信された複写定義が削除されると、11.0.*x* と互換性のある最も古い複写定義 (存在する場合) が 11.0.*x* 以前のサイトに送信されます。混合バージョン環境での複写定義の機能の詳細については、「**create replication definition**」を参照してください。
- Replication Server は、削除された複写定義に関する情報を、条件を満たしているサイトへ複写システムを介して分配します。複写システムで通常の遅延時間が発生するため、変更はすぐにはレプリケートサイトに反映されません。

パーミッション

drop replication definition には、"create object" パーミッションが必要です。

参照：

- alter replication definition (199 ページ)
- check subscription (233 ページ)
- create replication definition (346 ページ)
- create subscription (376 ページ)
- define subscription (395 ページ)

- drop article (403 ページ)
- drop publication (418 ページ)
- drop subscription (424 ページ)
- rs_send_repserver_cmd (735 ページ)

drop route

別の Replication Server へのルートをクローズします。

構文

```
drop route to dest_replication_server
 [with primary at dataserver.database]
 [with nowait]
```

パラメータ

- **dest_replication_server –** ルートを削除する Replication Server の名前です。
- **with primary –** 専用ルートを削除するプライマリデータベースからのコネクションを指定します。
- **with nowait –** 送信先 Replication Server と通信できない場合でもルートをクローズするように、Replication Server に指示します。**with nowait** は、最後の手段として使用してください。この句は、サブスクリプションを持つルート、または間接ルートによって使用されているルートを削除するよう、Replication Server に強制します。通常は、影響を受ける Replication Server の RSSD から無効なリファレンスを削除するための手順が必要になります。

例

- **例 1 –** コマンドを入力したサイトから SYDNEY_RS Replication Server へのルートを削除します。

  ```
  drop route to SYDNEY_RS
  ```

- **例 2 –** `NY_DS.pdb1` プライマリコネクションのために、RS_NY プライマリ Replication Server と RS_LON レプリケート Replication Server との間の専用ルートを削除するには、RS_NY で次のように入力します。

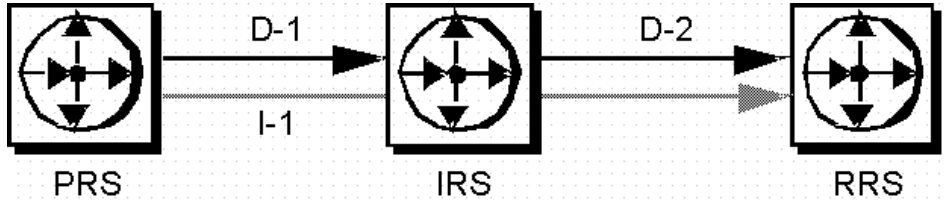
  ```
  drop route to RS_LON
   with primary at NY_DS.pdb1
   go
  ```

使用法

- **drop route** は、このコマンドが入力された Replication Server から指定した Replication Server へのルートをクローズします。
- 専用ルートを削除してから、共有ルートを削除してください。
  専用ルートが削除されると、指定プライマリコネクションから送信先 Replication Server へのトランザクションは、共有ルートを通るようになります。
  『Replication Server 管理ガイド 第 2 巻』>「パフォーマンスチューニング」>「Multi-Path Replication」の「専用ルート」を参照してください。
- ルートを削除する前に、次のことを行っておく必要があります。
  - 送信先 Replication Server で、送信元 Replication Server が管理しているデータベースのプライマリデータに対するすべてのサブスクリプションを削除する。
  - そのルートを使用するすべての間接ルートを削除する。

  たとえば、この図では、ルート I-1 は中間 Replication Server (IRS) を介したプライマリ Replication Server (PRS) からレプリケート Replication Server (RRS) への間接ルートを示します。この間接ルートでは、直接ルート D-1 と D-2 を使用しています。

図 4 : 直接ルートと間接ルートの例



直接ルート D-2 を削除するには、レプリケート Replication Server で、プライマリ Replication Server または中間 Replication Server の複写定義のサブスクリプションをすべて削除してから、間接ルート I-1 を削除します。

警告！ with nowait 句は、最後の手段としてのみ使用してください。**with nowait** 句は、送信先 Replication Server を今後使用する予定がない場合、または送信先 Replication Server がダウンしているときに送信元 Replication Server からのルートを削除しなければならない場合、または直接ルートのログイン名とパスワードを追加または変更しようとする場合にだけ使用してください。送信先 Replication Server を正常に更新するには、できるだけ **with nowait** 句を使用しないでください。

この句を使用すると、ルートのアウトバウンドキューにトランザクションが含まれている場合でも、ルートが強制的に削除されます。その結果、Replication Server はプライマリコネクションからトランザクションを破棄する可能性があ

ります。この句は、専用ルートが送信先 Replication Server と通信できない場合でも、専用ルートを削除するように、Replication Server に指示します。

with nowait 句を使用した後、**sysadmin purge_route_at_replicate** コマンドを使用して、サブスクリプションやルート情報などのプライマリ Replication Server に対する参照をレプリケート Replication Server のシステムテーブルからすべて削除します。

- **with nowait** を使用してルートを削除したら、(以前の) 送信先サイトで **sysadmin purge_route_at_replicate** を使用して、送信先のシステムテーブルからサブスクリプションとルート情報を削除できます。
- ルートを削除する Replication Server が他の Replication Server の中間サイトである場合は、ルートを削除できません。詳細については、『Replication Server 管理ガイド 第 1 巻』を参照してください。
- ERSSD を使用する Replication Server では、削除するルートがこの送信元を始点とする最後のルートである場合、次のようになります。
  - ERSSD の Replication Agent が停止する。
  - ルートの削除処理の最後に、ERSSD からのログ転送が無効になる。

パーミッション

drop route には、"sa" パーミッションが必要です。

参照：

- alter route (213 ページ)
- create connection (287 ページ)
- create route (368 ページ)
- sysadmin purge_route_at_replicate (499 ページ)

drop schedule

コマンドを実行するスケジュールを削除します。

構文

```
drop schedule sched_name
```

パラメータ

- **sched_name –** 削除するスケジュールの名前です。

例

- **例 1 – schedule1** を削除するには、次のように入力します。

```
drop schedule schedule1
```

使用法

Replication Server からスケジュールを削除します。

パーミッション

drop schedule には、"sa" パーミッションが必要です。

参照：

- admin schedule (69 ページ)
- alter schedule (221 ページ)
- create schedule (373 ページ)

drop subscription

データベース複写定義、テーブル複写定義、ファンクション複写定義、アーティクル、パブリケーションのサブスクリプションを削除します。

構文

```
drop subscription sub_name
for {table_rep_def | function_rep_def |
{article article_name in pub_name |
    publication pub_name | database replication definition db_repdef
        with primary at data_server.database}
 with replicate at data_server.database
[without purge [with suspension
     [at active replicate only]] |
   [incrementally] with purge]
```

パラメータ

- **sub_name –** 削除するサブスクリプションの名前です。パブリケーション内のアーティクルのサブスクリプションを削除する場合は、そのパブリケーションサブスクリプション名を指定します。
- **for table_rep_def** – サブスクリプションの対象となるテーブル複写定義の名前を指定します。
- **for function_rep_def –** サブスクリプションの対象となるファンクション複写定義の名前を指定します。

- **for article article_name in pub_name –** サブスクリプションの対象となるアーティクルの名前と、そのアーティクルを含むパブリケーションの名前を指定します。
- **for publication pub_name –** サブスクリプションの対象となるパブリケーションの名前を指定します。
- **for database replication definition db_repdef –** サブスクリプションの対象となるデータベース複写定義の名前を指定します。
- **with primary at data_server.database –** プライマリデータのロケーションを指定します。プライマリデータベースがウォームスタンバイアプリケーションの一部である場合、*data_server.database* は論理データサーバと論理データベースの名前になります。パブリケーションまたはアーティクルへのサブスクリプションの場合にのみ、この句を指定してください。
- **with replicate at data_server.database –** レプリケートデータのロケーションを指定します。レプリケートデータベースがウォームスタンバイアプリケーションの一部である場合、*data_server.database* は論理データサーバと論理データベースの名前になります。
- **without purge –** サブスクリプションによって複写された複写コピーのローをそのまま残すように、Replication Server に指示します。

  ファンクション複写定義へのサブスクリプションは、常に、レプリケートデータをパージすることなく削除されます。 テーブル複写定義またはパブリケーションへのサブスクリプションでは、**without purge** または **with purge** のどちらかを選択してください。 データベース複写定義へのサブスクリプションの場合は、**without purge** を指定する必要があります。 レプリケートデータベースが Adaptive Server でない場合は、**without purge** オプションを指定してテーブルサブスクリプションを削除することのみが可能です。

  direct_load を指定して作成されたサブスクリプションがまだ有効になっていない、または回復不能なエラーが発生した場合は、**without purge** オプションを指定してサブスクリプションを削除することのみが可能で、レプリケートデータベースの DSI が起動している必要があります。 **direct_load** を指定して作成されたサブスクリプションに対して **drop subscription** コマンドを発行すると、すべてのマテリアライゼーションスレッドが停止し、キャッチアップキューが削除され、テーブルの DML 操作が省略されます。
- **with suspension – without purge** 句とともに使用すると、サブスクリプションを削除した後に DSI をサスペンドするため、サブスクリプションローを手動で削除できるようになります。データベースがウォームスタンバイアプリケーションの一部である場合、**with suspension** はアクティブデータベースとスタンバイデータベースの DSI スレッドをサスペンドします。両方のデータベースからサブスクリプションローを削除してください。
- **with suspension at active replicate only – without purge** 句とともに使用すると、サブスクリプションを削除した後に DSI をサスペンドするため、サブスクリプ

ションローを手動で削除できるようになります。ウォームスタンバイアプリケーションでは、スタンバイ DSI はサスペンドされません。これにより、Replication Server が、アクティブデータベースからスタンバイデータベースに削除トランザクションを複写できるようになります。

- **incrementally – with purge** 句とともに使用すると、一度に 1000 ローずつ削除することを指定します。
- **with purge –** テーブル複写定義、アーティクル、またはパブリケーションとともに使用して、サブスクリプションが複写した (レプリケートテーブル内の) ローを削除するように、Replication Server に指示します。 レプリケートデータベースが Adaptive Server でない場合は、**without purge** オプションを指定してテーブルサブスクリプションを削除することのみが可能です。

  ファンクション複写定義へのサブスクリプションは、常に、レプリケートデータをパージすることなく削除されます。テーブル複写定義またはパブリケーションへのサブスクリプションでは、**without purge** または **with purge** のどちらかを選択してください。

例

- **例 1 –** *authors_rep* テーブル複写定義に対する *authors_sub* サブスクリプションを削除します。レプリケートデータは SYDNEY_DS データサーバの *pubs2* データベース内にあります。サブスクリプションによって複写されたローが別のサブスクリプションに含まれていない場合、これらのローはレプリケートテーブルからパージされます。

```
drop subscription authors_sub
for authors_rep
with replicate at SYDNEY_DS.pubs2
with purge
```

- **例 2 –** *titles_rep* テーブル複写定義の *titles_sub* サブスクリプションを削除します。レプリケートデータは SYDNEY_DS データサーバの *pubs2* データベース内にあります。サブスクリプションによって複写されたローは、レプリケートテーブルに保持されます。

```
drop subscription titles_sub
for titles_rep
with replicate at SYDNEY_DS.pubs2
without purge
```

- **例 3 –** *myproc_rep* ファンクション複写定義の *myproc_sub* サブスクリプションを削除します。レプリケートデータは SYDNEY_DS データサーバの *pubs2* データベース内にあります。サブスクリプションデータはパージされません。

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

- **例 4 –** アーティクル *titles_art* のサブスクリプションを削除します。このアーティクルは、パブリケーション *pubs2_pub* のサブスクリプション *pubs2_sub* に含まれています。プライマリデータは、TOKYO_DS データサーバの *pubs2* データベース内にあり、レプリケートデータは、SYDNEY_DS データサーバの *pubs2* データベース内にあります。サブスクリプションを介して複写されたローは、対象となっているレプリケートテーブル内に残ります。アーティクルサブスクリプションを削除したら、アーティクルを削除できます。

```
drop subscription pubs2_sub
for article titles_art in pubs2_pub
with primary at TOKYO_DS.pubs2
with replicate at SYDNEY_DS.pubs2
without purge
```

- **例 5 –** パブリケーション *pubs2_pub* の *pubs2_sub* というサブスクリプションを削除します。ここでは、プライマリデータは、TOKYO_DS データサーバの *pubs2* データベースにあり、レプリケートデータは、SYDNEY_DS データサーバの *pubs2* データベースにあります。サブスクリプションによって複写されたローが他のサブスクリプション含まれていない場合、対象となるレプリケートテーブルからパージされます。

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

- **例 6 –** *pubs2_sub* というデータベースサブスクリプションを削除します。**without purge** オプションを指定すると、このサブスクリプションによってレプリケートに追加されたローは削除されません。

```
drop subscription pubs2_sub
for database replication definition pubs2_rep
with primary at NEWYORK_DS.pubs2
with replicate at TOKYO_DS.pubs2
without purge
```

使用法

- サブスクリプションを削除すると、Replication Server はそのサブスクリプションによって指定されていたデータの複写を停止します。
- **drop subscription** は、サブスクリプションを作成した Replication Server で実行します。
- オブジェクトに対するサブスクリプションをすべて削除するまでは、テーブル複写定義、ファンクション複写定義、アーティクル、またはパブリケーションサブスクリプションは削除できません。

without purge 句

- **without purge** は、テーブル複写定義、データベース複写定義、またはパブリケーションのサブスクリプションを削除するときに使用します。複写されたローは、レプリケートテーブルに保持されます。
- テーブル複写定義またはパブリケーションのサブスクリプションを削除するときは、**without purge** または **with purge** のいずれかを指定します。
- ファンクション複写定義へのサブスクリプションを削除する場合、このサブスクリプションは必ず "パージされずに" 削除されるため、**without purge** を指定する必要はありません。
- パブリケーションサブスクリプションを "パージせず" に削除すると、そのアーティクルサブスクリプションもすべて削除されます。

with purge 句

- **with purge** 句は、サブスクリプションによって複写された (レプリケートテーブル内の) ローを削除するときに使用します。サブスクリプションローは、レプリケートサイトで他のサブスクリプションに属していないかぎり、すべてパージされます。
- **with purge** を使用すると、Replication Server は、レプリケートデータベースから削除される一連のローを選択し、次に、選択されたローを他のサブスクリプションに対して評価し、そのローを削除するかどうかを決定します。レプリケートデータベースのメンテナンスユーザは、テーブルに対して **select** パーミッションを持っていなければなりません。
- **with purge** を使用した削除は、レプリケートデータベース内の **rs_select_with_lock** ファンクション文字列によって実行された単一のトランザクション内で発生します。
- **with purge** と **incrementally** を使用した削除は、一度に 1000 ローに対して実行されます。このオペレーションは、レプリケートデータベース内の **rs_select** ファンクション文字列によって実行されます。
- パブリケーションサブスクリプションを "パージして" 削除する場合、そのアーティクルサブスクリプションは、アーティクルがパブリケーションに追加された順序とは逆の順序で、一度に 1 つずつ削除されます。
- レプリケートデータベースが Adaptive Server でない場合、**with purge** オプションでサブスクリプションを削除することはできません。

パーミッション

drop subscription は、レプリケートサイトでは "create object"、プライマリ Replication Server では "primary subscribe" パーミッションが必要です。

drop subscription ... with purge では、メンテナンスユーザがレプリケートテーブルに対して **select** パーミッションを持っていることも必要です。

参照：
- check subscription (233 ページ)
- create subscription (376 ページ)
- define subscription (395 ページ)
- drop article (403 ページ)
- drop function replication definition (411 ページ)
- drop publication (418 ページ)
- drop replication definition (419 ページ)
- resume connection (436 ページ)
- rs_select (571 ページ)
- rs_select_with_lock (573 ページ)

drop user

Replication Server のユーザログイン名を削除します。

構文

```
drop user user
```

パラメータ

- **user –** 削除するユーザログイン名です。

例

- **例 1 –** Replication Server からログイン名 "louise" を削除します。

  ```
  drop user louise
  ```

使用法

- **drop user** は、Replication Server ログイン名を削除するときに使用します。
- ログイン名を作成した Replication Server で、このコマンドを実行してください。

パーミッション

drop user には、"sa" パーミッションが必要です。

参照：
- alter user (224 ページ)

- create user (393 ページ)

grant

ユーザにパーミッションを割り当てます。

構文

```
grant {sa | create object | primary subscribe |
 connect source}
 to user
```

パラメータ

- **sa –** "sa" パーミッションを持つユーザは、どの RCL コマンドでも実行できます。
- **create object –** 複写定義、サブスクリプション、ファンクション文字列などの Replication Server オブジェクトを作成、変更、削除するための権限を、指定したユーザに付与します。
- **primary subscribe –** プライマリデータが現在の Replication Server によって管理されている複写テーブルのサブスクリプションを作成する権限を、指定したユーザに付与します。
- **connect source –** このパーミッションは、Replication Server にログインする RepAgent と他の Replication Server に付与されます。
- **user –** パーミッションを付与するユーザのログイン名です。

例

- **例 1 –** ユーザ "thom" がすべての Replication Server コマンドを実行できるようにします。

  ```
  grant sa to thom
  ```

- **例 2 –** ユーザ "louise" がサブスクリプションを作成できるようにします。

  ```
  grant primary subscribe to louise
  ```

使用法

- "sa" ユーザの "sa" パーミッションを取り消すことはできません。
- RSI または RepAgent には、"connect source" パーミッションが必要です。詳細については、使用しているプラットフォーム用の『Replication Server インストールガイド』および『Replication Server 設定ガイド』を参照してください。
- このマニュアルで説明する各 RCL コマンドには、そのコマンドを実行するために必要な最小限のパーミッションが示されています。すべてのコマンドの最

小限のパーミッションのリストについては、『Replication Server 管理ガイド 第 1 巻』を参照してください。
- どのパーミッションも付与されていないユーザは、Replication Server アクティビティのモニタのみを行うことができます。

パーミッション

grant には、"sa" パーミッションが必要です。

参照：
- revoke (443 ページ)

ignore loss

Replication Server がロスを検出した後に、メッセージを受け入れることができるようにします。

構文

```
ignore loss
 from data_server.database
[to {data_server.database | replication_server}]
```

パラメータ

- **from data_server.database** – メッセージのロスを無視するプライマリデータサーバとデータベースを指定します。
- **to data_server.database –** 失ったメッセージの送信先データサーバとデータベースを指定します。
- **to replication_server –** 失ったメッセージの送信先 Replication Server を指定します。

使用法

- Replication Server は、キューの再構築時、またはリカバリモードでのトランザクションログのリプレイ時にロスを検出します。
- Replication Server は、管理しているレプリケートデータベースへのコネクションでメッセージのロスを検出します。
- Replication Server がアクティブデータベースとスタンバイデータベース間で検出するロスを除き、ウォームスタンバイデータベースでは、*data_server.database* に論理コネクション名を使用してください。これらのロスを無視するには、物理 *data_server.database* 名を使用します。

- 直接ルートが存在する場合、送信先 Replication Server は、送信元 Replication Server からのメッセージのロスを検出します。両方の Replication Server ログファイルをチェックして、ロスが検出されているかどうかを確認してください。
- Replication Server がロスを検出すると、**ignore loss** を実行するまで、コネクションでメッセージが拒否されます。
- **ignore loss** を実行した後、メッセージの送信が再び開始される前に、多少の更新が必要な場合があります。
- **ignore loss** を実行した後、複写データを最新の状態にするため、いくつかのプロシージャが必要です。

リカバリ手順の詳細については、『Replication Server 管理ガイド 第 2 巻』を参照してください。

パーミッション

ignore loss には、"sa" パーミッションが必要です。

参照：
- allow connections (129 ページ)
- configure route (266 ページ)
- rebuild queues (434 ページ)
- set log recovery (447 ページ)

move primary

エラークラスまたはファンクション文字列クラスのプライマリ Replication Server を変更する。

構文

```
move primary
of {[replication server] error class | function string class}
class_name
to replication_server
```

パラメータ

- **replication server –** Replication Server エラークラスを修正する場合に使用します。データサーバのエラークラスは修正されません。

- **error class –** エラークラスのプライマリ Replication Server を変更することを指定します。
- **function string class –** ファンクション文字列クラスのプライマリ Replication Server を変更することを指定します。
- **class_name –** プライマリ Replicaation Server を変更する、エラークラスまたはファンクション文字列クラスの名前です。
- **replication_server –** エラークラスまたはファンクション文字列クラスの新しいプライマリ Replication Server を指定します。**move primary** は新しいプライマリ Replication Server で実行する必要があるため、これはコマンドが実行される Replication Server の名前になります。

例

- **例 1 –** *pubs2_db_err_class* エラークラスのプライマリ Replication Server を SYDNEY_RS Replication Server に変更します。このコマンドは SYDNEY_RS で入力します。

```
move primary
of error class pubs2_db_err_class
to SYDNEY_RS
```

- **例 2 –** Replication Server エラークラス *my_rs_error_class* のプライマリ Replication Server を SYDNEY_RS Replication Server に変更します。このコマンドは SYDNEY_RS で入力します。

```
move primary
of replication server error class my_rs_error_class
to SYDNEY_RS
```

- **例 3 –** *sqlserver2_function_class* ファンクション文字列クラスのプライマリ Replication Server を SYDNEY_RS Replication Server に変更します。このコマンドは SYDNEY_RS で入力します。

```
move primary
of function string class sqlserver2_function_class
to SYDNEY_RS
```

使用法

- ルート指定構成を変更した場合は、**move primary** を使用して、エラー応答とファンクション文字列を、これらを必要とする Replication Server に新しいルート経由で分配できるようにします。
- **move primary** は、新しいプライマリ Replication Server で実行してください。
- **move primary** を使ってプライマリ Replication Server を A から B に変更できるのは、A から B と B から A へのルートがある場合だけです。
- システム提供の *rs_sqlserver_function_class* のプライマリサイトは、ユーザが割り当てるまで存在しません。*rs_default_function_class* と *rs_db2_function_class*

は、システム提供クラスであり、修正することはできません。また、これらのクラスにはプライマリサイトはありません。

- 親クラスが *rs_default_function_class* または *rs_db2_function_class* である場合を除き、派生ファンクション文字列クラスのプライマリサイトはその親クラスのサイトです。この場合、派生クラスのプライマリサイトは、そのクラスが作成されたサイトになります。
- *rs_sqlserver_function_class* を使用する場合は、プライマリサイトを指定してから、デフォルトのファンクション文字列を修正します。ファンクション文字列クラスのプライマリサイトを指定するには、プライマリサイトで **create function string class rs_sqlserver_function_class** を実行します。その後、**move primary** コマンドを使用して、そのクラスのプライマリサイトを変更します。
- デフォルトのエラークラス *rs_sqlserver_error_class* と *rs_repserver_error_class* には、ユーザが割り当てるまでプライマリサイトはありません。**assign action** を使用してデフォルトのエラーアクションを変更する前に、プライマリサイトを指定する必要があります。プライマリサイトを指定するには、プライマリサイトで **create error class rs_sqlserver_error_class** または **create replication server error class rs_repserver_error_class** を実行してください。その後、**move primary** を使用してプライマリサイトを変更します。

パーミッション

move primary には、"sa" パーミッションが必要です。

参照：

- alter error class (182 ページ)
- alter route (213 ページ)
- assign action (226 ページ)
- create error class (308 ページ)
- create function string class (334 ページ)

rebuild queues

Replication Server のステーブルキューを再構築します。

構文

```
rebuild queues
```

使用法

- 失敗または消失したパーティションをリカバリするために、ステーブルキューを再構築します。

警告！ このコマンドを使用するときは、『Replication Server 管理ガイド 第 2 巻』の説明に必ず従ってください。**rebuild queues** を使用すると、複写システムからメッセージが削除されるため、他の問題の解決が難しくなることがあります。

- キューを再構築する前に、障害のあったパーティションを削除し、必要に応じてそれらを置き換えてください。削除されたパーティションは、**rebuild queues** が実行されるまでは、実際にシステムから削除されないことがあります。
- **rebuild queues** は、このコマンドが実行された Replication Server から、他のすべての Replication Server を切断します。キューが再構築されるまで接続は拒否されます。
- **rebuild queues** は、Replication Server のすべてのステーブルキューをクリアします。キューをクリアするとき、使用中の障害があるパーティションは "処理しません"。
- (**-M** コマンドラインフラグを使用して) Replication Server をスタンドアロンモードで起動し、**rebuild queues** を実行すると、Replication Server はリカバリモードになります。
- 再構築されたステーブルキューにメッセージをリストアするときに、Replication Server は、キューからクリアされたデータがリカバリされたのか、失われたのかを判断します。キューが再構築された Replication Server のログファイルと、キューが再構築された Replication Server からの直接ルートを持つ Replication Server のログファイルを見て、エラーメッセージが書き込まれていないか確認してください。ロス検出には時間がかかることがあります。これは、各プライマリデータベースまたはアップストリームサイトから、新しいデータを流す必要があるためです。
- ロスが検出された場合は、サブスクリプションを再作成するか、オフラインダンプからデータをリカバリする必要があることがあります。
- **rebuild queues** を使用するときにサブスクリプションがマテリアライズ中の場合は、サブスクリプションを削除して、もう一度作成してください。マテリアライゼーションが正常に完了したように見える場合でも、一部のデータが失われている可能性があります。
- キューが再構築されると、Replication Server は、現在の Replication Server へのルートを持つ Replication Server からのバックログされたメッセージを要求することによって、失われたメッセージのリストアを試みます。
- 特定のデータベースコネクションまたはルートに対するキューを再構築することはできません。

リカバリ手順の詳細については、『Replication Server 管理ガイド 第 2 巻』を参照してください。

パーミッション

rebuild queues には、"sa" パーミッションが必要です。

参照：

- add partition (52 ページ)
- alter partition (196 ページ)
- configure connection (238 ページ)
- create partition (339 ページ)
- drop partition (417 ページ)
- ignore loss (431 ページ)
- resume log transfer (440 ページ)
- set log recovery (447 ページ)

resume connection

サスペンドされているコネクションをレジュームします。

構文

```
resume connection to data_server.database
[skip [n] transaction | execute transaction | skip to resync marker]
```

パラメータ

- **data_server** – コネクションをレジュームするデータベースを保持するデータサーバの名前です。
- **database** – コネクションをレジュームするデータベースの名前です。
- **skip [n] transaction** – コネクションキュー内の指定した数のトランザクションを省略してからコネクションをレジュームするように、Replication Server に指示します。 省略されたトランザクションは、データベースの例外ログと、Replication Server ログまたは **sysadmin dump_file** コマンドで指定した代替ログファイルに書き込まれます。 **resume connection** が省略できるトランザクションの最大数は、ステーブルキュートランザクション (SQT) にある読み取られていないコミット済みトランザクションの数です。

  SQT トランザクション数が指定された省略数の値より少ない場合、Replication Server は後続のトランザクションの受信を待ちません。 このため、Replication Server は SQT 内に存在する数より多くの省略対象トランザクションを送信することはありません。

  n が指定されていない場合、Replication Server はコネクションのキューにある 2 番目のトランザクションの実行をレジュームします。

- **execute transaction –** システムトランザクションが DSI キューにある最初のトランザクションの場合、DSI の起動後にシステムトランザクションの適用に対する Replication Server の制限を無効にします。
- **skip to resync marker –** Replication Server が Replication Agent からのデータベースダンプマーカを受け取るまでは、指定されたレプリケートデータベースで DSI アウトバウンドキュー内のトランザクションをスキップするよう Replication Server に指示します。レプリケートデータベース内のデータはダンプの内容によって置き換えられることになっているので、Replication Server はアウトバウンドキュー内のレコードの処理をスキップします。

例

- **例 1 –** SYDNEY_DS データサーバの *pubs2* データベースへのコネクションをレジュームします。

```
resume connection to SYDNEY_DS.pubs2
```

- **例 2 –** 2 つのトランザクションを省略した後に、SYDNEY_DS データサーバの *pubs2* データベースへのコネクションをレジュームします。トランザクションは、データベースの例外ログと Replication Server ログに書き込まれます。

```
resume connection to SYDNEY_DS.pubs2 skip 2 transaction
```

- **例 3 –** 2 つのトランザクションを省略した後に、SYDNEY_DS データサーバの *pubs2* データベースへのコネクションをレジュームします。 トランザクションは、データベースの例外ログと `SYDNEY_RS.log` ファイルに書き込まれます。最後の **sysadmin dump_file** コマンドによって、`SYDNEY_RS.log` ファイルがクローズされます。

```
sysadmin dump_file SYDNEY_RS.log
resume connection to SYDNEY_DS.pubs2 skip 2 transaction
sysadmin dump_file
```

- **例 4 –** レプリケートデータベースのアウトバウンドキューからデータを削除し、プライマリデータベースの Replication Agent からの再同期マーカを待機するように Replication Server に指示します。

```
resume connection to SYDNEY_DS.pubs2 skip to
resync marker
```

使用法

- コネクションをレジュームすることによって、サスペンドされていたデータベースで複写アクティビティを再び開始できます。
- コネクションをサスペンドすると、**alter connection** を使用してコネクションを変更したり、サスペンドされたデータベースでメンテナンスを実行することが

できます。コネクションは、サブスクリプションのマテリアライゼーション中またはマテリアライゼーション解除中にもサスペンドされます。

- Replication Server は、エラー発生時にデータベースコネクションをサスペンドすることがあります。
- **resume connection** は、エラーによってサスペンドされたコネクションをレジュームするためにも使用されます。
- システムトランザクションが実行されたと判断した場合には、**skip transaction** 句を使用してください。
- **execute transaction** 句は、システムトランザクションの実行に失敗したときに、その実行の妨げとなった問題を解決できた場合にのみ使用してください。システムトランザクションは、**begin tran/commit tran** で囲まれていません。
  Replication Server が、最初のトランザクションとしてシステムトランザクションを使用して再起動されると、次のメッセージが表示されます。

```
E. 1998/02/16 14:43:49. ERROR #5152 DSI (206 hookip01.rdb1) -
dsisched.c (2196)
 There is a system transaction whose state is not known. DSI will
be shut down.
```

  データベースでこのトランザクションが実行されたかどうかを確認し、必要に応じて **skip transaction** または **execute transaction** を使用してください。
- **skip to resync** を設定すると、Replication Server は Replication Server ログ内またはデータベース例外ログ内でスキップされたトランザクションをログに記録しません。**skip [n] transaction** を設定すると、Replication Server はスキップされたトランザクションをログに記録します。
  resume connection に **skip to resync marker** を付けて実行した後、Replication Agent が正しいマーカを発行しないか、間違ったコネクションに対してマーカが発行されるか、あるいはその他の理由で DSI コネクションがデータベース再同期マーカを処理することは想定されていない場合、データベース再同期マーカを待たずに通常のレプリケーション処理をレジュームできます。そのためには、**suspend connection** を実行してから **resume connection** を実行します。その際、**skip to resync** オプションは指定しません。

注意： **resume connection** を **skip to resync marker** オプションを付けて間違ったコネクションで実行すると、レプリケートデータベースのデータが非同期となります。

パーミッション

resume connection には、"sa" パーミッションが必要です。

参照：

- activate subscription (48 ページ)
- alter connection (134 ページ)

- assign action (226 ページ)
- create connection (287 ページ)
- drop connection (407 ページ)
- drop subscription (424 ページ)
- suspend connection (452 ページ)

resume distributor

データベースへのコネクションのサスペンドされているディストリビュータスレッドをレジュームする。

構文

```
resume distributor data_server.database [skip transaction]
```

パラメータ

- **data_server –** データサーバの名前です。データベースがウォームスタンバイアプリケーションの一部である場合、*data_server* は論理データサーバ名になります。
- **database –** データベースの名前です。データベースがウォームスタンバイアプリケーションの一部である場合、*database* は論理データベース名になります。
- **skip transaction –** コネクションのキューにある 2 番目のトランザクションの実行をレジュームするように、Replication Server に指示します。最初のトランザクションは、データベースの例外ログに書き込まれます。

例

- **例 1 –** 論理データサーバ LDS と *pubs2* データベースのディストリビュータスレッドをレジュームします。

  ```
  resume distributor LDS.pubs2
  ```

使用法

- **resume distributor** は、**suspend distributor** を使用してサスペンドしたディストリビュータスレッド、または Replication Server によってサスペンドされたディストリビュータスレッドをレジュームするために使用します。
- 次の理由によってディストリビュータが停止したときに、**skip transaction** を使用してコネクションをレジュームします。
  - インバウンドキューのメッセージが 16,000 バイトより長く、サイトバージョンが Replication Server 12.5 以降にアップグレードされていない。

- ダウンストリーム Replication Server が *bigint* などの新機能のコマンドを受け入れることができない。

パーミッション

resume distributor には、"sa" パーミッションが必要です。

参照：
- suspend distributor (453 ページ)

resume log transfer

RepAgent が Replication Server に接続できるようにします。

構文

```
resume log transfer
 from {data_server.database | all}
```

パラメータ

- **data_server –** RepAgent が Replication Server に接続するデータベースがあるデータサーバの名前です。
- **database –** RepAgent が Replication Server に接続するデータベースです。
- **all –** Replication Server が管理するすべてのデータベースの RepAgent に接続を許可します。

例

- **例 1 –** Replication Server は、すべての RepAgent からのコネクションを受け入れます。

  ```
  resume log transfer from all
  ```

- **例 2 –** Replication Server は、SYDNEY_DS データサーバにある *pubs2* データベースの RepAgent からのコネクションを受け入れます。

  ```
  resume log transfer from SYDNEY_DS.pubs2
  ```

使用法

- Replication Server または複写システムをクワイスするときは、**suspend log transfer** を使用します。このコマンドを使用すると、Replication Server は RepAgent のコネクションを拒否します。

- **resume log transfer** を使用すると、RepAgent スレッドは **suspend log transfer** が実行された Replication Server に接続できるようになります。
- 通常、RepAgent は、**suspend log transfer** の実行後、**resume log transfer** によって Replication Server に再接続できるようになるまで、Replication Server へのコネクションをリトライします。ただし、何らかの理由で RepAgent が停止している場合、**resume log transfer** はコネクションを再起動しません。
- ERSSD からのログ転送をレジュームした後、リカバリデーモンはウェイクアップ時に ERSSD RepAgent を自動的に再起動します。

パーミッション

resume log transfer には、"sa" パーミッションが必要です。

参照：
- admin quiesce_check (66 ページ)
- admin quiesce_force_rsi (68 ページ)
- resume connection (436 ページ)

resume queue

16 キロバイトを超えるメッセージが渡された後に停止したステーブルキューを再起動します。このコマンドは、Replication Server のバージョンが 12.5 以降で、サイトバージョンが同じバージョンにアップグレードされていない場合のみ適用されます。

構文

```
resume queue, q_number, q_type [, skip transaction with large
message]
```

パラメータ

- **q_number –** ステーブルキューのキュー番号です。
- **q_type –** ステーブルキューのキュータイプです。アウトバウンドキューの場合は "0"、インバウンドキューの場合は "1" です。
- **skip transaction with large message –** 再起動後、最初に検出した大きなメッセージを SQM がスキップするように指定します。

例

- **例 1 –** アウトバウンドキュー #2 が RepAgent によって渡された最初の大きいメッセージをスキップするように指定します。

```
resume queue, 2, 0, skip transaction with large message
```

使用法

- このコマンドは、Replication Server がバージョン 12.5 以降であり、サイトバージョンがアップグレードされていない場合にのみ適用されます。
- サイトバージョンが 12.5 以降の場合、**resume queue** はメッセージをスキップしません。

パーミッション

resume route には、"sa" パーミッションが必要です。

参照：

- alter queue (197 ページ)

resume route

サスペンドされているルートをレジュームします。

構文

```
resume route to dest_replication_server
 [with primary at dataserver.database |
 skip transaction with large message]
```

パラメータ

- **dest_replication_server –** レジュームするサスペンドされているルートのある送信先 Replication Server の名前です。
- **with primary –** 専用ルートをレジュームするプライマリデータベースからのコネクションを指定します。
- **skip transaction with large message –** 16,000 バイトより大きいメッセージを持つ、最初に検出されたトランザクションを無視します。

例

- **例 1 –** SYDNEY_RS Replication Server へのルートをレジュームします。

  ```
  resume route to SYDNEY_RS
  ```

- **例 2 –** `NY_DS.pdb1` プライマリコネクションのために、RS_NY プライマリ Replication Server と RS_LON レプリケート Replication Server との間の専用ルートをレジュームするには、RS_NY で次のように入力します。

```
resume route to RS_LON
 with primary at NY_DS.pdb1
 go
```

使用法

- ルートをレジュームすると、Replication Server は、リモート Replication Server へのキューメッセージの送信を再開します。
- **resume route** は、エラーによってサスペンドされたルートをレジュームするためにも使用できます。
- **skip transaction with large message** は、レプリケートサイトのサイトバージョンが 12.1 以前の場合に直接ルートにのみ適用されます。

パーミッション

resume route には、"sa" パーミッションが必要です。

参照：
- alter route (213 ページ)
- create route (368 ページ)
- drop route (421 ページ)
- suspend route (455 ページ)

revoke

ユーザのパーミッションを取り消します。

構文

```
revoke {sa | connect source | create object |
 primary subscribe}
 from user
```

パラメータ

- **sa** – "sa" パーミッションが必要なコマンドを実行するためのパーミッションを取り消します。
- **connect source** – RepAgent または他の Replication Server が使用する RCL コマンドを実行するためのパーミッションを拒否します。
- **create object** – 複写定義、サブスクリプション、ファンクション文字列などの Replication Server オブジェクトを作成、変更、削除するためのパーミッションを拒否します。

- **primary subscribe –** プライマリデータが現在の Replication Server によって管理されている場合、複写テーブルに対してサブスクリプションを作成するパーミッションを取り消します。
- **user –** パーミッションを取り消すユーザのログイン名です。

例

- **例 1 –** ユーザ "thom" が Replication Server オブジェクトを作成または修正するコマンドを実行できないようにします。

```
revoke create object from thom
```

- **例 2 –** ユーザ "louise" がプライマリ Replication Server で "create object" パーミッションまたは "sa" パーミッションを持っていないかぎり、この Replication Server が管理するプライマリデータに対してサブスクリプションを作成できないようにします。

```
revoke primary subscribe from louise
```

使用法

- **revoke** には、"sa" パーミッションが必要です。
- "sa" ユーザログイン名の "sa" パーミッションを取り消すことはできません。

パーミッション

revoke には、"administrator" パーミッションが必要です。

参照：

- create replication definition (346 ページ)
- check subscription (233 ページ)
- create user (393 ページ)
- grant (430 ページ)

set

レプリケートコネクションの複写定義プロパティを制御します。

構文

```
set {autocorrection | dynamic_sql} {on | off}
 for replication_definition
with replicate at data_server.database
```

パラメータ

- **autocorrection –** 複写テーブル内の消失ローまたは重複ローが原因で発生する障害を防ぎます。デフォルトは off です。
- **dynamic_sql –** テーブルで動的 SQL の適用を考慮するかどうかを制御します。デフォルトは on です。
- **on –** 指定した複写定義のオートコレクションまたは動的 SQL を有効にします。
- **off –** 指定した複写定義のオートコレクションまたは動的 SQL を無効にします。
- **replication_definition –** オートコレクションまたは動的 SQL のステータスを変更する複写定義の名前です。
- **data_server –** オートコレクションまたは動的 SQL のステータスを変更するレプリケートデータベースがあるデータサーバの名前です。レプリケートデータベースがウォームスタンバイアプリケーションの一部である場合、*data_server* は論理データサーバ名になります。
- **database –** オートコレクションまたは動的 SQL のステータスを変更するレプリケートデータベースの名前です。レプリケートデータベースがウォームスタンバイアプリケーションの一部である場合、*database* は論理データベース名になります。

例

- **例 1 –** *publishers_rep* 複写定義 (SYDNEY_DS データサーバの *pubs2* データベースにあります) に対して、オートコレクションを有効にします。

```
set autocorrection on
 for publishers_rep
 with replicate at SYDNEY_DS.pubs2
```

- **例 2 –** *publishers_rep* 複写定義 (SYDNEY_DS データサーバの *pubs2* データベースにあります) に対して、動的 SQL を無効にします。

```
set dynamic_sql off
for publishers_rep
with replicate at SYDNEY_DS.pubs2
```

使用法

- **set dynamic_sql off** は、指定した複写定義とレプリケートコネクションに対して動的 SQL コマンドを無効にするときに使用します。
- **set autocorrection** は、ノンアトミックマテリアライゼーションの実行中に、キーの重複によるエラーの発生を防ぐために使用します。
- オートコレクションは、複写定義のサブスクリプションでノンアトミックマテリアライゼーション (**create subscription** で **without holdlock** を指定) を使用する場合にのみ有効にします。マテリアライゼーションが完了し、サブスクリプ

ションが VALID になったら、パフォーマンスを向上させるため、オートコレクションを無効にしてください。
- 作成した複写定義に対するオートコレクションのデフォルトは off です。

オートコレクションの動作

- **set autocorrection** は、Replication Server が複写テーブルへの挿入と更新を行う方法を決定します。オートコレクションをオンにすると、Replication Server は、更新または挿入の各オペレーションを、削除の後に挿入するよう変換します。たとえば、テーブルのプライマリバージョンにローを挿入したとき、すでにその複写コピーが存在しており、オートコレクションがオフである場合、このオペレーションはエラーになります。オートコレクションがオンであれば、Replication Server は、挿入オペレーションを削除してから挿入するように変換するため、既存のローによる挿入の失敗は起こりません。
  複写されるローでプライマリキーが変更された場合、Replication Server はローを挿入する前に、複写テーブルにある 2 つのローを削除します。更新前イメージと一致するプライマリキーのローと、更新後イメージと一致するプライマリキーのローを削除します。
- オートコレクションがオンの場合、プライマリデータベースでの挿入または更新は、レプリケートデータベースでの削除と挿入のトリガを起動します。削除トリガが起動されるのは、プライマリデータベースで挿入または更新されたローが、すでにレプリケートデータベースに存在していた場合だけです。
- Replication Server は、*rs_repobjs* システムテーブル内に、オートコレクションを有効にした複写定義用エントリを作成します。

オートコレクションと複写ストアドプロシージャ

- プライマリデータを修正する複写ストアドプロシージャを使用したことによってレプリケートデータベースで更新されたローに対してオートコレクションを実行することはできません。ストアドプロシージャの複写の詳細については、『Replication Server 管理ガイド 第 1 巻』を参照してください。

注意： 複写ストアドプロシージャを使用してプライマリデータを修正する場合は、レプリケート Replication Server で、ノンアトミックマテリアライゼーション中に失敗した更新と挿入を訂正するストアドプロシージャを記述してください。レプリケート Replication Server のストアドプロシージャは、更新または挿入オペレーションを削除と挿入を組み合わせたオペレーションとして実行することで、オートコレクションと同じように動作するように記述してください。または、更新や挿入の失敗を検出した後に訂正を行うストアドプロシージャを作成してください。

オートコレクションと replicate minimal columns

- 複写定義が **replicate minimal columns** を使用している場合は、**set autocorrection on** を設定できません。最少カラムを指定 (たとえば **alter replication definition** を

使用) する前に **set autocorrection on** を設定しても、オートコレクションは実行されません。Replication Server は、更新オペレーションに関する情報メッセージをログに記録します。

オートコレクションと `text`、`unitext`、または `image` データ型

- 複写定義の **replicate_if_changed** カラムリストに *text*、*unitext*、または *image* カラムが含まれている場合、複写定義に対してオートコレクションを有効にしようとするとエラーが発生します。オートコレクションを有効にするには、複写定義の **always_replicate** リストに *text*、*unitext*、*image* のすべてのカラムが含まれている必要があります。

オートコレクションとバルクコピーイン

通常の複写では、バルクオペレーションは、オートコレクションが有効な場合に無効になります。ただし、サブスクリプションマテリアライゼーションでは、障害からリカバリするノンアトミックサブスクリプションでない場合は、オートコレクションが有効になっていても、バルクオペレーションが適用されます。

パーミッション

set には、"create object" パーミッションが必要です。

参照：
- alter replication definition (199 ページ)
- create replication definition (346 ページ)
- create subscription (376 ページ)

set log recovery

オフラインダンプからログをリカバリするデータベースを指定します。

構文

```
set log recovery
 for data_server.database
```

パラメータ

- **data_server –** リカバリするデータベースがあるデータサーバです
- **database –** リカバリするデータベースです。

使用法

- Replication Server をスタンドアロンモードで再起動した後、**set log recovery** を実行します。
- リカバリモードに入るために、**set log recovery** を実行してから **allow connections** を実行します。Replication Server は、**set log recovery** で指定されたデータベースのリカバリモードで起動した RepAgent からのコネクションだけを受け入れます。これにより、新しいログレコードを受け入れる前に、古いログレコードが確実にリプレイされます。

リカバリ手順の詳細については、『Replication Server 管理ガイド 第 2 巻』を参照してください。

パーミッション

set log recovery には、"sa" パーミッションが必要です。

参照：
- allow connections (129 ページ)
- ignore loss (431 ページ)
- rebuild queues (434 ページ)

set proxy

別のユーザに切り替えます。

構文

```
set proxy [to] [user_name [verify password passwd]]
```

パラメータ

- **user_name –** 有効な Replication Server のログイン名です。
- **verify password –** Replication Server ユーザのパスワードを検証します。
- **passwd –** 有効な Replication Server ユーザのパスワードです。

使用法

- **set proxy***user_name* は、新しいユーザのパーミッションをすべて持ち、元のユーザのパーミッションは何も持たない新しいユーザに切り替えます。

- ユーザ名を指定しないで **set proxy** を入力すると、新しいユーザは "sa" パーミッションを持っているかどうかに関係なく、常に元のユーザに戻ることができます。
- **set proxy***user_name***verify password***passwd* を使用すると、**sa** パーミッションを持たないユーザを別のユーザに切り替えることができます。ただし、*user_name* に対して正しいパスワードを入力することが条件です。

パーミッション

set proxy*user_name* には、"sa" パーミッションが必要です。**set proxy** と **set proxy***user_name***verify password***passwd* は、すべてのユーザが実行できます。

参照：

- alter connection (134 ページ)
- alter route (213 ページ)
- configure replication server (238 ページ)
- create connection (287 ページ)
- create route (368 ページ)

show connection

コネクションスタックの内容をリストします。

構文

```
show connection
```

例

- **例 1 –** ost_replinuxvm_02 (ID サーバ) の後のコネクションスタックを表示し、ost_replinuxvm_03 へのゲートウェイを作成します。

```
isql -Usa -P -S ost_replinuxvm_02
1> connect to ost_replinuxvm_03
2> go
```

```
Gateway connection to 'ost_replinuxvm_03' is created.
```

```
1> show connection
2> go
```

```
ost_replinuxvm_03
ost_replinuxvm_02(IDServer)
```

使用法

- ゲートウェイで作成されたカスケードコネクションは、コネクションスタックで保持され、最初の **connect** コマンドを発行した Replication Server がスタックの一番下に置かれます。
- Replication Server 15.1 以前では、**disconnect** コマンドの動作が異なります。これらのバージョンでは、**disconnect** コマンドは、ゲートウェイモードを終了し、最初の **connect** コマンドを発行した Replication Server に稼働中のサーバのステータスを返します。コネクションスタックに Replication Server バージョン 15.2 と 15.1 以前が含まれる場合に **disconnect** コマンドを発行すると、**show connection** コマンドや **show server** コマンドを実行したときに、想定した出力が表示されない可能性があります。

パーミッション

このコマンドは、すべてのユーザが実行できます。

参照：

- connect (266 ページ)
- disconnect (402 ページ)
- show server (450 ページ)

show server

コネクションのスタックを指定して、現在稼働中のサーバを表示します。

構文

```
show server
```

例

- **例 1 –** ost_replinuxvm_02 から ost_replinuxvm_03 へのコネクションが作成された後、現在稼働中のサーバを表示します。

```
isql -Usa -P -S ost_replinuxvm_02
1> connect to ost_replinuxvm_03
2> go
```

```
Gateway connection to 'ost_replinuxvm_03' is created.
```

```
1> show server
2> go
```

```
ost_replinuxvm_03
```

使用法

使用法については、「**show connection**」を参照してください。

パーミッション

このコマンドは、すべてのユーザが実行できます。

参照：

- connect (266 ページ)
- disconnect (402 ページ)
- show connection (449 ページ)

shutdown

Replication Server を停止します。

構文

```
shutdown
```

例

- **例 1 –** Replication Server に停止するよう指示します。

  ```
  shutdown
  ```

使用法

shutdown コマンドは、Replication Server を停止するときに使用します。このコマンドは、追加コネクションを拒否し、プロセスを停止してから終了するように、Replication Server に指示します。

パーミッション

shutdown には、"sa" パーミッションが必要です。

suspend connection

データベースへのコネクションをサスペンドします。

構文

```
suspend connection
 to data_server.database
[with nowait]
```

パラメータ

- **data_server –** コネクションをサスペンドするデータベースがあるデータサーバの名前です。
- **database –** コネクションをサスペンドするデータベースの名前です。
- **with nowait –** コネクションをただちにサスペンドします。

例

- **例 1 –** SYDNEY_DS データサーバの *pubs2* データベースへのコネクションをサスペンドします。

  ```
  suspend connection to SYDNEY_DS.pubs2
  ```

使用法

- コネクションをサスペンドすることにより、データベースに対する複写のアクティビティが一時的に停止します。
- コネクションをサスペンドすると、その間に **alter connection** を使用してコネクションを変更したり、メンテナンスを実行したりできます。また、**suspend connection** を使用して、レプリケートデータベースを更新するタイミングを制御することもできます。
- コネクションがサスペンドされている間は、Replication Server がデータベースのトランザクションをステーブルキューに格納します。
- **with nowait** 句を指定せずに **suspend connection** を実行すると、Replication Server は実行中のトランザクションをすべて完了しようとします。しかし、データサーバへのコネクションは、トランザクションが完了する前にサスペンドされる場合があります。
- コネクションを再度アクティブにするには、**resume connection** を使用します。

パーミッション

suspend connection には、"sa" パーミッションが必要です。

参照：

- alter connection (134 ページ)
- create connection (287 ページ)
- drop connection (407 ページ)
- resume connection (436 ページ)

suspend distributor

プライマリデータベースへのコネクションのディストリビュータスレッドをサスペンドします。

構文

```
suspend distributor data_server.database
```

パラメータ

- **data_server –** データサーバの名前です。データベースがウォームスタンバイアプリケーションの一部である場合、*data_server* は論理データサーバ名になります。
- **database –** データベースの名前です。データベースがウォームスタンバイアプリケーションの一部である場合、*database* は論理データベース名になります。

例

- **例 1 –** LDS データサーバの *pubs2* データベースのディストリビュータスレッドをサスペンドします。

  ```
  suspend distributor LDS.pubs2
  ```

使用法

- **suspend distributor** は、プライマリデータベースへの論理コネクションまたは物理コネクションのディストリビュータスレッドをサスペンドするときに使用します。
- ディストリビュータスレッドをレジュームするには、**resume distributor** を使用します。
- ディストリビュータスレッドは、受信プライマリデータベーストランザクションを読み込み、サブスクライバに転送します。サブスクライバがなく、スタンバイデータベースだけのウォームスタンバイ環境でパフォーマンスを改善するには、ディストリビュータを停止してください。

パーミッション

suspend distributor には、"sa" パーミッションが必要です。

参照：

- resume distributor (439 ページ)

suspend log transfer

Replication Server から RepAgent を切断し、RepAgent が接続できないようにします。

構文

```
suspend log transfer
 from {data_server.database | all}
```

パラメータ

- **data_server –** RepAgent をサスペンドするデータベースがあるデータサーバです。
- **database –** RepAgent をサスペンドするデータベース、またはコネクションを禁止するデータベースです。
- **all –** すべての ReplAgent をサスペンドし、今後すべての RepAgent のコネクションを禁止するように Replication Server に指示します。

例

- **例 1 –** *pubs2* データベースの RepAgent を切断し、RepAgent が再接続できないようにします。

  ```
  suspend log transfer from TOKYO_DS.pubs2
  ```

- **例 2 –** 接続されているすべての RepAgent を切断し、どの RepAgent も Replication Server に再接続できないようにします。

  ```
  suspend log transfer from all
  ```

使用法

- **suspend log transfer** は、RepAgent を切断するときに使用します。これは、複写システムをクワイスするときの最初の手順です。**suspend log transfer** は、RepAgent を停止するわけではありません。

- RepAgent をサスペンドした後に、システムがクワイスされているかどうかを確認するには、**admin quiesce_check** を使用します。
- RepAgent が Replication Server に接続できるようにするには、**resume log transfer** を実行します。

パーミッション

suspend log transfer には、"sa" パーミッションが必要です。

参照：

- admin quiesce_check (66 ページ)
- admin quiesce_force_rsi (68 ページ)
- resume log transfer (440 ページ)

suspend route

別の Replication Server へのルートをサスペンドします。

構文

```
suspend route to dest_replication_server
 [with primary at dataserver.database]
```

パラメータ

- **dest_replication_server –** ルートをサスペンドする送信先 Replication Server の名前です。
- **with primary –** 専用ルートをサスペンドするプライマリデータベースからのコネクションを指定します。

例

- **例 1 –** SYDNEY_RS Replication Server へのルートをサスペンドします。

  ```
  suspend route to SYDNEY_RS
  ```

- **例 2 –** `NY_DS.pdb1` プライマリコネクションのために、RS_NY プライマリ Replication Server と RS_LON レプリケート Replication Server との間の専用ルートをサスペンドするには、RS_NY で次のように入力します。

  ```
  suspend route to RS_LON
   with primary at NY_DS.pdb1
   go
  ```

使用法

- **suspend route** は、別の Replicationi Server へのルートをサスペンドするときに使用します。このコマンドを使用すると、1 つの Replication Server から別の Replication Server へメッセージを送信するタイミングを制御することによって、ネットワークの使用を管理できます。
- ルートがサスペンドされている間、Replication Server は送信先 Replication Server へのメッセージをステーブルキュー内に保持します。
- 直接ルートだけをサスペンドできます。
- サスペンドしたルートを再度アクティブにするには、**resume route** を使用します。

パーミッション

suspend route には、"sa" パーミッションが必要です。

参照：
- alter route (213 ページ)
- resume connection (436 ページ)
- resume route (442 ページ)
- suspend connection (452 ページ)

switch active

ウォームスタンバイアプリケーションでアクティブデータベースを変更します。

構文

```
switch active
 for logical_ds.logical_db
to data_server.database
[with suspension]
```

パラメータ

- **logical_ds –** 論理コネクションの論理データサーバ名です。
- **logical_db –** 論理コネクションの論理データベース名です。
- **data_server –** 論理コネクションの新しいアクティブデータベースのデータサーバ名です。
- **database –** 論理コネクションの新しいアクティブデータベースのデータベース名です。

- **with suspension –** 切り替えが完了した後、新しいアクティブデータベースへの DSI コネクションをサスペンドします。

例

- **例 1 –** このコマンドは、アクティブなプロセスの切り替えを開始します。

```
switch active for LDS.pubs2 to OSAKA.pubs2
```

```
Switch of the active for this logical database is in progress.
```

使用法

- **switch active** は、ウォームスタンバイアプリケーションでスタンバイデータベースに切り替える手順の一部です。手順全体については、『Replication Server 管理ガイド 第 2 巻』を参照してください。
- **switch active** の実行はすぐに返されますが、**admin logical_status** で State of Operation in Progress (進行中のオペレーションのステータス) に “None” と表示されるまで、切り替えは完了しません。
- アクティブなプロセスの切り替えステータスをモニタするには、**admin logical_status** を使用してください。
- **with suspension** オプションを使用する場合は、切り替えが完了した後に、手動で新しいアクティブデータベースへの DSI コネクションをレジュームします。
- **switch active** の入力後、このコマンドをキャンセルするには、**abort switch** を使用します。

パーミッション

switch active には、“sa” パーミッションが必要です。

参照：

- abort switch (47 ページ)
- admin logical_status (64 ページ)
- create logical connection (338 ページ)
- wait for switch (529 ページ)

sysadmin apply_truncate_table

特定のテーブルに対する既存のすべてのサブスクリプションに対して、“subscribe to truncate table” オプションをオンまたはオフにすることによって、**truncate table** の複写を有効または無効にします。

構文

```
sysadmin apply_truncate_table, data_server,
 database, {table_owner | '' | ""}, table_name
{'on'| 'off'}
```

パラメータ

- **data_server** – レプリケートデータサーバの名前です。
- **database** – データサーバが管理するレプリケートデータベースの名前です。
- **table_owner** – レプリケートテーブルの所有者を指定します。所有者を指定しなかった場合、Replication Server は所有者を “dbo” に設定します。
- **table_name** – 既存のサブスクリプションに対して、“subscribe to truncate table” オプションをオンまたはオフにするレプリケートテーブルを示します。
- **on** – 既存のサブスクリプションに対して、“subscribe to truncate table” オプションをオンにします。
- **off** – 既存のサブスクリプションに対して、“subscribe to truncate table” オプションをオフにします。

例

- **例 1** – *pubs2* データベースで *emily* が所有する *publishers* テーブルのすべてのサブスクリプションに対して、“subscribe to truncate table” をオンにします。

```
sysadmin apply_truncate_table, SYDNEY_DS,
 pubs2, emily, publishers, 'on'
```

使用法

- **sysadmin apply_truncate_table** は、Adaptive Server バージョン 11.5 以降のデータベースで使用します。
- 複写定義でレプリケートテーブルの所有者を指定しなかった場合は、テーブル所有者名として '' (2 つの一重引用符文字) または "" (2 つの二重引用符文字) を入力します。
- 特定のデータベースの特定のテーブルのサブスクリプションでは、**truncate table** の複写をすべてサポートするか、一切サポートしないかのどちらかである必要があります。たとえば、**sysadminapply_truncate_table** がオフの場合、そ

のテーブルのすべてのサブスクリプションに対して、**sysadminapply_truncate_table** をオンにしない限り、“subscribe to truncate table” オプションを含む新しいサブスクリプションは作成できません。
新しいサブスクリプションに対して “subscribe to truncate table” オプションを設定する方法の詳細については、「**create subscription**」または「**define subscription**」を参照してください。

- Replication Server は、メンテナンスユーザとしてレプリケートデータベースで **truncate table** を実行します。メンテナンスユーザに付与されるパーミッションの中に “replication_role” があります。メンテナンスユーザの “replication_role” を取り消すと、次の場合を除き、**truncate table** を複写できなくなります。
  - メンテナンスユーザに “sa_role” が付与されている。
  - メンテナンスユーザがテーブルを所有している。
  - メンテナンスユーザに、データベース所有者としてのエイリアスが与えられている。
- ウォームスタンバイデータベースでは、**truncate table** に対してサブスクリプションを作成する必要はありません。**truncate table** コマンドの実行は、スタンバイデータベースに自動的に複写されます。スタンバイデータベースに対する **truncate table** の複写を、**alter logical connection** コマンドを使用してオンにしてください。

パーミッション

sysadmin apply_truncate_table には、“sa” パーミッションが必要です。

参照：

- create subscription (376 ページ)
- define subscription (395 ページ)

sysadmin cdb

SAP IQ への Real-Time Loading (RTL) の複写および Adaptive Server への High Volume Adaptive Replication (HVAR) において、最終的な変更を保管するデータベースを管理します。

構文

最終的な変更を保管するデータベースのホールド、点検、および解除には、以下を使用します。

```
sysadmin cdb, q_number, q_type,{hold | hold_next | unhold}
```

注意： データサーバインタフェースエグゼキュータ (DSI/E) スレッドでトランザクションをアクティブに実行している場合は、最初に **sysadmin cdb** に **hold** または

hold next を指定して実行しないと、**sysadmin cdb** を使用して最終的な変更を保管するデータベースの情報を表示できません。

最終的な変更を保管するデータベースの情報をすべて表示するか、特定の追跡テーブルのみの情報を表示するには、以下を使用します。

```
sysadmin cdb,
[q_number[,q_type][list[,["table_owner.]table_name"] |
[[dump_i | dump_d | dump_u | dump_nc],table_name] |dump_nc]]
```

パラメータ

- **hold –** 最終的な変更を保管するデータベースの現在のインスタンスのサスペンドを DSI/E に指示し、インスタンスを点検できるようにします。
- **hold_next –** コミットの準備が整った最初のトランザクションをコミットし、データベースインスタンスを解除した後、次のトランザクションを保持するように DSI/E に指示します。
- **unhold –** DSI によって保持されている最終的な変更を保管するデータベースのインスタンスをすべて解除し、通常の DSI/E アクティビティを再開するように DSI/E に指示します。
- **q_number –** レプリケートデータベースのアウトバウンド DSI ステーブルキューを指定します。キュー番号を確認するには、**admin who, sqm** コマンドの出力を調べます。
- **q_type –** ステーブルキューのタイプを指定します。0 はアウトバウンドキューで、1 はインバウンドキューです。デフォルトは 0 です。*q_type* を指定しない場合、デフォルト値が使用されます。
- **table_name –** レプリケートテーブル名を指定します。
- **list –** 最終的な変更を保管するデータベースに関する情報を表示します。テーブル名を指定しない場合、**list** によって、*q_number* で指定したアウトバウンド DSI ステーブルキューのすべてのインスタンスが表示されます。テーブルを指定すると、そのテーブルの内容のみが表示されます。
- **dump_i –** メモリ内 *Insert_Table* テーブルのすべてのカラムとローを含む結果を返します。
- **dump_u –** メモリ内 *Update_Table* テーブルのすべてのカラムとローを含む結果を返します。
- **dump_d –** メモリ内 *Delete_Table* テーブルのすべてのカラムとローを含む結果を返します。
- **dump_nc –** レプリケートテーブルに適用されるコンパイルできないコマンドを含む結果を返します。挿入では、すべてのカラムが返されます。削除では、プライマリキーのみが返されます。更新では、プライマリキーと更新されたカラムのみが返されます。

例

- **例 1 –** 最終的な変更を保管するデータベースに値が完全に格納された後に、検査のためにデータベースをサスペンドするように DSI/E に指示します。DSI/E がトランザクションをアクティブに処理していない場合は、次回最終的な変更を保管するデータベースを作成して値を格納するときに、ホールドのコマンドが有効になります。Replication Server は、最終的な変更を保管するデータベースが作成され、値が格納された後、最終的な変更を保管するデータベースの内容をレプリケートデータベースに適用できるようになるまで、DSI/E をサスペンドします。たとえば、現在の最終的な変更を保管するデータベースをサスペンドするには、以下を使用します。

  ```
  sysadmin cdb,101,hold
  ```

- **例 2 –** アクティブな DSI エグゼキュータスレッド、および Replication Server が処理中の最終的な変更を保管するデータベースの情報などの該当するステータスをリストします。

  ```
  sysadmin cdb
  ```

  出力には、2 つのデータサーバとそれぞれのデータベースの RTL ステータス、およびアクティブな DSI エグゼキュータ (DSI/E) スレッドのキュー番号とキュータイプが示されます。

  ```
  DSNameDBNameQueueQTypeCompileHoldCdbNameCommands_in_Group
  ------------------------------------------------------------
  IQSRVR2asiqdemo1050OnNo0
  IQSRVRiqdemo1040OnNo0
  ```

  ステータスカラムは次のとおりです。

  - Compile - RTL がアクティブな場合、ステータスは “On”。
  - Hold - 特定の DSI/E をホールドするために、**sysadmin cdb** に **hold** を指定して、同じ *q_number* と *q_type* に対して実行した場合、ステータスは “Yes”。
  - CdbName - Replication Server が処理中であるか、その DSI/E スレッドで “hold” 状態の最終的な変更を保管するデータベースの内部名。この例で、Replication Server は最終的な変更を保管するデータベースを処理していません。
  - Commands_in_Group - Replication Server がグループとしてコンパイルしているコマンドの数。この例では、処理中のコマンドはありません。

- **例 3 –** 特定の DSI/E スレッドに関する情報をリストする前に、**hold** 状態または **hold_next** 状態に設定することで、DSI/E をサスペンドする必要はありません。DSI/E が **hold** 状態や **hold_next** 状態でないため、コマンドを再び実行すると、Queue カラムと QType カラムの値を除く値が変わることがあります。

  ```
  sysadmin cdb,107,1
  ```

  出力：

```
QueueQTypeCdbNameTargetDBCompilable_Tables
------------------------------------------------------
1071asiqdemo_ws_46_3asiqdemo_ws1

Non_Compilable_TablesCommands_in_GroupCompiled_RowsNon_Compilable
Commands
------------------------------------------------------------
0320
```

- **例 4 –** DSI/E が実行している最終的な変更を保管するデータベースの情報を表示します。

  注意： DSI/E が実行している最終的な変更を保管するデータベースの情報を表示する前に、データベースを “**hold**” 状態でサスペンドする必要があります。

  ```
  sysadmin cdb,107,1,hold
  go
  sysadmin cdb,107,1,list
  go
  ```

  出力を以下に示します。

  ```
  CdbName Replicate_TableStatusCmd_Convert
  ------------------------------------------------
  asiqdemo_ws_46_3dbo.test_alltypes_ws_1compilablei2di

  AutoCorrectionNb_ColumnsPK_ColsCdbTable
  ------------------------------------------------------
  No2522test_allpes_ws_1_46_1

  Insert_TableInsertsUpdate_TableUpdates
  ---------------------------------------------
  rs_itest_allpes_ws_1_46_11rs_utest_allpes_ws_1_46_10

  Delete_TableDeletesNon_Compilable_Cmds
  ------------------------------------------------
  rs_dtest_allpes_ws_1_46_110

  Update_WorktableDelete_Worktable
  ----------------------------------------
  #rs_dtest_allpes_ws_1_46_1

  Reduced_InsertsReduced_UpdatesReduced_Deletes
  ---------------------------------------------
  000
  (1 rows affected)
  ```

  カラムは次のとおりです。

  - CdbName - Replication Server が処理中であるか、その DSI/E スレッドで “hold” 状態の最終的な変更を保管するデータベースの内部名。
  - Replicate_Table - レプリケートテーブル名。
  - Status - “compilable” または “noncompilable” のテーブル。

- Cmd_Convert - **none**、**ud2i**、**i2di**、**i2none** などの適用されたコマンドの変換。
- AutoCorrection - オートコレクションが適用されているかどうか。
- Nb_Columns - 最終的な変更を保管するデータベーステーブルのカラムの数。
- PK_Cols - 最終的な変更を保管するデータベーステーブルのプライマリキーカラムの数。
- CdbTable - 最終的な変更を保管するデータベーステーブルのユニークな名前。
- Insert_Table - 最終的な変更を保管するデータベースの挿入オペレーション用のメモリ内テーブルの名前。
- Inserts - 挿入数。
- Update_Table - 最終的な変更を保管するデータベースの更新オペレーション用のメモリ内テーブルの名前。
- Updates - 更新数。
- Delete_Table - 最終的な変更を保管するデータベースの削除オペレーション用のメモリ内テーブルの名前。
- Deletes - 削除数。
- Non_Compilable_Cmds - コンパイルできないコマンドの数。
- Update_Worktable - 更新の適用時にレプリケートデータサーバで作成されるワークテーブルの名前。このワークテーブルには値が格納され、レプリケートテーブルにジョインされます。
- Delete_Worktable - 削除の適用時にレプリケートデータサーバで作成されるワークテーブルの名前。このワークテーブルには値が格納され、レプリケートテーブルにジョインされます。
- Reduced_Inserts - コンパイルによって減少した挿入数。
- Reduced_Updates - コンパイルによって減少した更新数。
- Reduced_Deletes - コンパイルによって減少した削除数。

- **例 5 –** テーブルの情報を返すためのクエリに **dump_i**、**dump_u**、**dump_d**、または **dump_nc** のオプションを含めることで、最終的な変更を保管するデータベースの特定テーブルの詳細情報をリストできます。これらのオプションは、最終的な変更を保管するデータベースに実行される SQL **select** 文です。

  たとえば、*dbo.test_alltypes_msa_1* の内容および *Insert_Table* メモリ内テーブルを表示するには、以下を使用します。

  ```
  sysadmin cdb,106,0,dump_i,dbo.test_alltypes_msa_1
  ```

  複写が正常に実行されると、次の出力が表示されます。

  ```
  c1c2c3
  --------- --------- ---------
  4vddd
  3updqqq
  ```

```
(2 rows affected)
```

- **例 6 –** すべてのコンパイルできないコマンドを表示するには、以下を使用します。

```
sysadmin cdb,105,1,dump_nc
```

出力は次のようになります。

```
CmdText
-------------------------------------------------------------
1insert into dbo.test_alltypes_msa_1 (c1, c2, c3, c4, c5, c6, c7,
c8, c9, c10, c11, c12, c13, c14, c15, c16, c17, c18, c19, c20,
c21,
c22, c23, c24, c25) values (2, 1, 1, 'abcdefg', 'fghijkl',
‘20091125 08:50:42:100’, ‘20091125 08:51:00:000’, 1001.0000, 10,
10, 0.55554444333322221111000011122223333, 0x01234567,
‘20091125’, 1, 254.0000, 4967295, 65500, 92233720, 922337203,
‘08:50:42:113’, ‘mnopqrst’, ‘abcdefghijklmnopqrtsuvwxyz
jjjjjzzzzzzzzjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjz 0
ddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddd
ddddddddddddddddddd e
ddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddd
ddddddddddddddddddd f’, ‘uvwxy’,
convert(varchar,0x6100640064006400640064006400640064006400640064
0064006400640064006400640064006400640064006400640064006400640064
0064006400640064006400640064006400640064006400640064006400640064006400640
0640064006400640064006400640064006400640064006400640064006400640064006400
6400640064006400640064006400640064006400640064006400640064006400640064006
4006400640064006400640064006400640064002000650020006400640064006400640064006
4006400640064006400640064006400640064006400640064006400640064006400640064006
4006400640064006400640064006400640064006400640064006400640064006400640064006
4006400640064006400640064006400640064006400640064006400640064006400640064006
4006400640064006400640064006400640064006400640064006400640064006400640064006
4006400640064006400640064006400640020006600), 0x48656c6c6f20)
2update article set price = price*1.015 where id = 108
3delete from dbo.test_alltypes_msa_1 where c1=1 and c2=1 and c3=1
and c4=‘ABCDEFG’ and c5=‘FGHIJKL’ and c6=‘20091125 8:50:42:100’
and
c7=‘20091125 08:51:00:000’ and c8=1001.0000 and c9=10 and c10=10
and c11=0.55554444333322221111000011122223333 and c12=0x01234567
and c13= ‘20091125’ and c14=1 and c15=254.0000 and c16=4967295 and
c17=65500 and c18=92233720 and c19=922337203 and
c20=‘08:50:42:113’ and c21= ‘MNOPQRST’ and c23=‘UVWXY’
```

- **例 7 –** 最終的な変更を保管するデータベース内の特定テーブルに関する詳細情報を表示するには、以下を使用します。

```
sysadmin cdb,107,1,hold
go
sysadmin cdb,107,1,list,test_alltypes_ws_1
go
```

出力には、以下を含む情報が表示されます。

1. オペレーションのステータスおよびメモリ内テーブルの名前

```
CdbName Replicate_TableStatusCmd_Convert
-------------------------------------------------------------
```

```
asiqdemo_ws_46_3dbo.test_alltypes_ws_1compilablei2di

AutoCorrectionNb_ColumnsPK_ColsCdbTable
-------------------------------------------------
No2522test_allpes_ws_1_46_1

Insert_TableInsertsUpdate_TableUpdates
-----------------------------------------------------------------
--
rs_itest_allpes_ws_1_46_11rs_utest_allpes_ws_1_46_10

Delete_TableDeletesNon_Compilable_Cmds
-------------------------------------------------
rs_dtest_allpes_ws_1_46_110

Update_WorktableDelete_Worktable
----------------------------------------
#rs_dtest_allpes_ws_1_46_1

Reduced_InsertsReduced_UpdatesReduced_Deletes
---------------------------------------------
000
(1 row affected)
```

2. テーブルのすべてのカラムの情報

```
ColnameColtypeMaxlengthCdbtypeCdbvtypePrimary_keyChangedHasNul
l
-----------------------------------------------------------------
-
c1int488110
...
c8money1010110
...c25image5519011
(25 rows affected)
```

使用法

これらの SQL コマンドのいずれかをクエリに含めることで、最終的な変更を保管するデータベース内の特定メモリ内テーブルの詳細情報をリストできます。 メモリ内テーブルは内部処理用で、その内容はディスクに格納されません。

最初に **sysadmin net_change_db hold** または **sysadmin net_change_db hold next** を実行しないと、**sysadmin net_change_db list** を使用して最終的な変更を保管するデータベース情報を表示できません。

パーミッション

sysadmin net_change_db には、“sa” パーミッションが必要です。

参照：

- admin who (107 ページ)
- admin config (54 ページ)

sysadmin dropdb

ID サーバからデータベースを削除します。

構文

```
sysadmin dropdb, data_server, database
```

パラメータ

- **data_server –** データサーバの名前です。
- **database –** 削除するデータベースの名前です。

例

- **例 1 –** ID サーバから、SYDNEY_DS データサーバの *pubs2* データベースを削除します。

  ```
  sysadmin dropdb, SYDNEY_DS, pubs2
  ```

使用法

- **sysadmin dropdb** は、ID サーバからデータベースを削除するときに使用します。このコマンドは、ID サーバで実行してください。
- **sysadmin dropdb** は、ID サーバのシステムテーブルに、システムに存在しないデータベースについての情報が含まれているときにだけ使用してください。このような状況は、システム障害が起きた後にだけ発生します。
  たとえば、**drop connection** を使用してデータベースを削除した場合、データベースを ID サーバのテーブルから削除できることが、ネットワーク障害によって ID サーバに通知されない可能性があります。このようなとき、後から同じデータサーバとデータベースをシステムに追加しようとしても、そのデータベースとデータサーバは ID サーバのシステムテーブルにすでに登録されているため、この要求は失敗します。
- Replication Server を再インストールする場合は、**sysadmin dropdb** を使用して、Replication Server が管理していた RSSD を含む各データベースに対する ID サーバ情報を削除します。これが削除されていないと、Replication Server を再インストールするときにエラーが発生します。
- このコマンドで不正な引数を入力しても、ユーザには通知されません。

警告！ アクティブなコネクションを持つデータベースには、**sysadmin dropdb** を使用しないでください。

パーミッション

sysadmin dropdb には、“sa” パーミッションが必要です。

参照：

- sysadmin dropldb (467 ページ)

sysadmin dropldb

ID サーバから論理データベースを削除します。

構文

```
sysadmin dropldb, data_server, database
```

パラメータ

- **data_server –** 論理データサーバの名前です。
- **database –** 削除する論理データベースの名前です。

例

- **例 1 –** ID サーバから、LDS 論理データサーバの *pubs2* 論理データベースを削除します。

  ```
  sysadmin dropldb, LDS, pubs2
  ```

使用法

- **sysadmin dropldb** は、ID サーバから論理データベースを削除するときに使用します。このコマンドは、ID サーバで実行してください。
- **sysadmin dropldb** は、ID サーバのシステムテーブルに、システムに存在しない論理データベースについての情報が含まれているときにだけ使用してください。このような状況は、システム障害が起きた後にだけ発生します。
  たとえば、**drop logical connection** を使用して論理データベースを削除した場合、その論理データベースを ID サーバのテーブルから削除できることが、ネットワーク障害によって ID サーバに通知されない可能性があります。このようなとき、後から同じ論理データサーバと論理データベースをシステムに追加しようとすると、その論理データベースと論理データサーバは ID サーバのシステムテーブルにすでに登録されているため、この要求は失敗します。
- Replication Server を再インストールする場合は、まず **sysadmin dropldb** を使用して、Replication Server が管理していた各論理データベースに対する ID サーバ

情報を削除します。これが削除されていないと、Replication Server を再インストールするときにエラーが発生します。
- このコマンドで不正な引数を入力しても、ユーザには通知されません。

警告！ アクティブなコネクションを持つ論理データベースには、**sysadmin dropldb** を使用しないでください。

パーミッション

sysadmin dropldb には、“sa” パーミッションが必要です。

参照：
- sysadmin dropdb (466 ページ)

sysadmin drop_queue

ステーブルキューを削除します。このコマンドを使用して、失敗したマテリアライゼーションキューを削除します。

構文

```
sysadmin drop_queue, q_number, q_type
```

パラメータ

- **q_number –** Replication Server のサイト ID、またはキューの送信元か送信先となるデータベースのサイト ID です。
- **q_type –** キューのタイプです。

使用法

- **sysadmin drop_queue** は、サブスクリプションでリカバリできないエラーが発生し、手動でクリーンアップする必要があった後に、残っているマテリアライゼーションキューを停止し、削除するために使用します。

  警告！ sysadmin drop_queue は、失敗したマテリアライゼーションキューを削除する場合にのみ使用してください。
- **admin who** は、キューの *q_number* と *q_type* を調べるときに使用します。値は、コマンドの SQM スレッドの出力に表示されます。

パーミッション

sysadmin drop_queue には、“sa” パーミッションが必要です。

参照：

- rebuild queues (434 ページ)
- sysadmin purge_route_at_replicate (499 ページ)

sysadmin droprs

ID サーバから Replication Server を削除します。

構文

```
sysadmin droprs, replication_server
```

パラメータ

- **replication_server –** 削除する Replication Server の名前です。

例

- **例 1 –** ID サーバから SYDNEY_RS Replication Server を削除します。

  ```
  sysadmin droprs, SYDNEY_RS
  ```

使用法

- **sysadmin droprs** は、ID サーバから Replication Server を削除するときに使用します。このコマンドは、ID サーバでのみ実行できます。
- **sysadmin droprs** を使用できるのは、複写システムに存在しない Replication Server の情報が ID サーバに含まれている場合です。通常、このような状況はシステム障害によって発生します。たとえば、Replication Server のインストールが失敗したときに、ID サーバのシステムテーブルにその Replication Server のエントリが入力されてしまった場合、それ以降 Replication Server をインストールできなくなります。
- 無効な引数を入力しても、ユーザには通知されません。

警告！ アクティブな Replication Server を削除するときは、**sysadmin droprs** を慎重に使用してください。アクティブな Replication Server の適切な削除手順については、『Replication Server 管理ガイド 第 1 巻』を参照してください。

パーミッション

sysadmin droprs には、“sa” パーミッションが必要です。

sysadmin dump_file

Replication Server のステーブルキューをダンプするときに使用する代替ログファイル名を指定します。

構文

```
sysadmin dump_file [, file_name]
```

パラメータ

- **file_name –** ステーブルキューのダンプを書き込む新しいログファイルの名前です。

例

- **例 1 –** ステーブルキューのログ出力ファイルとして、*pubs2.log* を指定します。

  ```
  sysadmin dump_file, 'pubs2.log'
  ```

使用法

- **sysadmin dump_file** を使用してログファイル名を指定してから、**sysadmin dump_queue** を使用してログをファイルにダンプします。
- 現在のダンプファイルをデフォルトに再設定するには、ファイル名を指定しないで **sysadmin dump_file** を実行します。
- ファイル名を指定した場合、現在のダンプファイルはクローズされ、新しいファイルがオープンされます。新しいファイルは、指定したファイル名になります。
- デフォルトのダンプファイルは Replication Server ログです。このファイルのパスを表示するには、**admin log_name** を使用します。
- ログファイル名に文字と数字以外を入力する場合は、そのファイル名を引用符で囲んでください。

パーミッション

sysadmin dump_file には、“sa” パーミッションが必要です。

参照：

- admin log_name (63 ページ)
- sysadmin dump_queue (471 ページ)
- sysadmin sqt_dump_queue (516 ページ)

sysadmin dump_queue

Replication Server のステーブルキューの内容をダンプします。

構文

```
sysadmin dump_queue {, q_number | server[,database]}, qtype
{
, seg, blk, cnt
[, num_cmds]
[, {L0 | L1 | L2 | L3}]
[, {RSSD | client | "log" | file_name}]
|
"next" [, num_cmds]
}
```

パラメータ

- **q_number | server[, database] –** ダンプするステーブルキューを指定します。*q_number* または *server*[, *database*] を使用して、キュー番号を指定します。**admin who**、**admin who, sqm**、**admin who, sqt** を使用すると、キュー番号を確認できます。
- **q_type –** ステーブルキューのキュータイプです。値は、アウトバウンドキューの場合は 0、インバウンドキューの場合は 1 です。キュータイプを確認するには、**admin who**、**admin who, sqm**、**admin who, sqt** を使用します。
- **seg –** 開始セグメントを指定します。
- **blk –** ダンプを開始するセグメントの 16K ブロックを指定します。ブロック番号は 1 ～ 64 です。

  sysadmin dump_queue コマンドでは、*seg* と *blk* に対して、次のように 4 つの特別な設定を行うことができます。

  - *seg* を -1 に設定すると、キューにある最初のアクティブなセグメントから開始されます。
  - *seg* を -2 に設定すると、セーブインターバルを設定することによって保持される非アクティブセグメントを含め、キューにある最初のセグメントから開始されます。
  - *seg* を -1、*blk* を -1 に設定すると、キューにあるまだ削除されていない最初のブロックから開始されます。
  - *seg* を -1、*blk* を -2 に設定すると、キューにあるまだ読み取られていない最初のブロックから開始されます。
- **cnt –** ダンプするブロックの数を指定します。この数は、複数のセグメントにまたがることができます。*cnt* を -1 に設定すると、現在のセグメントの終わりが、

ダンプされる最後のブロックになります。-2 に設定すると、キューの終わりがダンプされる最後のブロックになります。

- **num_cmds –** ダンプするコマンドの数を指定します。この数は、*cnt* よりも優先されます。*num_cmds* を -1 に設定すると、現在のセグメントの終わりがダンプされる最後のコマンドになります。*num_cmds* を -2 に設定すると、キューの終わりがダンプされる最後のコマンドになります。
- **L0 –** ステーブルキューのすべての内容をダンプします。**L0**、**L1**、**L2**、または **L3** が指定されていない場合、これがデフォルトの動作です。
- **L1 –** ステーブル キューにあるトランザクションの **begin** コマンドと **end** コマンドだけをダンプします。
- **L2 –** トランザクションに含まれる他のすべてのコマンドの最初の 100 文字とともに、ステーブルキュートランザクションの **begin** コマンドと **end** コマンドをダンプします。
- **L3 –** ステーブルキューのすべての内容をダンプします。**SQL** 文を除き、他のすべてのコマンドがコメントとして出力されます。**L3** を使用できるのは、*file_name* オプションまたは **sysadmin dump_file** コマンドを使用して代替ログファイルを指定した場合だけです。**L3** は、**RSSD** オプションまたは **client** オプションとともに使用することはできません。
- **RSSD –** RSSD のシステムテーブルが出力先として指定されます。
- **client –** このコマンドを発行するクライアントが出力先として指定されます。
- **"log" –** Replication Server ログファイルが出力先として指定されます。
- **file_name –** *file_name* ログファイルが出力先として指定されます。**sysadmin dump_file** コマンドを使用して、代替ログファイルを設定することもできます。このファイルのロケーションは、Replication Server ログに記録されます。
- **"next"[, num_cmds] –** 特定のキューおよびセッションに対する **sysadmin dump_queue** の前回の実行が中止された場所から開始し、前回実行したときと同数のコマンドまたはブロックをダンプします。*num_cmds* を使用すると、*cnt* または *num_cmds* の以前の値を上書きできます。

  sysadmin dump_queue を事前に呼び出さずに **"next"[, num_cmds]** を使用すると、seg -1、blk -1、cnt -2 をデフォルト値として使用してキューの最初からダンプが開始され、*num_cmds* がコマンドの数として処理されます。

例

- **例 1 –** キュー 103:1 で、セグメント 0 のブロック 15 ～ 64 と、セグメント 1 のブロック 1 ～ 15 を Replication Server ログにダンプします。

```
sysadmin dump_queue, 103, 1, 0, 15, 65
```

- **例 2 –** キュー 103:1 のすべての内容を RSSD にダンプします。

```
sysadmin dump_queue, 103, 1, -1, 1, -2, RSSD
```

- **例 3 –** キュー 103:1 の内容を `SYDNEY_RS.log` ログファイルにダンプします。最後の **sysadmin dump_file** コマンドによって、`SYDNEY_RS.log` がクローズされ、以降のダンプ先は Replication Server ログになります。

```
sysadmin dump_file, SYDNEY_RS.log
sysadmin dump_queue, 103, 1, -1, 1, -2
sysadmin dump_file
```

- **例 4 –** SYDNEY_DS.*pubs2* のインバウンドキューの内容を Replication Server ログにダンプします。

```
sysadmin dump_queue, SYDNEY_DS, pubs2, 1, -1, 1,
-2, 10, "log"
```

- **例 5 –** キュー 103:1 の 10 個のコマンドを Replication Server ログにダンプします。

```
sysadmin dump_queue, 103, 1, -1, 1, -2, 10, "log"
```

- **例 6 –** キュー 103:1 の **begin** コマンドと **end** コマンドだけを Replication Server ログにダンプします。

```
sysadmin dump_queue, 103, 1, -1, 1, -2, L1
```

- **例 7 –** キュー 103:1 の内容を Replication Server ログにダンプします。

```
sysadmin dump_queue, 103, 1, -1, 1, -2, "next"
```

- **例 8 –** キュー 103:1 の内容をチャンク単位で Replication Server ログにダンプします。**"next"** は、**sysadmin dump_queue** の前回の実行が中止された場所からキューをダンプします。この例では、**sysadmin dump_queue** の最初の呼び出しで最初の 10 個のコマンド、2 番目の呼び出しで次の 10 個のコマンド、最後の呼び出しで次の 20 個のコマンドがそれぞれダンプされます。

```
sysadmin dump_queue, 103, 1, -1, 1, -2, 10
sysadmin dump_queue, 103, 1, "next"
sysadmin dump_queue, 103, 1, "next", 20
```

使用法

- **sysadmin dump_queue** は、Replication Server のステーブルキューの内容をダンプするときに使用します。
- **sysadmin dump_queue** は、ステーブルキューを次のいずれかにダンプします。
  - Replication Server ログ
  - 代替ログファイル
  - RSSD
  - コマンドを発行したクライアント

  キューを RSSD またはクライアントにダンプするには、**sysadmin dump_queue** の最後の引数に **RSSD** または **client** を指定する必要があります。

RSSD オプションまたは **client** オプションが指定されていない場合、または "**log**" オプションが指定されている場合は、出力先が Replication Server ログになります。

sysadmindump_file コマンドまたは *file_name* オプションを使用して、キューをダンプする代替ログファイルが指定されている場合、出力先はその代替ダンプファイルになります。

- **queue_dump_buffer_size** 設定パラメータを設定して、**sysadmin dump_queue** コマンドの最大長を指定します。

RSSD へのダンプ

RSSD オプションを使用すると、RSSD の 2 つのシステムテーブル (*rs_queuemsg* と *rs_queuemsgtxt*) にダンプが書き込まれます。

キューが RSSD にダンプされると、システムテーブルではダンプされるブロックと同じ *q_number*、*q_type*、*seg*、*blk* を持つセグメントが最初にクリアされます。

rs_queuemsg システムテーブルの内容については、「Replication Server システムテーブル」を参照してください。

rs_queuemsgtxt システムテーブルには、ステーブルキューからダンプされたコマンドのテキストが保持されます。コマンドのテキストが 255 文字を超える場合は、*q_seq* カラムによって番号が付けられた複数のローに格納されます。

クライアントへのダンプ

client オプションを使用すると、このコマンドを発行したクライアント (**isql** や Replication Server Manager など) にダンプが書き込まれます。

パーミッション

sysadmin dump_queue には、“sa” パーミッションが必要です。

参照：

- admin who (107 ページ)
- rs_queuemsg (813 ページ)
- rs_queuemsgtxt (815 ページ)
- sysadmin dump_file (470 ページ)

sysadmin dump_thread_stacks

Replication Server のスタックをダンプします。

構文

```
sysadmin dump_thread_stacks [, module_name]
```

パラメータ

- **module_name –** Replication Server スレッドのタイプです。有効なモジュール名は、**admin who** コマンドによって表示される *name* カラムの値と同じです。

例

- **例 1 –** RSI キュースタックをダンプします。

```
sysadmin dump_thread_stacks, RSI
```

```
T. 2006/10/23 15:37:39. (259): RS Thread Type  = 'RSI'
T. 2006/10/23 15:37:39. (259): RS Thread State =
'Awaiting Wakeup'
T. 2006/10/23 15:37:39. (259): RS Thread Info  =
'ost_columbia_02'
T. 2006/10/23 15:37:39. (259): Open Server Process ID:
50, SRV_PROC address 0xed79c8
T. 2006/10/23 15:37:39. (259): Start of stack trace for
spid 50.
T. 2006/10/23 15:37:39. (259): Native thread #70,
FramePointer: 0xfe34f050
T. 2006/10/23 15:37:39. (259): 0x00362fc8
sqm_read_message     (0x3345ed0, 0xfe34fdf4, 0xea60,
0x0, 0xfe34fdf0, 0x47105f0) +0x48
T. 2006/10/23 15:37:39. (259): 0x00300908
_rsi_sender_wrapper  (0x30c390, 0x30c230, 0x476f1f0,
0x47105f0, 0x1f2, 0x47105f0) +0x2f28
T. 2006/10/23 15:37:39. (259): 0x002fe960
_rsi_sender_wrapper  (0x1d794f0, 0xffffd8f1,
0x268d14, 0xffffd800, 0x800, 0x0) +0xf80
T. 2006/10/23 15:37:39. (259): 0x0054dabc
srv__start_function  (0xed79c8, 0x0, 0x800,
0x862a04, 0x0, 0x0) +0x1c0
T. 2006/10/23 15:37:39. (259): 0xff265d48 _resume_ret
(0x0, 0x0, 0x0, 0x0, 0x0, 0x0) +0x2d0
T. 2006/10/23 15:37:39. (259): End of stack trace for
spid 50.
T. 2006/10/23 15:37:39. (259):
```

使用法

- **sysadmin dump_thread_stacks** は、Replication Server の処理速度が異常に遅いときに、Replication Server の内部処理をチェックするために使用します。
- **sysadmin dump_thread_stacks** は、次のプラットフォームで使用できます。
  - Sun Solaris
  - HP-UX
  - Linux
  - IBM

「**srv_dbg_stack()**」(『Open Server Server-Library/C リファレンスマニュアル』) を参照してください。

パーミッション

sysadmin dump_thread_stacks には、“sa” パーミッションが必要です。

sysadmin dump_tran

特定のステーブルキュートランザクションの文をログファイルにダンプします。

構文

```
sysadmin dump_tran {{, q_number, | server [,database]},
q_type, lqid
[, num_cmds]
[, {L0 | L1 | L2 | L3}]
[, {RSSD | client | "log" | file_name}] |
"next" [, num_cmds]}
```

パラメータ

- **q_number | server[, database] –** ステーブルキューを指定します。*q_number* または *server*[, *database*] を使用して、キュー番号を指定します。**admin who**、**admin who, sqm**、**admin who, sqt** を使用すると、キュー番号を確認できます。
- **q_type –** ステーブルキューのキュータイプです。値は、アウトバウンドキューの場合は 0、インバウンドキューの場合は 1 です。キュータイプを確認するには、**admin who**、**admin who, sqm**、**admin who, sqt** を使用します。
- **lqid –** ステーブルキュートランザクションのコマンドのローカルキュー ID です。*lqid* によって、ダンプするトランザクションが識別されます。フォーマット：*seg,blk,row*
- **num_cmds –** ダンプするコマンドの数を指定します。

- **L0 –** 指定したトランザクションの内容をダンプします。**L0**、**L1**、**L2**、または **L3** が指定されていない場合、これがデフォルトの動作です。
- **L1 –** 指定したトランザクションの **begin** コマンドと **end** コマンドだけをダンプします。
- **L2 –** トランザクションに含まれる他のコマンドの最初の 100 文字とともに、指定したトランザクションの **begin** コマンドと **end** コマンドをダンプします。
- **L3 –** 指定したトランザクションのすべてのコマンドをダンプします。**SQL** 文を除き、他のすべてのコマンドがコメントとして出力されます。**L3** を使用できるのは、*file_name* オプションまたは **sysadmin dump_file** コマンドを使用して代替ログファイルを指定した場合だけです。**L3** は、**RSSD** オプションまたは **client** オプションとともに使用することはできません。
- **RSSD –** RSSD のシステムテーブルが出力先として指定されます。
- **client –** このコマンドを発行したクライアントが出力先として指定されます。
- **"log" –** Replication Server ログファイルが出力先として指定されます。
- **file_name –** *file_name* ログファイルが出力先として指定されます。**sysadmin dump_file** コマンドを使用して、代替ログファイルを設定できます。
- **"next"[, num_cmds] –** このオプションは、**sysadmin dump_tran** の前回の実行を続行します。**"next"[, num_cmds]** は、特定のトランザクションに対する **sysadmin dump_tran** の前回の実行が中止された場所から開始し、前回実行したときと同数のコマンドをダンプします。*num_cmds* を使用すると、*cnt* または *num_cmds* の以前の値を上書きできます。

  "next"[, num_cmds] は **sysadmin dump_tran** を事前に呼び出しておかないと使用できません。

例

- **例 1 –** キュー 103:1 の LQID が 0:15:2 であるトランザクションを Replication Server ログにダンプします。

```
sysadmin dump_tran, 103, 1, 0, 15, 2
```

- **例 2 –** *SYDNEY_DS.pubs2* のインバウンドキューの LQID が 0:15:2 であるトランザクションの 10 個のコマンドを Replication Server ログにダンプします。

```
sysadmin dump_tran, SYDNEY_DS, pubs2, 1, 0, 15, 2,
10, “log”
```

- **例 3 –** キュー 103:1 の LQID が 0:15:2 であるトランザクションの **begin** コマンドと **end** コマンドだけを Replication Server ログにダンプします。

```
sysadmin dump_tran, 103,1, 0, 15, 2, L1
```

- **例 4 –** キュー 103:1 の LQID が 0:15:2 であるトランザクションのすべてのコマンドを Replication Server ログにダンプします。コマンドはすべて 100 文字にトランケートされます。

```
sysadmin dump_tran, 103,1, 0, 15, 2, L2
```

- **例 5 –** キュー 103:1 の LQID が 0:15:2 であるトランザクションを `SYDNEY_RS.log` ファイルにダンプします。

```
sysadmin dump_tran, 103,1, 0, 15, 2, L3, SYDNEY_RS.log
```

- **例 6 –** キュー 103:1 の LQID が 0:15:2 であるトランザクションを RSSD にダンプします。

```
sysadmin dump_tran, 103, 1, 0, 15, 2, RSSD
```

- **例 7 –** キュー 103:1 の LQID が 0:15:2 であるトランザクションをクライアントにダンプします。

```
sysadmin dump_tran, 103, 1, 0, 15, 2, client
```

- **例 8 –** キュー 103:1 の LQID が 0:15:2 であるトランザクションをチャンク単位で Replication Server ログにダンプします。**"next"** は、**sysadmin dump_tran** の前回の実行が中止された場所からトランザクションをダンプします。この例では、**sysadmin dump_tran** の最初の呼び出しでトランザクションの最初の 10 個のコマンド、2 番目の呼び出しでトランザクションの次の 10 個のコマンド、最後の呼び出しでトランザクションの次の 20 個のコマンドがそれぞれダンプされます。

```
sysadmin dump_tran, 103,1, 0, 15, 2, 10
sysadmin dump_tran, “next”
sysadmin dump_tran, “next”, 20
```

使用法

- **sysadmin dump_tran** は、LQID で識別されたステーブルキュートランザクションの内容をダンプするときに使用します。
- **sysadmin dump_tran** の出力は、次のいずれかに書き込まれます。
  - Replication Server ログ
  - 代替ログファイル
  - RSSD
  - コマンドを発行したクライアント

  ステーブルキュートランザクションを RSSD またはクライアントにダンプするには、**sysadmin dump_tran** の最後の引数に **RSSD** または **client** を指定する必要があります。

  RSSD オプションまたは **client** オプションが指定されていない場合、または **log** オプションが指定されている場合は、出力先が Replication Server ログになります。

  sysadmindump_file コマンドまたは *file_name* オプションを使用して、ステーブルキュートランザクションをダンプする代替ログファイルが指定されている場合、出力先はその代替ダンプファイルになります。

- **queue_dump_buffer_size** 設定パラメータを設定して、**sysadmin dump_tran** コマンドの最大長を指定します。

RSSD へのダンプ

RSSD オプションを使用すると、RSSD の 2 つのシステムテーブル (*rs_queuemsg* と *rs_queuemsgtxt*) にダンプが書き込まれます。

トランザクションを RSSD にダンプすると、システムテーブルでは、ダンプされるトランザクションと同じ *q_number*、*q_type*、*seg*、*blk* が指定されたセグメントが最初にクリアされます。

rs_queuemsg システムテーブルの内容については、「Replication Server システムテーブル」を参照してください。

rs_queuemsgtxt システムテーブルには、ステーブルキューからダンプされたコマンドのテキストが保持されます。コマンドのテキストが 255 文字を超える場合は、*q_seq* カラムによって番号が付けられた複数のローに格納されます。

クライアントへのダンプ

client オプションを使用すると、このコマンドを発行したクライアント (**isql** や Replication Server Manager など) にダンプが書き込まれます。

パーミッション

sysadmin dump_tran には、“sa” パーミッションが必要です。

参照：

- admin who (107 ページ)
- rs_queuemsg (813 ページ)
- rs_queuemsgtxt (815 ページ)
- sysadmin dump_file (470 ページ)

sysadmin erssd

ERSSD ファイルのロケーションを確認して設定をバックアップしたり、ERSSD のスケジュールされていないバックアップを実行したりできるようにします。

このコマンドが返す ERSSD のステータスは次のとおりです。

- ERSSD 名
- データベースファイルのロケーション
- トランザクションログファイルのロケーション

- トランザクションミラーのロケーション
- バックアップの開始時刻、開始日、間隔
- バックアップディレクトリのロケーション

構文

```
sysadmin erssd [, backup | dbfile_dir, 'path' | translog_dir, 'path'
|
logmirror_dir, 'path' | defrag]
```

パラメータ

- **backup –** ERSSD のスケジュールされていない単一のバックアップを実行します。
- **dbfile_dir, 'path' –** ERSSD データベースファイルの新しいディレクトリを指定します。
- **translog_dir, 'path' –** トランザクションログファイルの新しいディレクトリを指定します。
- **logmirror_dir, 'path' –** トランザクションログミラーファイルの新しいディレクトリを指定します。
- **defrag –** 空のフラグメントなしで ERSSD データベースを再構築します。
- **path –** 新しいディレクトリのパス名です。

注意： これらのディレクトリパスの変更オプションは慎重に使用してください。これらのオプションを使用して **sysadmin erssd** を実行すると、ERSSD が自動的にリブートされ、システムが中断されることがあります。

例

- **例 1 –** 次は、**sysadmin erssd** の出力例です。

```
sysadmin erssd
---------------------

ERSSD NameERSSD Database FileERSSD Transaction Log
-----------------------------------------------
erssd.db/dbfile/erssd.db/log/erssd.log


ERSSD Transaction Log MirrorERSSD Backup Start Time
---------------------------------------------------
/backup/erssd.mlg2am

ERSSD Backup Start DateERSSD Backup Interval
--------------------------------------------
March 20, 200312 hours

ERSSD Backup Location
```

```
--------------------
/backup
```

使用法

- オプションを指定しないでこのコマンドを使用すると、データベースファイルのパス、トランザクションログのパス、トランザクションログミラーのパス、スケジュールされたトランザクションの開始時刻、開始日、ロケーションが表示されます。
- **backup** オプションを指定してこのコマンドを使用すると、スケジュールされていない単一のバックアップが実行されます。
- **dbfile_dir** オプションを指定してこのコマンドを使用すると、ERSSD が停止され、データベースが新しいディレクトリに移動されます。さらに、Replication Server 設定ファイルが更新され、新しいロケーションからデータベースを使用して ERSSD が再起動されます。
- **translog_dir** オプションを指定してこのコマンドを使用すると、ERSSD の停止、トランザクションログファイルの新しいディレクトリへの移動、新しいディレクトリでトランザクションログミラーを使用するための ERSSD の更新、Replication Server 設定ファイルの更新、ERSSD の再起動が実行されます。
- **logmirror_dir** オプションを指定してこのコマンドを使用すると、ERSSD が停止され、トランザクションログミラーファイルが新しいディレクトリに移動されます。さらに、その新しいディレクトリのトランザクションログミラーを使用するために ERSSD が更新され、Replication Server 設定ファイルが更新されて、ERSSD が再起動されます。
- **defrag** オプションを指定してこのコマンドを使用すると、ERSSD が停止され、データベースファイルが再構築されて、ERSSD が再起動されます。
- **defrag**、**dbfile_dir**、**translog_dir**、**logmirror_dir** の各オプションを指定してこのコマンドを使用すると負荷がかかります。このオペレーションの間、ERSSD を使用できないため、ERSSD にアクセスしようとするスレッドはすべて失敗します。これらのスレッドは、ERSSD が再起動されるまでブロックされたままになります。
- **defrag** を使用するには、サイトバージョンが 15.0 以上である必要があります。デフラグしたファイルは、このオプションによって SQL Anywhere 11.0 に自動的にアップグレードされます。コマンドの実行後にダウングレードすることはできません。
- このコマンドは、容量の大きい高速ディスクにファイルを移動する必要があるときに使用します。
- *path* には、二重引用符ではなく一重引用符を使用します。

パーミッション

このコマンドを実行するには、"sa" 権限が必要です。

sysadmin fast_route_upgrade

プライマリ Replication Server とレプリケート Replication Server のサイトバージョンのうち、いずれか低い方にルートバージョンを更新します。

ルートをアップグレードすると、システムテーブル内のデータが再マテリアライズされるため、新しくアップグレードした Replication Server の新機能についての情報を利用できるようになります。

注意： プライマリ Replication Server でマテリアライゼーションに必要な新機能が使用されていない場合にのみ、**sysadmin fast_route_upgrade** を使用してください。

構文

```
sysadmin fast_route_upgrade, dest_replication_server
```

パラメータ

- **dest_replication_server –** ルートの送信先 Replication Server です。

例

- **例 1 –** これらの例では、TOKYO_RS のサイトバージョンは 1200 です。SYDNEY_RS は 11.5 から 12.0 にアップグレードされたばかりであり、サイトバージョンは 1200 です。東京の Replication Server (TOKYO_RS) で終了するルートの送信元 Replication Server (SYDNEY_RS) でこのコマンドが発行されると、ルートのバージョンが 12.0 に設定されます。SYDNEY_RS では、新機能はまだ使用されていません。

  ```
  sysadmin fast_route_upgrade, TOKYO_RS
  ```

使用法

- ルートの両端にある Replication Server がアップグレードされ、サイトバージョンが 11.5 以降に設定されているときには、2 つのサーバを接続する各ルートを必ずアップグレードして、そのルート経由で新機能を使用できるようにしてください。このコマンドを送信元 Replication Server で発行して、ルートバージョンを更新します。
- 新しい機能が送信元 Replication Server で使用されていない場合は、**sysadmin fast_route_upgrade** を使用してルートをアップグレードしてください。
- 送信元 Replication Server で新機能をすでに使用している場合、このコマンドは拒否されます。

  ```
  sysadmin upgrade, “route”, dest_replication_server
  ```

を使用して、ルートをアップグレードしてください。

パーミッション

sysadmin fast_route_upgrade には、“sa” パーミッションが必要です。

参照：

- admin show_route_versions (82 ページ)
- admin show_site_version (83 ページ)
- sysadmin site_version (502 ページ)

sysadmin hibernate_off

Replication Server のハイバネーションモードをオフにして、アクティブステータスに戻します。

構文

```
sysadmin hibernate_off [, string_ID]
```

パラメータ

- **string_ID –** 有効な識別子です。**sysadmin hibernate_on** で *string_ID* を指定した場合は、**sysadmin hibernate_on** に対して使用したのと同じ識別子を指定してください。

  string_ID を忘れた場合は、*rs_recovery* システムテーブルの *text* カラムで検索できます。

  ルートアップグレードまたはルートアップグレードリカバリが成功した後で、レプリケート Replication Server のハイバネーションモードをオフにする必要がある場合は、*string_ID* にその Replication Server 名を使用してください。

例

- **例 1 –** このコマンドは、Replication Server (TOKYO_RS) のハイバネーションモードをオフにします。

  ```
  sysadmin hibernate_off, TOKYO_RS
  ```

使用法

- ハイバネーションモードとは、Replication Server の次のような状態のことです。
  - すべてのデータ定義言語 (DDL) コマンドが拒否される。

    - データサーバインタフェース (DSI)、ディストリビュータ、Replication Server インタフェース (RSI) 送信側スレッドなど、ほとんどのサービススレッドがサスペンドされる。
    - すべてのルートとコネクションがサスペンドされる。
    - RSI ユーザがログアウトされ、Replication Server に再びログインできない。
- ハイバネーションモードの間は、システム情報タイプのコマンド (**admin**) とシステム管理タイプのコマンド (**sysadmin**) を実行できます。
- ハイバネーションモードをオフにする Replication Server で、このコマンドを実行してください。
- ルートアップグレードが失敗したときに、送信先 Replication Server がハイバネーションモードになることがあります。 Replication Server を再度アクティブにするために、**sysadmin hibernate_off** を使用しないでください。 代わりに、以下を使用してルートアップグレードをリカバリします。
  ```
  sysadmin upgrade, "route", dest_replication_server, "recovery"
  ```
- 場合によっては、ルートアップグレードが成功した後に、送信先 Replication Server がハイバネーションモードになることがあります。**sysadmin hibernate_off** を使用して、送信先 Replication Server を再度アクティブにしてください。

パーミッション

sysadmin hibernate_off には、"sa" パーミッションが必要です。

参照：

- sysadmin hibernate_on (484 ページ)

sysadmin hibernate_on

Replication Server のハイバネーションモードをオンにします (Replication Server をサスペンドします)。

警告！ sysadmin hibernate_on コマンドを使用すると、Replication Server へのルートが存在する場合に、ロス検出になる場合があります。

構文

```
sysadmin hibernate_on [, string_ID]
```

パラメータ

- **string_ID –** 有効な識別子です。**sysadmin hibernate_off** を実行したときと同じ *string_ID* を指定してください。*string_ID* を使用することにより、Replication

Server で作業している間に、他のユーザが誤って Replication Server のハイバネーションモードをオフにすることを防止できます。

string_ID を忘れた場合は、*rs_recovery* システムテーブルの *text* カラムで検索できます。

例

- **例 1 –** このコマンドは、Replication Server (TOKYO_RS) のハイバネーションモードをオンにします。

```
sysadmin hibernate_on, TOKYO_RS
```

使用法

- ハイバネーションモードとは、Replication Server の次のような状態のことです。
  - すべてのデータ定義言語 (DDL) コマンドが拒否される。
  - データサーバインタフェース (DSI)、ディストリビュータ、Replication Server インタフェース (RSI) 送信側スレッドなど、ほとんどのサービススレッドがサスペンドされる。
  - すべてのルートとコネクションがサスペンドされる。
  - RSI ユーザがログアウトされ、Replication Server に再びログインできない。
- ハイバネーションモードの間は、システム情報タイプのコマンド (**admin**) とシステム管理タイプのコマンド (**sysadmin**) を実行できます。
- ハイバネーションモードをオンにする Replication Server で、このコマンドを実行してください。
- Replication Server のハイバネーションモードをオンにすると、問題をデバッグするのに役立ちます。

パーミッション

sysadmin hibernate_on には、“sa” パーミッションが必要です。

参照：

- sysadmin hibernate_off (483 ページ)

sysadmin issue_ticket

インバウンドキューまたはアウトバウンドキューに **rs_ticket** マーカを挿入します。

構文

```
sysadmin issue_ticket {,q_number} |{,ds_name, db_name},[,q_type], h1
[, h2 [, h3 [, h4]]] [,v]
```

パラメータ

- **ds_name –** データベースが常駐するデータサーバの名前。
- **db_name –** データベースの名前。
- **q_number –** ステーブルキューを指定します。
- **q_type –** ステーブルキューのタイプを指定します。 値は、アウトバウンドキューの場合は 0、インバウンドキューの場合は 1 です。q_type はオプションで、指定しない場合は、インバウンドキューがデフォルトキュータイプになります。
- **h1、h2、h3 –** 各パラメータには、1 ～ 10 文字が含まれます。これらのパラメータは、ユーザにとって適切と思われる方法で識別子として使用されるため、データベース識別子の命名規則に従う必要があります。ヘッダパラメータは、最初に数値を使用することができず、予約語とすることもできません。ヘッダパラメータを数値にする場合は、引用符で囲む必要があります。たとえば、'1' などとします。
- **h4 –** 1 ～ 50 文字が含まれます。h1、h2、および h3 と同様に、このパラメータは、ユーザにとって適切と思われる方法で識別子として使用できます。
- **v – rs_ticket** のバージョン番号を示します。1 または 2 とする必要があります。指定されていない場合、デフォルト値は 2 になります。

例

- **例 1 –** このコマンドは、Replication Server のインバウンドキューに 1 つのトランザクションを挿入します。

  ```
  sysadmin issue_ticket, 103, 'start'
  go
  ```

  構文の説明は次のとおりです。

  103 は Replication Server への論理コネクションの *q_number* です。

- **例 2 –** この例は、Replication Server のアウトバウンドキューにトランザクションを挿入します。

```
sysadmin issue_ticket,103,0,'t6'
go
```

構文の説明は次のとおりです。

103 は q_number、*0* はアウトバウンドキューのタイプ、*t6* は、論理 Replication Server の h1 ヘッダです。

使用法

sysadmin issue_ticket コマンドを使用する場合の注意点は、次のとおりです。

- Replication Server で複写データベースからのサブスクリプションを 1 つ以上持つ必要があります。サブスクリプションがない場合、ディストリビュータ (DIST) モジュールから対応するデータサーバインタフェース (DSI) に **rs_ticket** マーカが送信されません。
- プライマリデータベース (PDB) と EXEC モジュールのタイムスタンプは、挿入された **rs_ticket** マーカ内の任意の値です。
- ステーブルキューは、*q_number*、*q_type* または *ds_name*、*db_name*、および *q_type* を使用することでのみ指定できます。 ウォームスタンバイ環境では、インバウンドキューは論理接続に関連し、Replication Server にはスタンバイデータベースのインバウンドキューがありません。 ウォームスタンバイに **sysadmin issue_ticket** を使用する場合の注意点は、次のとおりです。
  - ユーザがアクティブデータベースに対する既存の論理コネクションまたは物理コネクションによりステーブルキューを指定した場合、Replication Server のインバウンドキューに特定の **rs_ticket** マーカが書き込まれます。対応する **rs_ticket** レコードは、プライマリサイトのレプリケートデータベースとスタンバイデータベースで確認できます。

    注意： 2 つの Replication Server (RS) DR のセットアップでは、プライマリサーバが停止した場合、プライマリ Adaptive Server Enterprise (ASE) と RS の両方を停止します。この場合は、クライアントを DR ASE にフェールオーバする前に、アウトバウンドキューがクリアされていることを確認してください。このためには、アウトバウンドキューにチケットを挿入する必要があります。rs_ticket_history テーブルにチケットが検出された場合は、クライアントがフェールオーバできます。
  - ユーザがスタンバイデータベースに対する既存の物理コネクションによりステーブルキューを指定した場合、このようなインバウンドキューが存在しないことを示すエラーメッセージが表示されます。

sysadmin ldap

LDAP URL を設定またはリストし、LDAP ユーザ認証用のアクセスアカウントを指定するか、LDAP URL またはログイン関連パラメータを確認します。

構文

```
sysadmin ldap [operation [,parameter1, [,parameter2]]]
```

有効な *operation* [,*parameter1*, [,*parameter2*]] オプションは次のとおりです。

```
set_primary_url, 'ldapurl'
set_access_acct, 'account_distinguished_name', 'account_password'
list_urls
list_access_acct
check_url, 'ldapurl' [,'tls'] [,'dn', 'pwd']
check_login, 'login_name'
set_secondary_url, 'ldapurl'
set_secondary_access_acct,'account_distinguished_name',account_password
starttls_on_primary, 'true|false'
starttls_on_secondary, 'true|false'
set_timeout, timeout_in_milliseconds
set_retry_limit, retry_number
set_cacert_file, 'full/path/to/CARootCertFile'
refresh_ldapua_module
```

パラメータ

- **set_primary_url, '*ldapurl*' –** プライマリ LDAP URL 検索フィルタを指定します。*ldapurl* の構文は次のとおりです。

  ```
  ldapurl:=ldap://host:port/node?attributes?base | one |sub?filter
  ```

  構文の説明は次のとおりです。

  - **host –** LDAP サーバのホスト名です。
  - **port –** LDAP サーバのポート番号です。
  - **node –** 検索を開始するオブジェクト階層内のノードを指定します。
  - **attributes –** 結果セットで返される属性のリストです。 サポートされる属性リストは LDAP サーバごとに異なることができます。
  - **base –** ベースノードの検索を指定し、検索基準を修飾します。
  - **one –** node で指定したノードとその 1 つ下のレベルのノードの検索を指定し、検索基準を修飾します。
  - **sub –** node で指定したノードとその下位レベルのすべてのノードの検索を指定します。
  - **filter –** 認証する属性 (複数も可) を指定します。 filter は "uid=*" のように簡潔にすることも、"&(uid=*)(ou=group)" のように複雑にすることもできま

す。 ログイン名の標準属性は OpenLDAP では "uid" で、Microsoft Active Directory では "samaccountname" です。

注意： URL 構文は LDAP サーバに固有で、ログイン名の記述にワイルドカード (*) を使用します。

- **set_access_acct, '*account_distinguished_name*', '*account_password*' –** Replication Server が検索および administrative function の実行に使用する LDAP サーバのユーザアカウントの識別名 (DN) とパスワードを指定します。

  管理 DN とパスワードが指定されていない場合、Replication Server はユーザアカウントの検索に LDAP サーバへの匿名バインドを使用します。
- **list_urls –** LDAP URL 検索フィルタを表示します。
- **list_access_acct –** "**set_access_account**" パラメータで設定されている LDAP サーバのアクセスアカウント DN を表示します。
- **check_url, '*ldapurl*' [,'*tls*'] [, '*dn*', '*pwd*'] –** LDAP URL 検索フィルタを確認します。 LDAP サーバへの接続が実行されていることを確認します。
- **check_login, '*login_name*' –** LDAP サーバにユーザアカウントがあるかどうかを確認しますが、ユーザの認証は行いません。
- **set_secondary_url, '*ldapurl*' –** セカンダリ LDAP URL 検索フィルタを指定します。

注意： URL 文字列が NULL であるか、プライマリ LDAP URL への接続に失敗した場合、Replication Server はセカンダリ LDAP URL へのフェールオーバを試みます (指定されている場合)。 LDAP の検索操作で障害が発生した場合、Replication Server はセカンダリ URL にフェールオーバしません。

- **set_secondary_access_acct, 'account_distinguished_name', 'account_password' –** セカンダリ DN、および Replication Server が検索および administrative function の実行に使用する LDAP サーバのユーザアカウントのパスワードを指定します。

  検索フィルタが指定されていない場合、Replication Server はユーザアカウントの検索に LDAP サーバへの匿名バインドを使用します。
- **starttls_on_primary, 'true|false' –** プライマリ LDAP サーバで TLS 接続を開始するか停止するかを指定します。
- **starttls_on_secondary, 'true|false' –** セカンダリ LDAP サーバで TLS 接続を開始するか停止するかを指定します。
- **set_timeout –** Replication Server が要求を却下するまで LDAP サーバからの応答を待機するタイムアウト値をミリ秒単位で指定します。 **set_timeout** のデフォルト値は 10,000 ミリ秒 (10 秒) です。 有効な範囲は、1 から 3,600,000 (1 時間) です。
- **retry_limit –** 一時的なエラーの後の再試行回数の限度を指定します。 デフォルト値は 3 です。 **retry_limit** の有効な範囲は 1 から 60 です。

- **set_cacert_file, 'full/path/to/CARootCertFile' –** セキュアソケットレイヤ (SSL) 通信用の PEM 形式ファイルを受け入れる、認証された証明書 (CA) のルートファイルへのフルパスを設定します。 たとえば、ファイルのデフォルトの場所は '$SYBASE/config/trusted.txt' です。

  『Replication Server 管理ガイド 第 1 巻』>「Replication Server のセキュリティ管理」>「SSL セキュリティの管理」>「SSL の概要」を参照してください。
- **refresh_ldapua_module –** LDAP ユーザ認証モジュール全体を再初期化します。

  再初期化を有効にするのに Replication Server を再起動しないでください。 このパラメータは、LDAP ユーザ認証モジュールで保持されていたリソースを解放するか、CA ルートファイルの内容の変更など、Replication Server の外部でファイルに加えられた変更を再読み取りします。

例

- **例 1 –** サブレベルの基準を使用して、Replication Server で LDAP URL 検索フィルタを設定します。

```
sysadmin ldap, set_primary_url,
'ldap://myhost:389/dc=mycompany,dc=com?distinguishedName?sub?
uid=*?'
```

- **例 2 –** 認証用に LDAP サーバのログイン名とパスワードを指定します。

```
sysadmin ldap
set_access_acct, 'cn=Manager, dc=mycompany, dc=com', 'password'
```

- **例 3 –** LDAP サーバ接続を確認します。

```
sysadmin ldap, check_url, 'ldap://myhost:389'
```

```
sysadmin ldap, check_url,
'ldap://myhost;389', 'cn=Manager,dc=mycompany,dc=com', 'password'
```

```
sysadmin ldap, check_url, 'ldaps://myhost:636'
```

```
sysadmin ldap, check_url, 'ldap://myhost:389', 'tls'
```

- **例 4 –** プライマリ LDAP サーバで TLS 接続を開始します。

```
sysadmin ldap, starttls_on_primary, 'true'
```

- **例 5 –** SSL 接続用にターゲット SLAP サーバの CA ルートパスファイルを設定します。

```
sysadmin ldap, set_cacert_file, 'user/sybase/config/trusted.txt'
```

- **例 6 –** Replication Server が要求を却下するまで待機するタイムアウト値をミリ秒で設定します。

```
sysadmin ldap, set_timeout, 3000
```

- **例 7 –** 一時的なエラーの後の再試行回数を設定します。

```
sysadmin ldap, set_retry_limit, 6
```

使用法

- LDAP ベンダが検索フィルタの構文を決めています。 どの場合でも、検索フィルタはユーザをユニークに特定する属性名を指定します。形式は “*属性*=ワイルドカード“ で、たとえば “cn=*” のようになります。
- 複合フィルタのワイルドカードを持つ最初の属性は、相対識別名を定義する必要があり、それ以外の場合、認証は失敗します。 たとえば、“uid = ray, dc=sybase, dc=com“ がユーザ DN の場合、親 DN は “dc=sybase, dc=com“ で、相対識別名は “uid = ray“ です。
- 検索フィルタが追加されると、Replication Server は有効な LDAP URL 構文が使用され、実在のノードを参照していることを確認します。 有効な文字列が期待値を戻すには、検索フィルタを注意深く選択し、Replication Server の設定時に確認する必要があります。
- 次のいずれかによって、Replication Server で LDAP ユーザ認証に SSL または TLS を使用できます。
  - CA ルートファイルパスを設定し、"`ldaps://`" スキーマを入力して LDAP URL を指定する。
  - ターゲット LDAP URL で **sysamdin ldap** を使用して TLS を有効にする。 LDAP URL スキーマは "s" なしで "`ldap://`" にする必要があります。

パーミッション

sysadmin ldap には、"sa" パーミッションが必要です。

sysadmin lmconfig

Replication Server でライセンス管理に関連する情報を設定して表示します。

構文

```
sysadmin lmconfig,
[ , edition [, edition_type ]
  , license_type [, license_type_name ]
  , smtp_host [, smtp_host_name ]
  , smtp_port [, smtp_port_number ]
  , email_sender [, sender_email_address ]
  , email_recipients [, email_recipients ]
  , email_severity [, email_severity ]]
```

パラメータ

- **sysadmin lmconfig –** パラメータを指定しない場合は、基本的なライセンスステータス情報が表示されます。
- **edition, `edition_type` –** ライセンスエディションを指定する静的設定パラメータです。

  `edition_type` の値は次のとおりです。

  - `null` - デフォルト値。 null 値を指定した場合は、製品のエディションが設定されず、Replication Server はどのエディションのライセンスでも起動します。
  - `EE` - Enterprise Edition を示します。
  - `RL` - Real-Time Loading (RTL) エディションを示します。
- **license type, `license_type_name` –** Replication Server のインストール環境のライセンスの種類を示す静的設定パラメータで、null 以外のエディションを指定した場合にのみ有効です。

  `license_type_name` の有効な代表値は次のとおりです。

  - `SR` - サーバライセンス
  - `SV` - スタンバイサーバライセンス
  - `AR` - アプリケーションサーバライセンス
  - `BR` - アプリケーション固有のスタンバイサーバライセンス
  - `IC` - インターネットアクセス CPU ライセンス
  - `AC` - アプリケーション固有の CPU ライセンス
  - `BC` - アプリケーション固有のスタンバイ CPU ライセンス
  - `CP` - CPU ライセンス
  - `SF` - スタンバイ CPU ライセンス
  - `null` - デフォルト

  注意： **sysadmin lmconfig** は、このリストに加えて、特別なレガシーライセンスの 2 文字の省略形も受け入れます。このライセンスの種類が受け入れられない場合は、種類を null に設定し、ネットワークライセンスサーバオプションファイルを使用して、この Replication Server が使用するライセンスを制御します。
- **smtp host, smtp host name –** ライセンスイベント通知用の電子メールメッセージの送信に使用する SMTP ホストを指定します。
- **smtp port, smtp port number –** ライセンスイベント通知用の電子メールメッセージの送信に使用する SMTP ポートを指定します。

- **email sender, sender email address –** ライセンスイベントの電子メール通知の送信者のアドレスとして使用する電子メールアドレスを指定します。
- **email recipients, email recipients –** ライセンスイベントの電子メール通知を受け取る電子メール受信者のカンマ区切りのリストです。
- **email severity, email severity –** どの重大度以上のエラーについて電子メール通知を送信するかを指定します。デフォルトはエラーで、その他に警告と情報を指定できます。

例

- **例 1 –** システムの基本的なライセンス設定情報を表示します。

```
sysadmin lmconfig
go
```

結果は次のようになります。

```
Parameter Name     Config Value
-------------------------  ------------
edition                          null
license_type                     null
smtp_host                        smtp
email_recipients                cshi
email_severity                  ERROR
smtp_port                          25
email_sender                     cshi
License Name     Version     Quantity Status         Expiry Date
------------- ---------    -------- ------ --------------
REP_SERVER       2005.0425      1        graced     Sep 11 2005
2:40AM
REP_HVAR_ASE     2005.0425      1        graced     Sep 11 2005
2:40AM
Property Name     Property Value
-------------         --------------
PE                       null
LT                       null
```

使用法

- エディションを指定しない場合や “null” を使用した場合は、Replication Server が起動時にライセンスエディションを検索し、検出されたエディションを使用します。
- **sysadmin lmconfig** で設定する設定オプションは、`sylapi` プロパティファイルに格納されます。

パーミッション

sysadmin lmconfig には “sa” パーミッションが必要です。

sysadmin log_first_tran

DSI キューの最初のトランザクションを例外ログに書き込みます。

構文

```
sysadmin log_first_tran, [n], data_server, database
```

パラメータ

- **n** – データベースの例外ログと、Replication Server ログまたは **sysadmin dump_file** コマンドで指定した代替ログファイルに書き込むトランザクションの数を指定します。
- **data_server** – データベースがあるデータサーバの名前です。
- **database** – 最初のトランザクションが書き込まれる DSI キューがあるデータベースの名前です。

例

- **例 1** – この DSI キューの最初のトランザクションを例外ログに書き込みます。

  ```
  sysadmin log_first_tran, SYDNEY_DS, pubs2
  ```

- **例 2** – DSI キューの最初の 5 つのトランザクションを、データベースの例外ログと、Replication Server ログまたは **sysadmin dump_file** コマンドで指定したロケーションに書き込みます。

  ```
  sysadmin log_first_tran, 5, SYDNEY_DS, pubs2
  ```

- **例 3** – DSI キューの最初の 2 つのトランザクションを、データベースの例外ログと `SYDNEY_RS.log` ファイルに書き込みます。最後の **sysadmin dump_file** コマンドによって、`SYDNEY_RS.log` ファイルがクローズされます。

  ```
  sysadmin dump_file SYDNEY_RS.log
  sysadmin log_first_tran, 2, SYDNEY_DS, pubs2
  sysadmin dump_file
  ```

使用法

- **sysadmin log_first_tran** は、DSI キューの最初の *n* 個のトランザクションを、例外ログと Replication Server ログまたは **sysadmin dump_file** コマンドで指定した代替ログファイルに書き込むときに使用します。
- このコマンドは、キューから最初の *n* 個のトランザクションを削除するわけではありません。

- 例外ログは、*rs_exceptshdr*、*rs_exceptscmd*、*rs_systext* の 3 つのテーブルで構成されます。これらのテーブルの詳細については、「Replication Server システムテーブル」を参照してください。

パーミッション

sysadmin log_first_tran には、“sa” パーミッションが必要です。

参照：
- admin who (107 ページ)

sysadmin principal_users[,reload]

`rs_principal_users.cfg` 設定ファイルに保存されているすべての Replication Servers のプリンシパル名をリロードまたは表示します。

構文

```
sysadmin principal_users[,reload]
```

例

- **例 1 –** `rs_principal_users.cfg` 設定ファイルに保存されているすべての Replication Servers のプリンシパル名を表示します。

  ```
  sysadmin principal_users
  go
  ```

  結果は次のようになります。

  ```
  Server Name                  Principal Name
  ---------------------------  ---------------------------
  RRS2                         RRS2_princ
  PRS2                         PRS2_princ
  ```

- **例 2 –** 変更され `rs_principal_users.cfg` 設定ファイルに保存されているプリンシパル名をリロードします。

  ```
  sysadmin principal_users, reload
  go
  ```

  PRS1 のプリンシプル名を *PRS1_princ* に変更した場合。 *PRS2* を *PRS1* に接続する場合、*PRS2* で **sysadmin principal_users, reload** を実行し、再起動せずに *PRS1_princ* をリロードします。

  『Replication Server 管理ガイド第 1 巻』の 「rs_principal_users.cfg ファイルの設定」を参照してください。

パーミッション

このコマンドは、すべてのユーザが実行できます。

sysadmin purge_all_open

Replication Server のインバウンドキューから、すべてのオープントランザクションをパージします。

構文

```
sysadmin purge_all_open, q_number, q_type
```

パラメータ

- **q_number, q_type –** パージするステーブルキューを指定します。ステーブルキューの値は、**admin who**、**admin who, sqm**、**admin who, sqt** を使用して調べることができます。

例

- **例 1 –** キュー 103:1 からすべてのオープントランザクションをパージします。

  ```
  sysadmin purge_all_open, 103, 1
  ```

使用法

- **sysadmin purge_all_open** は、Replication Server のインバウンドキューからすべてのオープントランザクションをパージするときに使用します。オープントランザクションは、インバウンドキューからのみパージできます。

  注意： RepAgent がトランザクション開始レコードと、場合によってはトランザクション内の一部のコマンドをすでに転送していても、トランザクションのコミットレコードまたはアボートレコードがまだ転送されていない場合、そのトランザクションはオープンです。

- **sysadmin purge_all_open** は、データサーバログが Replication Server に完全に転送される前に、Replication Server のインバウンドキューにオープントランザクションを残して、データサーバログをトランケートする必要がある場合に便利です。残ったオープントランザクションは、**sysadmin purge_all_open** を使用して明示的に削除する必要があります。

  警告！ インバウンドキューにオープントランザクションがあり、RepAgent がログからコミットレコードまたはアボートレコードを転送しないことが確実である場合にのみ、**sysadmin purge_all_open** を使用してください。

- ステーブルキューをパージするには、Replication Server に十分な記憶領域が必要です。十分な領域がない場合は、次のエラーメッセージが表示されます。

```
This RS is out of Disk Space. Use another session to
add disk space for this command to proceed.
```

  このメッセージが表示されたら、別の **isql** セッションを開始して、Replication Server にステーブル領域を追加してください。十分な領域が使用可能になるまで、**sysadmin purge_all_open** は実行できません。
- 削除するトランザクションの内容を調べるには、このコマンドを使用する前に **sysadmin sqt_dump_queue** を実行してください。
- キューにオープントランザクションがない場合、このコマンドはキューを変更しません。トランザクションをパージした後に Replication Server を再起動すると、リカバリオペレーションの結果として、それらのトランザクションが再び表示されることがあります。

パーミッション

sysadmin purge_all_open には、“sa” パーミッションが必要です。

参照：

- admin who (107 ページ)
- alter partition (196 ページ)
- create partition (339 ページ)
- sysadmin purge_first_open (497 ページ)
- sysadmin sqt_dump_queue (516 ページ)

sysadmin purge_first_open

Replication Server のインバウンドキューから、最初のオープントランザクションをパージします。

構文

```
sysadmin purge_first_open, q_number, q_type
```

パラメータ

- **q_number, q_type** – パージするステーブルキューを示します。ステーブルキューの値は、**admin who**、**admin who, sqm**、**admin who, sqt** を使用して調べることができます。

例

- **例 1 –** キュー 103:1 から最初のオープントランザクションをパージします。

```
sysadmin purge_first_open, 103, 1
```

使用法

- **sysadmin purge_first_open** は、Replication Server のインバウンドキューから、最初のオープントランザクションを削除します。RepAgent スレッドは、トランザクションをデータベースログから 1 レコードずつ転送します。RepAgent がトランザクション開始レコードと、場合によってはトランザクション内の一部のコマンドをすでに転送していても、トランザクションのコミットレコードまたはアボートレコードがまだ転送されていない場合、そのトランザクションはオープンです。
- **sysadmin purge_first_open** は、インバウンドキューでだけ使用できます。
- ステーブルキューから最初のオープントランザクションをパージするには、Replication Server に十分な領域が必要です。十分なディスク領域がない場合は、次のエラーメッセージが表示されます。

```
This RS is out of Disk Space. Use another session to
add disk space for this command to proceed.
```

  このエラーが発生した場合は、別の **isql** セッションを開始して、Replication Server にステーブル領域 (ディスク領域) を追加してください。十分な領域が使用可能になるまで、**sysadmin purge_first_open** は実行できません。
- 削除するトランザクションの内容を調べるには、このコマンドを使用する前に **sysadmin sqt_dump_queue** を実行してください。
- インバウンドキューにある最初のトランザクションについての情報を表示するには、**adminwho, sqt** を使用します。最初のトランザクションのステータスが “open” (ST:O) であれば、キューから削除できます。
- **sysadmin purge_first_open** コマンドは、Adaptive Server ログにコミットされていないトランザクションがあるときに役立ちます。オープントランザクションは、RepAgent によって Replication Server に配信されます。オープントランザクションがあると、Replication Server はインバウンドキューをトランケートできません。このトランザクションが長時間オープンされていると、インバウンドキューが一杯になり、Replication Server のキュー領域を使い果たしてしまいます。
- キューにある最初のトランザクションがオープンでない場合は、このコマンドはキューを変更しません。トランザクションが削除された後に Replication Server を再起動すると、そのトランザクションは、リカバリオペレーションの結果として再び表示されることがあります。

警告！ sysadmin purge_first_open は、コミットされていないトランザクションでインバウンドキューがスタックされていることが (**admin who, sqt** と **admin who, sqm** を使用して) 確認済みである場合にのみ使用してください。

パーミッション

sysadmin purge_first_open には、“sa” パーミッションが必要です。

参照：

- admin who (107 ページ)
- alter partition (196 ページ)
- create partition (339 ページ)
- sysadmin dump_queue (471 ページ)
- sysadmin purge_all_open (496 ページ)

sysadmin purge_route_at_replicate

レプリケート Replication Server からプライマリ Replication Server へのすべての参照を削除します。

構文

```
sysadmin purge_route_at_replicate, replication_server
```

パラメータ

- **replication_server –** レプリケート側の RSSD からパージするプライマリ Replication Server の名前です。

例

- **例 1 –** レプリケート側の RSSD からプライマリ Replication Server である TOKYO_RS をパージします。

  ```
  sysadmin purge_route_at_replicate, TOKYO_RS
  ```

使用法

- **sysadmin purge_route_at_replicate** は、指定したプライマリ Replication Server からルートを削除した後に、そのプライマリ Replication Server のすべてのサブスクリプションとルート情報を削除するときに使用します。これは、プライマリ Replication Server で **drop route with nowait** を実行した後に使用すると便利です。

- 現在の Replication Server から指定したプライマリ Replication Server へのルートがある場合は、このコマンドを実行する前にルートを削除しておく必要があります。
- プライマリ Replication Server で **drop route with nowait** が実行されたときに、サブスクリプションがマテリアライゼーション中の場合は、レプリケート Replication Server にマテリアライゼーションキューが残ることがあります。このキューを削除するには、**sysadmin drop_queue** を使用してください。

警告！ sysadmin purge_route_at_replicate は、プライマリ Replication Server で **drop route with nowait** コマンドが実行された場合、またはプライマリ Replication Server が失われてリカバリできない場合にのみ使用してください。

パーミッション

sysadmin purge_route_at_replicate には、“sa” パーミッションが必要です。

参照：
- drop route (421 ページ)
- rs_helproute (730 ページ)

sysadmin restore_dsi_saved_segments

バックログされたトランザクションをリストアします。

構文

```
sysadmin restore_dsi_saved_segments, data_server, database
```

パラメータ

- **data_server –** データサーバの名前です。
- **database –** データベースの名前です。

例

- **例 1 –** TOKYO_DS データサーバの *pubs2* データベースのバックログされたトランザクションをリストアします。

  ```
  sysadmin restore_dsi_saved_segments, TOKYO_DS, pubs2
  ```

使用法

- 保存されたセグメントをこのコマンドでリストアする前に、DSI を明示的にサスペンドしてください。
- (**alter connection** を使用して) コネクションにセーブインターバルが指定されたことによって保存されたバックログトランザクションは、すべてデータベースにリストアする対象になります。Replication Server は、**rs_get_lastcommit** を使用してフィルタするトランザクションを決定します。

パーミッション

sysadmin restore_dsi_saved_segments には、“sa” パーミッションが必要です。

参照：

- configure connection (238 ページ)

sysadmin set_dsi_generation

レプリケートデータベースをリストアした後、DSI ステーブルキューのトランザクションが使用されないように、Replication Server のデータベース世代番号を変更します。

構文

```
sysadmin set_dsi_generation, gen_number, primary_data_server,
primary_database, replicate_data_server, replicate_database
```

パラメータ

- **gen_number** – データベースの新しい世代番号です。番号は 0 ～ 65,535 の整数です。
- **primary_data_server** – プライマリサイトのデータサーバの名前です。
- **primary_database** – プライマリデータベースの名前です。
- **replicate_data_server** – レプリケートデータサーバの名前です。
- **replicate_database** – レプリケートデータベースの名前です。

例

- **例 1** – 新しい DSI 世代番号を 105 に設定します。以前の番号は 104 以下でした。

```
sysadmin set_dsi_generation 105 NY_DS, ny_db, SF_DS,
sf_db
```

使用法

sysadmin set_dsi_generation は、データベースダンプのリカバリ時に使用します。リカバリ時以外に世代番号を変更すると、レプリケートデータベースで無効なデータが発生する可能性があります。

リカバリ手順の詳細については、『Replication Server 管理ガイド 第 2 巻』を参照してください。

パーミッション

sysadmin set_dsi_generation には、“sa” パーミッションが必要です。

参照：

- admin get_generation (59 ページ)
- configure connection (238 ページ)
- dbcc dbrepair (599 ページ)
- dbcc settrunc (602 ページ)
- rebuild queues (434 ページ)

sysadmin site_version

SAP Replication Server のサイトバージョン番号を設定します。 サイトバージョン番号を設定すると、対応するバージョンのソフトウェア機能を使用できるようになりますが、以前のバージョンにダウングレードできなくなります。 SAP Replication Server で ERSSD を使用している場合、このコマンドは ERSSD を停止し、データベースファイルをアップグレードして、ERSSD を再起動します。

注意： SAP Replication Server で ERSSD を使用している場合、このコマンドによって ERSSD が再起動されるため、一部のスレッドが停止することがあります。 停止したスレッドをすべて再起動した後に、複写が続行されます。

構文

```
sysadmin site_version [, version]
```

パラメータ

- **version –** SAP Replication Server のサイトバージョン番号です。

バージョン番号	サイトバージョン
11.5 より前	該当なし

バージョン番号	サイトバージョン
11.5	1150
12.0	1200
12.5	1250
12.6	1260
15.0、15.0.1	1500
15.1	1510
15.2	1520
15.6 および 15.5	1550
15.7	1570
15.7.1	1571
15.7.1 SP100	1571100
15.7.1 SP200	1571200

11.5 より前のバージョンに対応するサイトバージョン番号はありません。メンテナンスリリースでは、上位のサイトバージョン番号をサポートしている場合があります。

例

- **例 1 –** SAP Replication Server の現在のサイトバージョン番号を表示します。
```
sysadmin site_version
```
- **例 2 –** サイトバージョン番号をバージョン 15.5 に対応する番号に変更します。
```
sysadmin site_version, 1550
```

使用法

- 現在の SAP Replication Server のサイトバージョン番号を設定するには、**sysadmin site_version** に *version* パラメータを指定して実行します。
  入力するサイトバージョン番号は、ソフトウェアバージョン番号または SAP Replication Server のバージョンレベルよりも高くない番号にしてください。
- SAP Replication Server のサイトバージョン番号を表示するには、*version* パラメータを指定しないで **sysadmin site_version** を実行します。
- SAP Replication Server のサイトバージョンで設定されたバージョンに応じたソフトウェアの新機能を使用できます。

- 新しくインストールしたバージョン 15.7.1 SP200 の SAP Replication Server の場合、サイトバージョン番号は 1571200 です。
- SAP Replication Server の特定のソフトウェアバージョンで導入された機能の詳細については、該当のバージョンの『新機能ガイド』を参照してください。

警告！ サイトバージョン番号を設定すると、以前のバージョンにダウングレードできなくなります。

- SAP Replication Server のインストールとアップグレードの詳細については、使用しているプラットフォーム用の『インストールガイド』と『設定ガイド』を参照してください。

サイトバージョンを設定する際の問題

sysadmin site_version を実行した後に何か予期しないことが発生した場合、またはサイトバージョンの設定に失敗した場合は、**admin show_site_version** を使用して障害に関する情報を表示し、問題を解決してから、**sysadmin site_version** を再実行してプロセス全体を完了させてください。 たとえば、1571100 から 1571200 へのサイトバージョンの変更に失敗した場合、**admin show_site_version** は次の内容を表示します。

```
Site Version Status
------------
-------------------------------------------------------------------------
-----
1571200 The current site_version is '1571200', but eRSSD upgrade
experienced failure.
Please set site_version to '1571200' again to complete the
process.
```

混合バージョンの複写システム

混合バージョンの複写システムでは、SAP Replication Server ごとにサイトバージョンがそれぞれ異なります。 このようなシステムでは、サイトバージョンの高い SAP Replication Server でしか使用できない機能があります。 たとえば、プライマリ SAP Replication Server とそのレプリケート SAP Replication Server の 1 つのサイトバージョンが 1571100 であり、ほかのレプリケート SAP Replication Server のサイトバージョンが 1571200 であるとします。 テーブル複写定義に *timestamp* カラムが存在する場合、サイトバージョンの低いレプリケート SAP Replication Server では *timestamp* へのサブスクライブを *varbinary* (8) として行う必要がありますが、1550 サイトバージョンのレプリケート SAP Replication Server では *timestamp* カラムに直接サブスクライブできます。

ルートのアップグレード

- ルートのいずれか一方の端または両端で、SAP Replication Server を上位のバージョンレベルにアップグレードし、サイトバージョンを上位レベルに設定した場合、ルートをアップグレードする必要があります。 ルートをアップグレードすると、システムテーブル内のデータが再マテリアライズされるため、新し

くアップグレードした Replication Server の新機能についての情報を利用できるようになります。
ルートをアップグレードする場合、次の 2 つのケースが考えられます。
- アップグレード先の SAP Replication Server のバージョンに関係なく、**sysadmin upgrade, “route"** を使用します。
- アップグレード元の SAP Replication Server で新機能が使用されていなかった場合は、**sysadmin fast_route_upgrade** を使用してルートをアップグレードします。

たとえば、SAP Replication Server のバージョン 15.7.1 SP100 をバージョン 15.7.1 SP200 にアップグレードし、これに応じてサイトバージョンを設定する場合、バージョン 15.7.1 SP200 の別の SAP Replication Server からのルートをアップグレードする必要があります。 ルートをアップグレードすると、新しくアップグレードされた SAP Replication Server は、リリース 15.7.1 SP200 の Replication Server からテーブルの追加の複写定義などの情報を受け取ります。
ルートのアップグレードの詳細については、『設定ガイド』を参照してください。

バージョン情報のシステムテーブル

バージョン情報は、*rs_version* システムテーブルに格納されています。また、*rs_routes* システムテーブルにもバージョン情報が含まれています。ルートバージョン情報は、*rs_routeversions* システムテーブルに格納されています。

パーミッション

sysadmin site_version には、“sa” パーミッションが必要です。

参照：
- admin version (103 ページ)
- sysadmin fast_route_upgrade (482 ページ)
- sysadmin system_version (519 ページ)

sysadmin skip_bad_repserver_cmd

次回 Replication Agent が起動したときに、失敗した複写定義要求をスキップするように Replication Server に指示します。

sysadmin skip_bad_repserver_cmd は、複写定義の変更要求手順で使用します。**sysadmin skip_bad_repserver_cmd** を使用する前に、『Replication Server 管理ガイド第 1 巻』の「複写テーブルの管理」を参照してください。

警告！ sysadmin skip_bad_repserver_cmd は注意して使用してください。コマンドを実行した後に、プライマリ Replication Server で正しい複写定義コマンドを実行

せずに Replication Agent を再起動すると、正しくない複写定義バージョンでプライマリデータが複写される場合があります。

構文

```
sysadmin skip_bad_repserver_cmd, pds_name, pdb_name
```

パラメータ

- **pds_name –** プライマリデータサーバの名前。
- **pdb_name –** プライマリデータベースの名前。

例

- **例 1 –** この例では、**sysadmin skip_bad_repserver_cmd** が、SYDNEY_DS データサーバの pubs2 データベースで最後に失敗した複写定義コマンドを省略するよう、Replication Server と Replication Agent に指示します。

  ```
  sysadmin skip_bad_repserver_cmd, SYDNEY_DS, pubs2
  ```

使用法

- *pds_name* および *pdb_name* は、失敗した複写定義要求により影響を受ける特定の Replication Agent を示します。
- **sysadmin skip_bad_repserver_cmd** は、Replication Agent から送信された複写定義要求で失敗したものを省略するよう Replication Server に指示するために使用します。プライマリ Replication Server で複写定義コマンドが失敗すると、Replication Agent が停止します。Replication Server がそのコマンドを省略しない限り、Replication Agent を再起動すると、失敗したコマンドが再び実行されます。**sysadmin skip_bad_repserver_cmd** の実行後、Replication Agent を再起動する前に、Replication Server で正しい複写定義要求を実行します。

パーミッション

sysadmin skip_bad_repserver_cmd には、“sa” パーミッションが必要です。

参照：

- admin verify_repserver_cmd (101 ページ)
- rs_send_repserver_cmd (735 ページ)
- alter replication definition (199 ページ)
- alter applied function replication definition (129 ページ)
- alter request function replication definition (210 ページ)
- create replication definition (346 ページ)

- create applied function replication definition (274 ページ)
- create request function replication definition (362 ページ)
- drop replication definition (419 ページ)

sysadmin sqm_purge_queue

ステーブルキューからすべてのメッセージをパージします。

警告！ ステーブルキューからメッセージをパージするとデータが失われることがあるため、SAP 製品の保守契約を結んでいるサポートセンタからアドバイスを受けた場合にのみ、このコマンドを使用してください。 Replication Server は、パージされたメッセージを送信先データベースや Replication Server に送信できなくなるため、複写システムの一貫性が失われます。 キューにサブスクリプションマーカメッセージまたはルートメッセージが含まれているときにこのコマンドを使用すると、重大な問題が発生することがあります。

構文

```
sysadmin sqm_purge_queue, q_number, q_type
```

パラメータ

- **q_number, q_type** – パージするステーブルキューを示します。これらの値は、**admin who**、**admin who, sqm**、または **admin who, sqt** を使用して確認できます。

例

- **例 1** – インバウンドキュー番号 103 からすべてのメッセージをパージします。

  ```
  sysadmin sqm_purge_queue, 103, 1
  ```

使用法

- **sysadmin sqm_purge_queue** は、別の Replication Server に送信されるメッセージをステーブルキューから削除します。キューがメッセージで満杯になった場合に、このコマンドを使用してください。
- **sysadmin sqm_purge_queue** を実行できるのは、Replication Server をスタンドアロンモードで起動している場合だけです。

パーミッション

“sa” パーミッションが必要です。

参照：
- admin who (107 ページ)
- repserver (741 ページ)

sysadmin sqm_unzap_command

ステーブルキューにあるメッセージの削除を取り消す。

構文

```
sysadmin sqm_unzap_command, q_number, q_type,
 seg, blk, row
```

パラメータ

- **q_number, q_type –** リストアするメッセージがあるステーブルキューを指定します。ステーブルキューの値は、**admin who**、**admin who, sqm**、**admin who, sqt** を使用して調べることができます。
- **seg –** 削除を取り消すメッセージがあるステーブルキュー内のセグメントを指定します。
- **blk –** セグメント内の 16K ブロックを指定します。ブロック番号は 1 ～ 64 です。
- **row –** 削除を取り消すコマンドブロックにあるロー番号です。

使用法

- **sysadmin sqm_unzap_command** を使用するには、Replication Server がスタンドアロンモードでなければなりません。
- **sysadmin sqm_unzap_command** は、ステーブルキューのメッセージから削除マークを削除します。このコマンドは、**sysadmin sqm_zap_command** を使用して削除済みとしてマーク付けしたメッセージをリストアするときに使用します。
- リストアするメッセージを配置するには、**sysadmin dump_queue** を使用します。

パーミッション

sysadmin sqm_unzap_command には、“sa” パーミッションが必要です。

参照：
- admin who (107 ページ)
- sysadmin drop_queue (468 ページ)
- sysadmin sqm_zap_command (511 ページ)

- sysadmin dump_queue (471 ページ)

sysadmin sqm_unzap_tran

特定のトランザクションをステーブルキューにリストアし、リストアされたコマンドの数を示すメッセージを返します。

構文

```
sysadmin sqm_unzap_tran {, q_number, | server [,database]},
q_type, lqid
[, {L0 | L1 | L2 | L3}]
[, {RSSD | client | "log" | file_name}]
```

パラメータ

- **q_number | server[, database]** – ステーブルキューを指定します。*q_number* または *server*[, *database*] を使用して、キュー番号を指定します。**admin who**、**admin who, sqm**、**admin who, sqt** を使用すると、キュー番号を確認できます。
- **q_type** – ステーブルキューのキュータイプです。値は、アウトバウンドキューの場合は 0、インバウンドキューの場合は 1 です。キュータイプを確認するには、**admin who**、**admin who, sqm**、**admin who, sqt** を使用します。
- **lqid** – ステーブルキュートランザクションのコマンドのローカルキュー ID です。*lqid* によって、ステーブルキューにリストアするトランザクションが識別されます。フォーマット：*seg,blk,row*
- **L0** – リストアしたトランザクションの内容をダンプします。**L0**、**L1**、**L2**、または **L3** が指定されていない場合、これがデフォルトの動作です。
- **L1** – リストアしたトランザクションの **begin** コマンドと **end** コマンドだけをダンプします。
- **L2** – リストアしたトランザクションに含まれる他のコマンドの最初の 100 文字とともに、リストアしたトランザクションの **begin** コマンドと **end** コマンドをダンプします。
- **L3** – リストアしたトランザクションのすべてのコマンドをダンプします。SQL 文を除き、他のすべてのコマンドがコメントとして出力されます。**L3** を使用できるのは、*file_name* オプションまたは **sysadmin dump_file** コマンドを使用して代替ログファイルを指定した場合だけです。**L3** は、**RSSD** オプションまたは **client** オプションとともに使用することはできません。
- **RSSD** – RSSD のシステムテーブルが出力先として指定されます。
- **client** – このコマンドを発行したクライアントが出力先として指定されます。
- **"log"** – Replication Server ログファイルが出力先として指定されます。

- **file_name –** *file_name* ログファイルが出力先として指定されます。**sysadmin dump_file** コマンドを使用して、代替ログファイルを設定することもできます。

例

- **例 1 –** キュー 103:1 の LQID が 0:15:2 であるトランザクションをリストアし、このトランザクションを Replication Server ログにダンプします。

```
sysadmin sqm_unzap_tran, 103, 1, 0, 15, 2
```

- **例 2 –** SYDNEY_DS.*pubs2* のインバウンドキューの LQID が 0:15:2 であるトランザクションをリストアし、このトランザクションを Replication Server ログにダンプします。

```
sysadmin sqm_unzap_tran, SYDNEY_DS, pubs2, 1, 0, 15,
2, "log"
```

- **例 3 –** キュー 103:1 の LQID が 0:15:2 であるトランザクションをリストアし、このトランザクションの **begin** コマンドと **end** コマンドを Replication Server ログにダンプします。

```
sysadmin sqm_unzap_tran, 103,1, 0, 15, 2, L1
```

- **例 4 –** キュー 103:1 の LQID が 0:15:2 であるトランザクションをリストアし、このトランザクションを Replication Server ログにダンプします。コマンドはすべて 100 文字にトランケートされます。

```
sysadmin sqm_unzap_tran, 103,1, 0, 15, 2, L2
```

- **例 5 –** キュー 103:1 の LQID が 0:15:2 であるトランザクションをリストアし、このトランザクションを `SYDNEY_RS.log` ファイルにダンプします。

```
sysadmin sqm_unzap_tran, 103,1, 0, 15, 2, L3,
SYDNEY_RS.log
```

- **例 6 –** キュー 103:1 の LQID が 0:15:2 であるトランザクションをリストアし、このトランザクションを RSSD にダンプします。

```
sysadmin sqm_unzap_tran, 103, 1, 0, 15, 2, RSSD
```

- **例 7 –** キュー 103:1 の LQID が 0:15:2 であるトランザクションをリストアし、このトランザクションをクライアントにダンプします。

```
sysadmin sqm_unzap_tran, 103, 1, 0, 15, 2, client
```

使用法

- **sysadmin sqm_unzap_tran** を使用するには、Replication Server がスタンドアロンモードである必要があります。
- **sysadmin sqm_unzap_tran** は、ステーブルキューのトランザクションから削除マークを削除します。このコマンドは、**sysadmin sqm_zap_tran** を使用して削

除済みとしてマーク付けしたトランザクションをリストアするときに使用します。

- リストアするトランザクションを配置するには、**sysadmin dump_queue** を使用します。
- **sysadmin sqm_unzap_tran** は、リストアしたトランザクションの内容を次のいずれかにダンプします。
  - Replication Server ログ
  - 代替ログファイル
  - RSSD
  - コマンドを発行したクライアント

  キューを RSSD またはクライアントにダンプするには、**sysadmin dump_queue** の最後の引数に **RSSD** または **client** を指定する必要があります。

  RSSD オプションまたは **client** オプションが指定されていない場合、または "**log**" オプションが指定されている場合は、出力先が Replication Server ログになります。

  sysadmindump_file コマンドまたは *file_name* オプションを使用して、キューをダンプする代替ログファイルが指定されている場合、出力先はその代替ダンプファイルになります。

パーミッション

sysadmin sqm_unzap_tran には、“sa” パーミッションが必要です。

参照：

- admin who (107 ページ)
- sysadmin sqm_unzap_command (508 ページ)
- sysadmin sqm_zap_command (511 ページ)
- sysadmin sqm_zap_tran (513 ページ)

sysadmin sqm_zap_command

ステーブルキューの 1 つのメッセージを削除する。

構文

```
sysadmin sqm_zap_command, q_number, q_type,
 seg, blk, row
```

パラメータ

- **q_number, q_type –** 削除するメッセージがあるステーブルキューを指定します。ステーブルキューの値は、**admin who**、**admin who, sqm**、**admin who, sqt** を使用して調べることができます。
- **seg –** ステーブルキュー内のセグメントを指定します。
- **blk –** セグメント内の 16K ブロックを指定します。ブロック番号は 1 ～ 64 です。
- **row –** 削除するコマンドブロックにあるロー番号です。

例

- **例 1 –**

```
sysadmin sqm_zap_command
```

```
sysadmin sqm_zap_command, 103, 1, 15, 65, 2
```

使用法

- **sysadmin sqm_zap_command** を使用するには、Replication Server がスタンドアロンモードである必要があります。
- 削除するメッセージを配置するには、**sysadmin dump_queue** を使用します。
- **sysadmin sqm_zap_command** は、ステーブルキュー内のメッセージを「削除」とマーク付けします。Replication Server がキューを処理するときに、マーク付けされたメッセージは無視されます。
- **sysadmin sqm_unzap_command** を使用してメッセージをリストアできます。このコマンドは、メッセージから削除マークを削除します。
- メッセージを削除して Replication Server をノーマルモードで再起動すると、メッセージを保持しているキューの一部が処理されていることがあります。この場合は、**sysadmin sqm_unzap_command** を使用してメッセージをリストアすることはできません。

パーミッション

sysadmin sqm_zap_command には、“sa” パーミッションが必要です。

参照：

- admin who (107 ページ)
- sysadmin dump_queue (471 ページ)
- sysadmin sqm_unzap_command (508 ページ)

sysadmin sqm_zap_tran

特定のトランザクションをステーブルキューから削除し、削除されたコマンドの数を示すメッセージを返します。

構文

```
sysadmin sqm_zap_tran {, q_number, | server [,database]},
q_type, lqid
[, {L0 | L1 | L2 | L3}]
[, {RSSD | client | "log" | file_name}]
```

パラメータ

- **q_number | server[, database] –** ステーブルキューを指定します。*q_number* または *server*[, *database*] を使用して、キュー番号を指定します。**admin who**、**admin who, sqm**、**admin who, sqt** を使用すると、キュー番号を確認できます。
- **q_type –** ステーブルキューのキュータイプです。値は、アウトバウンドキューの場合は "0"、インバウンドキューの場合は "1" です。キュータイプを確認するには、**admin who**、**admin who, sqm**、**admin who, sqt** を使用します。
- **lqid –** ステーブルキュートランザクションのコマンドのローカルキュー ID です。*lqid* によって、ステーブルキューから削除するトランザクションが識別されます。フォーマット：*seg,blk,row*
- **L0 –** 削除したトランザクションの内容をダンプします。**L0**、**L1**、**L2**、または **L3** が指定されていない場合、これがデフォルトの動作です。
- **L1 –** 削除したトランザクションの **begin** コマンドと **end** コマンドだけをダンプします。
- **L2 –** 削除したトランザクションに含まれる他のコマンドの最初の 100 文字とともに、削除したトランザクションの **begin** コマンドと **end** コマンドをダンプします。
- **L3 –** 削除したトランザクションのすべてのコマンドをダンプします。**SQL** 文を除き、他のすべてのコマンドがコメントとして出力されます。**L3** を使用できるのは、*file_name* オプションまたは **sysadmin dump_file** コマンドを使用して代替ログファイルを指定した場合だけです。**L3** は、**RSSD** オプションまたは **client** オプションとともに使用することはできません。
- **RSSD –** RSSD のシステムテーブルが出力先として指定されます。
- **client –** このコマンドを発行したクライアントが出力先として指定されます。
- **"log" –** Replication Server ログファイルが出力先として指定されます。

- **file_name –** *file_name* ログファイルが出力先として指定されます。**sysadmin dump_file** コマンドを使用して、代替ログファイルを設定することもできます。

例

- **例 1 –** キュー 103:1 の LQID が 0:15:2 であるトランザクションを削除し、このトランザクションを Replication Server ログにダンプします。

```
sysadmin sqm_zap_tran, 103, 1, 0, 15, 2
```

- **例 2 –** *SYDNEY_DS.pubs2* のインバウンドキューの LQID が 0:15:2 であるトランザクションを削除し、このトランザクションを Replication Server ログにダンプします。

```
sysadmin sqm_zap_tran, SYDNEY_DS, pubs2, 1, 0, 15,
2, "log"
```

- **例 3 –** キュー 103:1 の LQID が 0:15:2 であるトランザクションを削除し、このトランザクションの **begin** コマンドと **end** コマンドを Replication Server ログにダンプします。

```
sysadmin sqm_zap_tran, 103,1, 0, 15, 2, L1
```

- **例 4 –** キュー 103:1 の LQID が 0:15:2 であるトランザクションを削除し、このトランザクションを Replication Server ログにダンプします。コマンドはすべて 100 文字にトランケートされます。

```
sysadmin sqm_zap_tran, 103,1, 0, 15, 2, L2
```

- **例 5 –** キュー 103:1 の LQID が 0:15:2 であるトランザクションを削除し、このトランザクションを `SYDNEY_RS.log` ファイルにダンプします。

```
sysadmin sqm_zap_tran, 103,1, 0, 15, 2, L3,
SYDNEY_RS.log
```

- **例 6 –** キュー 103:1 の LQID が 0:15:2 であるトランザクションを削除し、このトランザクションを RSSD にダンプします。

```
sysadmin sqm_zap_tran, 103, 1, 0, 15, 2, RSSD
```

- **例 7 –** キュー 103:1 の LQID が 0:15:2 であるトランザクションを削除し、このトランザクションをクライアントにダンプします。

```
sysadmin sqm_zap_tran, 103, 1, 0, 15, 2, client
```

使用法

- **sysadmin sqm_zap_tran** を使用するには、Replication Server がスタンドアロンモードである必要があります。
- 削除するトランザクションを配置するには、**sysadmin dump_queue** を使用します。

- **sysadmin sqm_zap_tran** は、ステーブルキューのトランザクションを削除済みとしてマーク付けします。Replication Server がキューを処理するときに、マーク付けされたトランザクションは無視されます。
- **sysadmin sqm_unzap_tran** を使用してトランザクションをリストアできます。**sysadmin sqm_unzap_tran** コマンドは、トランザクションから削除マークを削除します。
- トランザクションを削除して Replication Server をノーマルモードで再起動すると、トランザクションを保持しているキューの一部が処理されていることがあります。この場合は、**sysadmin sqm_unzap_tran** を使用してトランザクションをリストアすることはできません。
- **sysadmin sqm_zap_tran** は、削除対象としてマーク付けされたトランザクションを次のいずれかにダンプします。
  - Replication Server ログ
  - 代替ログファイル
  - RSSD
  - コマンドを発行したクライアント

  キューを RSSD またはクライアントにダンプするには、**sysadmin dump_queue** の最後の引数に **RSSD** または **client** を指定する必要があります。

  RSSD オプションまたは **client** オプションが指定されていない場合、または "**log**" オプションが指定されている場合は、出力先が Replication Server ログになります。

  sysadmindump_file コマンドまたは *file_name* オプションを使用して、キューをダンプする代替ログファイルが指定されている場合、出力先はその代替ダンプファイルになります。

パーミッション

sysadmin sqm_zap_command には、“sa” パーミッションが必要です。

参照：

- admin who (107 ページ)
- sysadmin dump_queue (471 ページ)
- sysadmin sqm_unzap_command (508 ページ)
- sysadmin sqm_unzap_tran (509 ページ)
- sysadmin sqm_zap_command (511 ページ)

sysadmin sqt_dump_queue

インバウンドキューまたは DSI キューのトランザクションキャッシュをダンプします。

構文

```
sysadmin sqt_dump_queue {, q_number | server[, database]},
q_type, reader
[, {open | closed | read}]
[, num_cmds]
[, {L0 | L1 | L2 | L3}]
[, {RSSD | client | "log" | file_name}]
```

パラメータ

- **q_number | server[, database]** – インバウンドキューまたは DSI キューを指定します。*q_number* または *server*[, *database*] を使用して、キュー番号を指定します。**admin who**、**admin who, sqm**、**admin who, sqt** を使用すると、キュー番号を確認できます。
- **q_type** – ステーブルキューのキュータイプです。値は、アウトバウンドキューの場合は 0、インバウンドキューの場合は 1 です。キュータイプを確認するには、**admin who**、**admin who, sqm**、**admin who, sqt** を使用します。
- **reader** – ステーブルキューをダンプするリーダを指定します。このパラメータは、ウォームスタンバイ・アプリケーションなどの複数のリーダを必要とする機能に使用します。リーダ番号は、**admin sqm_readers** または **admin who, sqt** を使用して調べることができます。複数のリーダを使用しない場合は、"0" を入力します。
- **open** – オープントランザクションだけをダンプします。このオプションを使用する場合は、*q_type* と **open** フラグの間にカンマを挿入してください。
- **closed** – SQT キャッシュにあるコミットされたすべてのトランザクションをダンプします。
- **read** – SQT キャッシュにあるリストアされたすべてのリードトランザクションをダンプします。
- **num_cmds** – ダンプするコマンドの数を指定します。*num_cmds* を -1 に設定すると、SQT キャッシュ内のすべてのコマンドがダンプされます。
- **L0** – SQT キャッシュのすべての内容をダンプします。**L0**、**L1**、**L2**、または **L3** が指定されていない場合、これがデフォルトの動作です。
- **L1** – SQT キャッシュにあるトランザクションの **begin** コマンドと **end** コマンドだけをダンプします。

- **L2 –** トランザクションに含まれる他のすべてのコマンドの短縮バージョンとともに、SQT キャッシュのトランザクションの **begin** コマンドと **end** コマンドをダンプします。
- **L3 –** キャッシュの内容をすべてダンプします。**SQL** 文を除き、他のすべてのコマンドがコメントとして出力されます。**L3** を使用できるのは、*file_name* オプションまたは **sysadmin dump_file** コマンドを使用して代替ログファイルを指定した場合だけです。**L3** は、**RSSD** オプションまたは **client** オプションとともに使用することはできません。
- **RSSD –** RSSD のシステムテーブルが出力先として指定されます。
- **client –** このコマンドを発行したクライアントが出力先として指定されます。
- **"log" –** Replication Server ログファイルが出力先として指定されます。
- **file_name –** *file_name* で指定した代替ログファイルが出力先として指定されます。代替ログファイルは、**sysadmin dump_file** コマンドを使用して設定することもできます。このファイルのロケーションは、Replication Server ログに記録されます。

例

- **例 1 –** トランザクションキャッシュから、キュー 103:1 にあるリストアされたすべてのトランザクションをダンプします。

```
sysadmin sqt_dump_queue, 103, 1, 0
```

- **例 2 –** SYDNEY_DS.*pubs2* のインバウンドキューのトランザクションキャッシュにあるリストアされたすべてのトランザクションを Replication Server ログにダンプします。

```
sysadmin sqt_dump_queue, SYDNEY_DS, pubs2, 1, 0
```

- **例 3 –** キュー 103:1 のトランザクションキャッシュにあるリストアされたすべてのオープントランザクションを Replication Server ログにダンプします。

```
sysadmin sqt_dump_queue, 103,1, 0, open
```

- **例 4 –** キュー 103:1 のトランザクションキャッシュにあるリストアされたすべてのクローズトランザクションを Replication Server ログにダンプします。

```
sysadmin sqt_dump_queue, 103,1, 0, closed
```

- **例 5 –** キュー 103:1 のトランザクションキャッシュにあるリストアされたすべてのリードトランザクションを Replication Server ログにダンプします。

```
sysadmin sqt_dump_queue, 103,1, 0, read
```

- **例 6 –** キュー 103:1 のトランザクションキャッシュにあるリストアされたトランザクションの最初の 10 個のコマンドを Replication Server ログにダンプします。

```
sysadmin sqt_dump_queue, 103,1, 0, 10
```

- **例 7 –** キュー 103:1 のトランザクションキャッシュにあるリストアされたすべてのトランザクションの **begin** コマンドと **end** コマンドを Replication Server ログにダンプします。
  ```
  sysadmin sqt_dump_queue, 103,1, 0, L1
  ```
- **例 8 –** キュー 103:1 のトランザクションキャッシュにあるリストアされたすべてのトランザクションを Replication Server ログにダンプします。コマンドはすべて 100 文字にトランケートされます。
  ```
  sysadmin sqt_dump_queue, 103,1, 0, L2
  ```
- **例 9 –** キュー 103:1 のトランザクションキャッシュにあるリストアされたすべてのトランザクションを `SYDNEY_RS.log` ファイルにダンプします。
  ```
  sysadmin sqt_dump_queue, 103,1, 0, L3, SYDNEY_RS.log
  ```
- **例 10 –** キュー 103:1 のリストアされたすべてのトランザクションをトランザクションログから RSSD にダンプします。
  ```
  sysadmin sqt_dump_queue, 103,1, 0, RSSD
  ```
- **例 11 –** キュー 103:1 のリストアされたすべてのトランザクションをトランザクションログからクライアントにダンプします。
  ```
  sysadmin sqt_dump_queue, 103,1, 0, client
  ```

使用法

- **sysadmin sqt_dump_queue** を使用する前に、**admin who, sqt** を実行してデータベースのトランザクションキャッシュが存在することを確認します。
- このコマンドは、トランザクションキャッシュにあるすべてのトランザクション文をダンプします。
- **sysadmin sqt_dump_queue** は、トランザクション文を次のいずれかにダンプします。
  - Replication Server ログ
  - 代替ログファイル
  - RSSD
  - コマンドを発行したクライアント

  トランザクションを RSSD またはクライアントにダンプするには、**sysadmin sqt_dump_queue** の最後の引数に **RSSD** または **client** を指定する必要があります。

  sysadmindump_file コマンドまたは *file_name* オプションを使用して、トランザクションをダンプする代替ログファイルが指定されている場合、出力先はその代替ダンプファイルになります。

  RSSD オプションまたは **client** オプション i が指定されていない場合、または **log** オプションが指定されている場合は、出力先が Replication Server ログになります。

- **sysadmin sqt_dump_queue** の出力には、トランザクションキャッシュ内のトランザクションのステータス (open、closed、または read) が示されます。オープン (open) トランザクションは、まだコミットされていないトランザクションです。クローズ (closed) トランザクションは、コミットされてはいますが、まだ完全に読み込まれていないことを示します。リードトランザクションは、完全に読み取られていますが、削除されていないことを示します。
- 設定パラメータ **sqt_max_cache_size** を設定して、キャッシュサイズを修正できます。

パーミッション

sysadmin sqt_dump_queue には、“sa” パーミッションが必要です。

参照：

- admin who (107 ページ)
- sysadmin dump_file (470 ページ)

sysadmin system_version

複写システム用のシステムワイドなバージョン番号を表示または設定し、対応するリリースレベルのソフトウェア機能を使用できるようにする。

バージョン 11.5 から、個々の Replication Server のサイトバージョンでも新機能を有効にできるようになりました。システムバージョン番号は、現在のソフトウェアバージョンと対応させる必要はありません。

構文

```
sysadmin system_version [, version]
```

パラメータ

- **version –** 複写システムで使用するシステムバージョン番号です。

例

- **例 1 –** ID サーバで実行すると、現在のシステムバージョン番号が表示されます。

  ```
  sysadmin system_version
  ```

- **例 2 –** ID サーバで実行すると、バージョン 15.1 に対応するシステムバージョン番号に変更されます。この番号は、次のような場合に使用できます。

- すべての Replication Server がバージョン 15.1 である場合
- Replication Server を以前のバージョンにダウングレードする必要がない場合
- 以前のバージョンの Replication Server をインストールする必要がない場合

```
sysadmin system_version, 1510
```

使用法

- システムバージョン番号を設定するには、ID サーバで **sysadmin system_version** を実行し、*version* パラメータを組み込んでください。
  - システムバージョン番号には、複写システム内の Replication Server の最も低いソフトウェアバージョン番号 (Replication Server のリリースレベル) よりも高いものを入力しないでください。
  - システムバージョン番号は、ID サーバ以外の Replication Server では設定できません。
- 現在のシステムバージョン番号を表示するには、ID サーバで *version* パラメータを指定せずに **sysadmin system_version** を実行します。

  このコマンドを別の Replication Server で実行すると、その Replication Server は、ID サーバに問い合わせて、現在のシステムバージョン番号を判別しようとします。まれに、Replication Server が ID サーバに接続できないことがあります。そのため、正しい値であることが保証されているのは ID サーバ内の値だけです。

システムバージョンとサイトバージョン

- Replication Server リリース 11.5 から、Replication Server のサイトバージョン番号が現在のソフトウェアバージョンに設定されている場合 (たとえば、リリース 15.1 の場合は 1510)、特定の新しいソフトウェア機能を使用できるようになりました。詳細については、「**sysadmin site_version**」の項を参照してください。

  最小システムバージョン番号である 1102 も必要です。
- バージョン 11.5 以降の Replication Server を新しい複写システムの ID サーバとしてインストールすると、システムバージョン番号が 1102 に設定されます。この番号を使用することによって、バージョン 11.0.2 以降の別の Replication Server をシステムにインストールできるようになります。
- Replication Server のインストールとアップグレードについては、使用しているプラットフォーム用の『Replication Server インストールガイド』と『Replication Server 設定ガイド』を参照してください。

混合バージョンの複写システム

すべての Replication Server がバージョン 11.0.2 以降の場合、システムバージョン番号に必要な最上位の設定は 1102 になります。システムバージョン番号を 1102 に設定したら、この値を設定し直す必要はありません。

システムバージョン番号 1102 と個々の Replication Server のサイトバージョン番号を使用すると、サイトバージョンの異なる Replication Server を同時に稼働させることができる、混合バージョンの複写システムが可能になります。各 Replication Server は、用意されている機能をすべて使用できます。

混合バージョンの複写システムでは、サイトバージョンの高い Replication Server でしか使用できない機能があります。たとえば、プライマリ Replication Server とレプリケート Replication Server の 1 つのサイトバージョンが 1510 であり、他のレプリケート Replication Server のサイトバージョンが 1260 であるとします。テーブル複写定義に *timestamp* カラムが含まれている場合、サイトバージョンの低いレプリケート Replication Server は、*timestamp* のサブスクリプション (*varbinary* (8) として) しか作成できませんが、サイトバージョンが 1510 であるレプリケート Replication Server は、*timestamp* カラムのサブスクリプションを直接作成できます。詳細については、「sysadmin site_version」の項を参照してください。

Replication Server の特定のソフトウェアバージョンで導入された機能の詳細については、該当のバージョンの『Replication Server 新機能ガイド』を参照してください。

システムバージョンと ID サーバ

ID サーバ以外の Replication Server は、特定の最小システムバージョンを必要とするコマンドを実行するときに、そのコマンドの使用を許可する前に ID サーバに問い合わせ、現在のシステムバージョン番号を確認します。

バージョン情報のシステムテーブル

バージョン情報は、*rs_version* システムテーブルに格納されています。また、*rs_routes* システムテーブルにもバージョン情報が含まれています。

パーミッション

sysadmin system_version には、“sa” パーミッションが必要です。

参照：

- admin version (103 ページ)
- sysadmin site_version (502 ページ)

sysadmin upgrade, "database"

Replication Server によってサービスされるユーザのデータベースをアップグレードします。

構文

```
sysadmin upgrade, "database"{, data_server,database | all}
```

パラメータ

- ***data_server, database*** – アップグレードするデータベースを指定します。データベースごとに別のコマンドを入力する必要があります。
- **all** – Replication Server がサービスを提供するデータベースをすべてアップグレードします。データベースがアップグレードの条件を満たしていない場合は、エラーメッセージが表示されます。

例

- **例 1** – `pds` データサーバの `pdb01` データベースをアップグレードします。`pdb01` サービスを提供している Replication Server で次のように入力します。

  ```
  sysadmin upgrade, database, pds, pdb01
  ```

  データベースのアップグレードに失敗した場合は、Replication Server のエラーログに次のようなエントリが表示されます。

  ```
  Database is not accessible.
  Fail to upgrade data_server.database.
  ```

使用法

- アップグレードした Replication Server で **admin version, “connection”** と入力して、アップグレードが必要なユーザデータベースを特定します。
- Replication Server を 15.7.1 以降にアップグレードすると、Replication Server は SAP IQ レプリケートへのレプリケートコネクションをサスペンドします。**admin who** を使用した場合は、"Awaiting Upgr" ステータスが表示されます。**sysadmin upgrade, "database"** を使用して SAP IQ データベースをアップグレードします。

『設定ガイド』の「sysadmin upgrade, "database" を使用したユーザデータベースアップグレードの修正自動アップグレード」を参照してください。

パーミッション

sysadmin upgrade, "database" には "sa" パーミッションが必要です。

参照：
- admin version, "connection" (104 ページ)
- admin who (107 ページ)
- repserver (741 ページ)

sysadmin upgrade, route

現在の Replication Server から送信先 Replication Server までのルートをアップグレードし、失敗したアップグレードルートをリカバリします。

構文

```
sysadmin upgrade, "route", dest_rs [,"recovery"]
```

パラメータ

- ***dest_rs*** – 送信先 Replication Server の名前です。
- ***recovery*** – ルートのアップグレードに失敗した場合に、リカバリします。

例

- **例 1** – NY_RS Replication Server から LON_RS Replication Server までのルートをアップグレードします。NY_RS で以下のコマンドを実行します。

  ```
  sysadmin upgrade, "route", LON_RS
  ```

  次のようなメッセージが表示されます。

  ```
  Route upgrade for route 'NY_RS.LON_RS' is in progress in the
  background.
  ```

- **例 2** – 失敗したルートアップグレードをリカバリします。NY_RS で以下のコマンドを実行します。

  ```
  sysadmin upgrade,"route", LON_RS, "recovery"
  ```

使用法

- **sysadmin upgrade, route, dest_rs** コマンドを使用して、現在の Replication Server から送信先 Replication Server までのルートをアップグレードします。`dest_rs` は送信先 Replication Server の名前です。

- ルートのアップグレードに失敗した場合は、**recovery** オプションを使用して前回のルートアップグレードからリカバリします。
- コマンドの実行時に使用するユーザ ID とパスワードは、送信先 Replication Server と送信先 Replication Server の RSSD にも存在する必要があります。

『設定ガイド』の「ルートのアップグレード」を参照してください。

パーミッション

sysadmin upgrade, route を使用する場合は、送信先 Replication Server で "sa" パーミッション、送信先 Replication Server の RSSD で dba パーミッションが必要です。

validate publication

パブリケーションのステータスを VALID に設定して、そのパブリケーションの新しいサブスクリプションを作成できるようにします。

構文

```
validate publication pub_name
with primary at data_server.database
```

パラメータ

- **pub_name –** 確定化するパブリケーションの名前です。
- **with primary at data_server.database –** プライマリデータのロケーションを指定します。プライマリデータベースがウォームスタンバイアプリケーションの一部である場合、*data_server.database* は論理データサーバと論理データベースの名前になります。

例

- **例 1 –** パブリケーション *pubs2_pub* を確定化します。

```
validate publication pubs2_pub
    with primary at TOKYO_DS.pubs2
```

使用法

- パブリケーションのすべてのアーティクルが作成されたら、**validate publication** を使用してパブリケーションを確定化してから、レプリケートサイトでそのパブリケーションのサブスクリプションを作成します。パブリケーションを確定化することによって、パブリケーションに 1 つ以上のアーティクルが含まれて

いることが確認され、そのパブリケーションには、サブスクリプションの作成準備ができていることを示すマークが付けられます。
- **validate publication** は、**create publication** を使用してパブリケーションを作成した Replication Server で実行してください。
- パブリケーションのステータスを確認するには、**check publication** を使用してください。このコマンドは、パブリケーションに含まれるアーティクルの数を表示し、パブリケーションが有効 (VALID) かどうかを示します。

サブスクリプションマテリアライゼーションの詳細については、『Replication Server 管理ガイド 第 1 巻』と『Replication Server 管理ガイド 第 2 巻』を参照してください。

パーミッション

validate publication には、“create object” パーミッションが必要です。

参照：
- check publication (231 ページ)
- check subscription (233 ページ)
- create publication (341 ページ)
- create subscription (376 ページ)
- define subscription (395 ページ)
- drop publication (418 ページ)

validate subscription

複写定義またはパブリケーションのサブスクリプションに対して、サブスクリプションステータスを VALID に設定します。

このコマンドは、バルクマテリアライゼーション処理の一部、またはパブリケーションサブスクリプションのリフレッシュ処理の一部です。

注意： **catchup_queue** オプションを指定してサブスクリプションをアクティブ化した場合、サブスクリプションが VALID になるのは、キャッチアップキュー内のすべての DML オペレーションがレプリケートテーブルに適用された後に限ります。

構文

```
validate subscription sub_name
for {table_rep_def | function_rep_def |
      publication pub_name
```

```
      with primary at data_server.database}
with replicate at data_server.database
```

パラメータ

- **sub_name –** 確定化するサブスクリプションの名前です。
- **for table_rep_def –** サブスクリプションの対象となるテーブル複写定義の名前を指定します。
- **for function_rep_def –** サブスクリプションの対象となるファンクション複写定義の名前を指定します。
- **for publication pub_name –** サブスクリプションの対象となるパブリケーションの名前を指定します。
- **with primary at data_server.database –** プライマリデータのロケーションを指定します。プライマリデータベースがウォームスタンバイアプリケーションの一部である場合、*data_server.database* は論理データサーバと論理データベースの名前になります。この句を使用するのは、パブリケーションのサブスクリプションの場合だけです。
- **with replicate at data_server.database –** レプリケートデータのロケーションを指定します。レプリケートデータベースがウォームスタンバイアプリケーションの一部である場合、*data_server.database* は論理データサーバと論理データベースの名前になります。

例

- **例 1 –** テーブル複写定義 *titles_rep* のサブスクリプション *titles_sub* を確定化します。ここでは、レプリケートデータベースは SYDNEY_DS.*pubs2* です。

```
validate subscription titles_sub
 for titles_rep
 with replicate at SYDNEY_DS.pubs2
```

- **例 2 –** ファンクション複写定義 *myproc_rep* のサブスクリプション *myproc_sub* を確定化します。ここでは、レプリケートデータベースは SYDNEY_DS.*pubs2* です。

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

- **例 3 –** パブリケーション *pubs2_pub* のサブスクリプション *pubs2_sub* を確定化します。ここでは、プライマリデータベースは TOKYO_DS.*pubs2*、レプリケートデータベースは SYDNEY_DS.*pubs2* です。

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

使用法

- **validate subscription** は、プライマリ Replication Server とレプリケート Replication Server で、サブスクリプションを確定化するときに使用します。このサブスクリプションとは、テーブル複写定義、ファンクション複写定義、またはパブリケーションへのサブスクリプションです。
- このコマンドは、バルクマテリアライゼーションを完了します。最初のステップでは、**define subscription** を使用してサブスクリプションを作成します。2 番目のステップでは、**activate subscription** を使用してサブスクリプションをアクティブ化します。
- 既存のサブスクリプションを持つパブリケーションに新しいアーティクルを追加した場合は、新しいアーティクルのサブスクリプションを作成するために、パブリケーションサブスクリプションをリフレッシュしてください。
  define subscription と **activate subscription** を使用して、パブリケーションサブスクリプション内に新しいアーティクルサブスクリプションを作成し、アクティブ化します。次に、新しいアーティクルサブスクリプションのサブスクリプションデータを手動でロードし、**validate subscription** を使用して、パブリケーションサブスクリプションを確定化します。
- **validate subscription** は、**define subscription** を使用してサブスクリプションを作成した Replication Server で実行してください。
- パブリケーションサブスクリプションを確定化すると、そのすべてのアーティクルサブスクリプションも同時に確定化されます。
- **validate subscription** は、サブスクリプションのステータスを ACTIVE から VALID へ変更します。これ以降、プライマリデータサーバで実行される更新は、プライマリ Replication Server を通じて分配され、レプリケート Replication Server で適用されます。
- このコマンドを使用すると、複数のサイトで RSSD テーブルが修正されます。プライマリおよびレプリケート Replication Server の両方で **check subscription** を使用して、それぞれの効果を確認してください。

サブスクリプションマテリアライゼーションの詳細については、『Replication Server 管理ガイド 第 1 巻』と『Replication Server 管理ガイド 第 2 巻』を参照してください。

パーミッション

validate subscription には、データを複写するサイトでは “create object” パーミッション、プライマリデータを格納するサイトでは “primary subscribe” または “create object” パーミッションが必要です。

参照：

- activate subscription (48 ページ)

- check subscription (233 ページ)
- create article (280 ページ)
- create publication (341 ページ)
- create subscription (376 ページ)
- define subscription (395 ページ)
- drop subscription (424 ページ)

wait for create standby

スタンバイデータベースの作成処理が完了するまで、Replication Server のクライアントセッションが待機できるようにするブロック用コマンドです。

構文

```
wait for create standby
 for logical_ds.logical_db
```

パラメータ

- **logical_ds –** 論理コネクションのデータサーバの名前です。
- **logical_db –** 論理コネクションのデータベースの名前です。

使用法

- スタンバイデータベースが作成されると、**wait for create standby** はステータス情報を表示します。
- **wait for create standby** は、スクリプトで使用すると非常に役立ちます。

パーミッション

wait for create standby には、“sa” パーミッションが必要です。

参照：

- abort switch (47 ページ)
- switch active (456 ページ)
- wait for switch (529 ページ)

wait for delay

このコマンドをブロックする時間間隔を指定します。

構文

```
wait for delay 'time_string'
```

パラメータ

- **time_string –** 実行前の経過時間です。hh:mm[:ss[.xxx]] [am|pm] のフォーマットを使用します。

例

- **例 1 –** このコマンドは、1 時間 30 分コマンドをブロックするように Replication Server に指示します。

  ```
  wait for delay '01:30'
  ```

使用法

- **wait for delay** は、指定した時間が経過するまで待機するように Replication Server に指示するときに使用します。サブスクリプションを実装するときに使用するのが一般的です。通常、**wait for delay** は、2 つのサブスクリプション間で発行されます。
- 時間、分、秒を指定できます。指定できる最大時間は 24 時間です。

パーミッション

このコマンドは、すべてのユーザが実行できます。

参照：

- wait for time (530 ページ)

wait for switch

新しいアクティブデータベースへの切り替えが完了するまで、Replication Server のクライアントセッションが待機できるようにするブロック用コマンドです。

構文

```
wait for switch
 for logical_ds.logical_db
```

パラメータ

- **logical_ds –** 論理コネクションのデータサーバの名前です。
- **logical_db –** 論理コネクションのデータベースの名前です。

使用法

- **switch active** オペレーションが完了すると、**wait for switch** はステータス情報を表示します。
- **wait for switch** は、スクリプトで使用すると非常に役立ちます。

パーミッション

wait for switch には、“sa” パーミッションが必要です。

参照：

- abort switch (47 ページ)
- switch active (456 ページ)
- wait for create standby (528 ページ)

wait for time

このコマンドをブロック解除する時刻を指定します。

構文

```
wait for time 'time_string'
```

パラメータ

- **time_string –** 特定の実行時刻です。hh:mm[:ss[.xxx]] [am|pm] のフォーマットを使用します。

例

- **例 1 –** このコマンドは、午後 5 時 30 分まで待機するように Replication Server に指示します。
  ```
  wait for time '05:30 pm'
  ```

使用法

- **wait for time** は、指定した時間まで待機するように Replication Server に指示するときに使用します。

- 時間、分、秒を指定できます。指定できる最大時間は 24 時間です。
  現在の時刻が午後 6 時の場合、**wait for time '5:00 pm'** は明日の午後 5 時を示します。

パーミッション

このコマンドは、すべてのユーザが実行できます。

参照：

- wait for delay (529 ページ)

SAP Replication Server システムファンクション

SAP Replication Server は、プライマリデータベースからのコマンドを、**insert**、**delete**、**select**、**begin transaction** などのデータサーバのオペレーションを表す SAP Replication Server ファンクションに変換します。 これらのファンクションは、システム内のリモート SAP Replication Server に分配されます。リモート SAP Replication Server は、これらのオペレーションをリモートデータベースで実行します。

システムファンクションのファンクション文字列のカスタマイズについては、『管理ガイド 第 2 巻』の「データベースオペレーションのカスタマイズ」を参照してください。

ここで説明するシステムファンクションには、「*ファンクション文字列クラススコープ*」がある場合と「*複写定義スコープ*」がある場合があります。

ファンクション文字列クラススコープがあるファンクションは、そのクラスに対して一度定義されます。その後、そのクラスが割り当てられるすべてのデータベースに同じように適用されます。

複写定義スコープがあるファンクションは、複写定義ごとに一度定義されます。その後、その複写定義を使用して複写されるすべてのオペレーション (更新、挿入など) に対して同じように適用されます。

rs_autoc_on

`rs_status` テーブルを更新して、オートコレクションが on に設定されていることを示します。

データサーバインタフェース (DSI) がプライマリデータベースログで **autocorrection on** レコードを検出すると、Replication Server は **rs_autoc_on** を呼び出します。

例

- **例 – rs_iq_function_class** の **rs_autoc_on** ファンクション文字列を作成します。

```
create function string rs_autoc_on
 for rs_iq_function_class
 output language
  'insert into rs_status (schema, tablename, action, starttime,
status) values
    (?rs_repl_objowner!sys?,
```

```
    ?rs_deliver_as_name!sys?,
    "A",
    current timestamp,
    "P");
  commit'
```

使用法

- **rs_autoc_on** 関数には、ファンクション文字列クラススコープがあります。
- インストール時に、Replication Server は初期 **rs_autoc_on** ファンクション文字列を作成します。
- **rs_autoc_on** では、*rs_deliver_as_name* システム定義変数を使用します。これは、オートコレクションの影響を受けるレプリケートデータベースのテーブルを示します。
- **rs_autoc_on** では、*rs_repl_objowner* システム定義変数を使用します。これは、オートコレクションの影響を受けるレプリケートデータベースのテーブルの所有者を示します。所有者が指定されない場合、**rs_repl_objowner** にはシングルスペースが格納されます。

rs_autoc_off

rs_status テーブルを更新して、オートコレクションが off に設定されていることを示します。

プライマリデータベースログで **autocorrection off** レコードを検出すると、Replication Server は **rs_autoc_off** を呼び出します。

例

- **例 – rs_iq_function_class** の **rs_autoc_off** ファンクション文字列を作成します。

```
create function string rs_autoc_off
 for rs_iq_function_class
 output language
  'update rs_status
   set endtime = current timestamp,
   status = "X" where schema = ?rs_repl_objowner!sys?
   and tablename = ?rs_deliver_as_name!sys?
   and action = "A" and endtime is null;
   insert into rs_status (schema, tablename, action, starttime,
status) values
   (?rs_repl_objowner!sys?,
    ?rs_deliver_as_name!sys?,
    "R",
    current timestamp,
    "P");
   commit'
```

使用法

- **rs_autoc_off** 関数には、ファンクション文字列クラススコープがあります。
- インストール時に、Replication Server は初期 **rs_autoc_off** ファンクション文字列を作成します。
- **rs_autoc_off** では、*rs_deliver_as_name* システム定義変数を使用します。これは、オートコレクションの影響を受けるレプリケートデータベースのテーブルを示します。
- **rs_autoc_off** では、*rs_repl_objowner* システム定義変数を使用します。これは、オートコレクションの影響を受けるレプリケートデータベースのテーブルの所有者を示します。所有者が指定されない場合、**rs_repl_objowner** にはシングルスペースが格納されます。

rs_autoc_ignore

`rs_status` テーブルを更新して、オートコレクションに失敗し、テーブルに対する DML が無視されることを示します。

オートコレクション中にプライマリキーが更新されると、Replication Server は **rs_autoc_ignore** を呼び出します。

例

- **例 – rs_iq_function_class** の **rs_autoc_ignore** ファンクション文字列を作成します。

```
create function string rs_autoc_ignore
 for rs_iq_function_class
 output language
  'update rs_status
   set endtime = current timestamp,
   status = 'E' where schema = ?rs_repl_objowner!sys?
   and tablename = ?rs_deliver_as_name!sys?
   and action = 'A' and endtime is null;
   commit'
```

使用法

- **rs_autoc_ignore** 関数には、ファンクション文字列クラススコープがあります。
- インストール時に、Replication Server は初期 **rs_autoc_ignore** ファンクション文字列を作成します。
- **rs_autoc_ignore** では、*rs_deliver_as_name* システム定義変数を使用します。これは、オートコレクションの影響を受けるレプリケートデータベースのテーブルを示します。

- **rs_autoc_ignore** では、*rs_repl_objowner* システム定義変数を使用します。これは、オートコレクションの影響を受けるレプリケートデータベースのテーブルの所有者を示します。所有者が指定されない場合、**rs_repl_objowner** にはシングルスペースが格納されます。

rs_batch_end

rs_batch_end では、コマンドを Adaptive Server 以外のデータベースサーバにバッチ処理できます。このファンクション文字列は、コマンドのバッチの最後をマーク付けするために必要な SQL 文を格納します。

例

- **例 1 – rs_batch_end** ファンクション文字列を変更して、ファンクション文字列クラス **sqlserver_derived_class** の SQL 出力が END になるようにします。

```
alter function string publishers.rs_batch_end
for sqlserver_derived_class
output language
'END'
```

使用法

- **rs_batch_end** ファンクションには、ファンクション文字列クラススコープがあります。
- このファンクション文字列は、**rs_batch_start** とともに使用されます。
- **rs_batch_end** は、コマンドのバッチの最後のコマンドとしてレプリケートデータサーバに送信されます。**use_batch_markers** が on に設定されている場合のみ送信されます。
- **rs_batch_end** は、データサーバの処理順に **rs_commit** の前に付きます。
- **dsi_cmd_batch_size** などの制限によりコマンドがフラッシュされるために複数のバッチが必要な場合には、コマンドのバッチ **rs_batch_start** および **rs_batch_end** を特定のトランザクションに繰り返すことができます。

参照：

- rs_batch_start (537 ページ)

rs_batch_start

rs_batch_start では、コマンドを Adaptive Server 以外のデータベースサーバにバッチ処理できます。このファンクション文字列は、コマンドのバッチの先頭をマーク付けするために必要な SQL 文を格納します。

例

- **例 1 – rs_batch_start** ファンクション文字列を変更して、ファンクション文字列クラス **sqlserver_derived_class** の SQL 出力が BEGIN になるようにします。

```
alter function string publishers.rs_batch_start
for sqlserver_derived_class
output language
'BEGIN'
```

使用法

- **rs_batch_start** ファンクションには、ファンクション文字列クラススコープがあります。
- Adaptive Server や、ファンクション文字列 **rs_begin** および **rs_commit** によってコマンドのバッチ処理がサポートされているその他のデータサーバでは、**rs_batch_start** を使用する必要はありません。
- **use_batch_markers** が on に設定されている場合にのみ、**rs_batch_start** とその後に続くコマンドのバッチがレプリケートデータサーバに送信されます。**rs_batch_start** は **rs_begin** の後で送信されます。
- Replication Server は **rs_batch_start** の後にコマンドセパレータを使用しません。レプリケートデータベースサーバでバッチの先頭のマーカの後にコマンドセパレータが必要な場合は、**rs_batch_start** の文字列の一部として含まれます。**dsi_cmd_separator** パラメータと同じ場合でも異なる場合でも、このセパレータがファンクション文字列の一部として含まれている必要があります。
- **dsi_cmd_batch_size** などの制限によりコマンドがフラッシュされるために複数のバッチが必要な場合には、**rs_batch_start**、コマンドのバッチ、および **rs_batch_end** を繰り返すことができます。

参照：

- rs_batch_end (536 ページ)

rs_begin

データサーバでトランザクションを開始します。

例

- **例 1 –** *oth_sql_class* ファンクション文字列クラスの **rs_begin** ファンクション文字列を作成します。トランザクションに名前がない場合は、*rs_origin_xact_name* システム変数の値は null になります。システム変数の前に “t_” を指定すると、データサーバの構文エラーを防ぐことができ、名前のあるトランザクションもないトランザクションもファンクション文字列で使用できます。

```
alter function string rs_begin
 for oth_sql_class
 output language
 'begin transaction
   t_?rs_origin_xact_name!sys_raw?'
```

- **例 2 – begin transaction** オペレーションをサポートしないデータサーバ用のファンクション文字列クラスに **rs_begin** ファンクション文字列を作成します。

```
create function string rs_begin
 for oth_sql_class
 output language ''
```

使用法

- **rs_begin** ファンクションには、ファンクション文字列クラススコープがあります。
- インストール中、Replication Server はシステム提供ファンクション文字列クラスの初期 **rs_begin** ファンクション文字列を作成します。
- ユーザが作成した基本ファンクション文字列クラスを使用する場合は、**rs_begin** ファンクション文字列を自分で作成してください。
- **rs_begin** ファンクション文字列の作成またはカスタマイズは、そのクラスのプライマリサイトである Replication Server で行ってください。
- データサーバによっては、明示的な **begin transaction** オペレーションをサポートしないものがあります。これらのサーバでは、前のトランザクションがコミットまたはロールバックされたときに、暗黙的にトランザクションが開始されます。このようなデータサーバでは、**rs_begin** ファンクション文字列は空の文字列 (") となります。
- このファンクションのファンクション文字列は通常、*rs_origin_xact_name* システム変数を使用します。この変数の値は RepAgent から受け取ります。トラン

ザクション名は、**begin transaction** を使用して Transact-SQL で割り当てられます。

参照：

- alter function string (188 ページ)
- create function string (318 ページ)
- rs_commit (540 ページ)
- rs_rollback (570 ページ)

rs_check_repl

テーブルが複写対象としてマーク付けされているかどうかを確認します。

例

- **例 1 – rs_check_repl_stat** ストアドプロシージャを実行する **rs_check_repl** ファンクション文字列を作成します。

```
create function string rs_check_repl
 for sqlserver_derived_class
output language
'execute rs_check_repl_stat
 @rs_repl_name = ?rs_repl_name!param?'
```

使用法

- **rs_check_repl** ファンクションには、ファンクション文字列クラススコープがあります。
- インストール中、Replication Server はシステム提供ファンクション文字列クラスの初期 **rs_check_repl** ファンクション文字列を作成します。
- ユーザが作成した基本ファンクション文字列クラスを使用する場合は、**rs_check_repl** ファンクション文字列を自分で作成してください。
- **rs_check_repl** ファンクション文字列の作成またはカスタマイズは、そのクラスのプライマリサイトである Replication Server で行ってください。

参照：

- create function string (318 ページ)
- create replication definition (346 ページ)

rs_commit

データサーバでトランザクションをコミットします。

例

- **例 1 –** これは、*rs_sqlserver_function_class* クラスと *rs_default_function_class* クラスに対するデフォルトの **rs_commit** ファンクション文字列の例です。このファンクション文字列は、まず **rs_update_lastcommit** という名前のストアドプロシージャを実行し、次に Transact-SQL の **commit transaction** コマンドを実行します。

```
create function string rs_commit
 for sqlserver_derived_class
 output language
 'execute rs_update_lastcommit
 @origin = ?rs_origin!sys?,
 @origin_qid = ?rs_origin_qid!sys?,
 @secondary_qid = ?rs_secondary_qid!sys?,
 @origin_time = ?rs_origin_commit_time!sys?;
 commit transaction'
```

以下は、*rs_sqlserver_function_class* に対する **rs_update_lastcommit** プロシージャの内容です。

```
/* Create a procedure to update the
 ** rs_lastcommit table. */
 create procedure rs_update_lastcommit
 @origininint,
 @origin_qidbinary(36),
 @secondary_qidbinary(36),
 @origin_timedatetime
 as
 begin
 update rs_lastcommit
 set origin_qid = @origin_qid,
 secondary_qid = @secondary_qid,
 origin_time = @origin_time,
 commit_time = getdate()
 where origin = @origin
 if (@@rowcount = 0)
 begin
 insert rs_lastcommit (origin,
 origin_qid, secondary_qid,
 origin_time, commit_time,
 pad1, pad2, pad3, pad4,
 pad5, pad6, pad7, pad8)
 values (@origin, @origin_qid,
 @secondary_qid,@origin_time,
 getdate(), 0x00, 0x00, 0x00,
 0x00, 0x00, 0x00, 0x00, 0x00)
```

```
end
end
```

使用法

- **rs_commit** ファンクションには、ファンクション文字列クラススコープがあります。
- インストール中、Replication Server はシステム提供ファンクション文字列クラスの初期 **rs_commit** ファンクション文字列を作成します。
- ユーザが作成した基本ファンクション文字列クラスを使用する場合は、**rs_commit** ファンクション文字列を自分で作成してください。
- **rs_commit** ファンクション文字列の作成またはカスタマイズは、そのクラスのプライマリサイトである Replication Server で行ってください。
- **rs_commit** ファンクション文字列で *rs_lastcommit* システムテーブルを更新してください。トランザクション内でこのテーブルを更新することで、データの整合性が維持されます。

警告！ トランザクションがコミットされるたびに *rs_lastcommit* システムテーブルが適切に更新されないと、Replication Server の再起動後にトランザクションが何度も適用されたり、またはスキップされることがあります。

参照：

- alter function string (188 ページ)
- create function string (318 ページ)
- rs_begin (538 ページ)
- rs_get_lastcommit (556 ページ)
- rs_rollback (570 ページ)

rs_datarow_for_writetext

Transact-SQL の **writetext** コマンド、Client-Library 関数の **ct_send_data**、または DB-Library™ 関数の **dbwritetext** と **dbmoretext** で更新される、*text*、*unitext*、または *image* カラムに関連付けられているデータローのイメージを提供します。

例

- **例 1 – capture_datarow** という名前のストアドプロシージャを実行して、*@au_id* の値を *au_id* カラムの値に、*@copy* の値を *copy* カラムのステータス値にそれぞれ設定します。

```
create function string
 blurbs_rep.rs_datarow_for_writetext
```

```
for sqlserver_derived_class
output rpc
'execute capture_datarow
@au_id = ?au_id!new?,
@copy = ?copy!text_status?'
```

使用法

- Replication Server は、更新された *text*、*unitext*、または *image* データをレプリケートデータサーバに送信する前に、**rs_datarow_for_writetext** を実行します。**rs_datarow_for_writetext** は、プライマリキーカラムとサーチャブルカラムの値をローから提供するため、サブスクリプションを処理してデータをレプリケートデータベースに転送できるようになります。
- **rs_datarow_for_writetext** は、*text*、*unitext*、*image* カラムを除く、ローにあるすべてのカラムの値にアクセスします。*text*、*unitext*、または *image* カラムについての情報を取得するには、ファンクション文字列に *text_status* 変更子を指定します。*text_status* によって返される値については、「text、unitext、image データの text_status 値」表で説明しています。
- **rs_datarow_for_writetext** ファンクションには、複写定義スコープがあります。
- 複写定義を作成すると、Replication Server は、*rs_sqlserver_function_class* クラスと *rs_default_function_class* クラスに **rs_datarow_for_writetext** ファンクション文字列を生成します。
- ユーザが作成した基本ファンクション文字列クラスを使用する場合は、*text*、*unitext*、*image* カラムを含む複写定義を作成するたびに **rs_datarow_for_writetext** ファンクション文字列を自分で作成してください。
- **rs_datarow_for_writetext** ファンクション文字列の作成またはカスタマイズは、複写定義を作成した Replication Server で行ってください。
- ローイメージには修正データは含まれないため、*rs_sqlserver_function_class* クラスと *rs_default_function_class* クラスに対してデフォルトで生成されるファンクション文字列は、レプリケートデータベースではコマンドを実行しません。
- ゲートウェイに渡すプライマリキーの値を集めるために、新しい **rs_datarow_for_writetext** ファンクション文字列を作成できます。*old* および *new* の 2 つの変更子は、どちらもカラムの値へのアクセスを提供します。
- *text_status* 変更子は、*text*、*unitext*、または *image* カラムのステータスを取得します。表 37 : text、unitext、image データの text_status 値 (543 ページ)に、*text_status* 変更子の値を示します。

表 37：text、unitext、image データの text_status 値

値	説明
0x0001	カラムには null テキストポインタがある。*text*、*unitext*、または *image* カラムへの修正はない。
0x0002	プライマリデータベースで修正が行われ、テキストポインタが割り付けられた。Replication Server は、テキストポインタを割り付けるために **rs_textptr_init** ファンクションを実行する。
0x0004	現在のデータの値が後に続く。Replication Server は、レプリケートデータベースで *text*、*unitext*、または *image* データを変更するために、**rs_writetext** ファンクションを実行する。
0x0008	*text*、*unitext*、または *image* カラムは複写されていない。データの値は変更されておらず、*text*、*unitext*、または *image* カラムのステータスは **replicate_if_changed** となっているため、レプリケートデータベースで実行する必要のあるコマンドはない。
0x0010	プライマリデータベースでオペレーションが実行された後、*text*、*unitext*、または *image* カラムに null 値が設定されている。たとえば、テキストポインタが割り付けられた後、*text* カラムと *image* カラムのデータ値がある場合は、プライマリデータベースのアプリケーションがそれらの値を null に設定する。*text_status* が 0x0008 以外の場合、Replication Server は *text*、*unitext*、または *image* カラムの値を null に設定することで、これらのカラムをトランケートする。

参照：

- rs_get_textptr (558 ページ)
- rs_textptr_init (582 ページ)
- rs_writetext (592 ページ)

rs_ddlsession_setting

Adaptive Server の事前計算済み結果セット DDL コマンドの複写をサポートするように、Adaptive Server の複数のセッションレベル **set** オプションを自動的に適用します。

例

- **例 1**

  rs_ddlsession_setting ファンクション文字列のインスタンスを作成します。

```
create function string rs_ddlsession_setting
 for sqlserver_derived_class
```

```
 output language
 'set ansinull on; set arithabort on; set arighignore off; set
string_rtruncation on'
```

使用法

- Replication Server が、データベースへの DSI コネクションに対して **rs_ddlsession_setting** を実行します。
- **rs_ddlsession_setting** を手動で実行する必要はありません。
- データサーバインタフェース (DSI) スレッドが DDL コマンドを適用する前に、Replication Server が **rs_ddlsession_setting** を実行します。
- **rs_ddlsession_setting** は事前計算済み結果セット DDL コマンドの複写を行えるようにするため、Adaptive Server のセッションレベル **set** オプションを関連した値に設定します。
  - **set ansinull on**
  - **set arithabort on**
  - **set arithignore off**
  - **set string_rtruncation on**
- **rs_ddlsession_setting** には、ファンクション文字列クラススコープがあります。
- インストール中、Replication Server は、システム提供ファンクション文字列クラスの初期 **rs_ddlsession_setting** ファンクション文字列を作成します。
- Replication Server は **rs_ddlsession_setting** を **rs_ddlsession_resetting** と共に使用します。
- ユーザが作成した基本ファンクション文字列クラスを、デフォルト以外の方法で使用する場合には、**rs_ddlsession_setting** ファンクション文字列を作成してください。
- **rs_ddlsession_setting** ファンクション文字列の作成またはカスタマイズは、そのクラスのプライマリサイトである Replication Server で行ってください。

rs_ddlsession_resetting

Adaptive Server DDL コマンドの複写後に、Adaptive Server の複数のセッションレベル **set** オプションを自動でデフォルト値にリセットします。

例

- **例 1**

  rs_ddlsession_resetting ファンクション文字列のインスタンスを作成します。

```
create function string rs_ddlsession_resetting
 for sqlserver_derived_class
 output language
```

```
 'set ansinull off; set arithabort on; set arighignore off; set
string_rtruncation off'
```

使用法

- Replication Server が、データベースへの DSI コネクションに対して **rs_ddlsession_resetting** を実行します。
- **rs_ddlsession_resetting** を手動で実行する必要はありません。
- データサーバインタフェース (DSI) スレッドが DDL コマンドの処理を完了すると、Replication Server が **rs_ddlsession_resetting** を呼び出します。
- **rs_ddlsession_resetting** は、DDL コマンドの処理後にセッションレベルのオプションをデフォルト値にリセットします。
  - **set ansinull off**
  - **set arithabort on**
  - **set arithignore off**
  - **set string_rtruncation off**
- **rs_ddlsession_resetting** には、ファンクション文字列クラススコープがあります。
- インストール中、Replication Server は、システム提供ファンクション文字列クラスの初期 **rs_ddlsession_resetting** ファンクション文字列を作成します。
- Replication Server は **rs_ddlsession_resetting** を **rs_ddlsession_setting** と共に使用します。
- ユーザが作成した基本ファンクション文字列クラスを、デフォルト以外の方法で使用する場合には、**rs_ddlsession_resetting** ファンクション文字列を作成してください。 たとえば、**rs_ddlsession_resetting** に対して、セッションレベルの **set** オプションの値が前に設定した値に戻されるようにするカスタムファンクション文字列を作成できます。
- **rs_ddlsession_resetting** ファンクション文字列の作成またはカスタマイズは、そのクラスのプライマリサイトである Replication Server で行ってください。

rs_delete

複写テーブルにあるローを削除します。

例

- **例 1 –** *titles_rep* 複写定義の **rs_delete** ファンクション文字列を変更して、**del_title** という名前のストアドプロシージャを実行するようにします。

```
alter function string titles_rep.rs_delete
 for sqlserver_derived_class
 output rpc
```

```
'execute del_title
  @title=?title!old?'
```

使用法

- Replication Server は、**rs_delete** を実行してテーブル内の 1 つのローを削除します。ローは、テーブルの複写定義で定義されているプライマリキーカラムによって識別されます。
- **rs_delete** には、複写定義スコープがあります。
- 複写定義を作成すると、Replication Server は、システム提供ファンクション文字列クラスに対して **rs_delete** ファンクション文字列を生成します。
- ユーザが作成した基本ファンクション文字列クラスを使用する場合、複写定義を作成するたびに **rs_delete** ファンクション文字列を自分で作成してください。
- **rs_delete** ファンクション文字列の作成またはカスタマイズは、複写定義を作成した Replication Server で行ってください。
- システム提供クラス *rs_sqlserver_function_class* と *rs_default_function_class* に対して生成される **rs_delete** ファンクション文字列では、Transact-SQL の **delete** コマンドの構文が使用されます。削除するローは、プライマリキーカラムの削除前の値、つまり削除前のイメージを指定する **where** 句で識別されます。

参照：

- create function string (318 ページ)
- create replication definition (346 ページ)
- rs_insert (563 ページ)
- rs_update (588 ページ)

rs_dsi_check_thread_lock

DSI エグゼキュータスレッドがレプリケートデータベースプロセスをブロックするロックを保持しているかどうかを判別します。戻り値が 0 より大きい場合、別のデータベースプロセスに必要なリソースをスレッドが保持しており、スレッドがトランザクションをロールバックして再試行する必要があることを示します。

例

- **例 1 –** 現在の DSI エグゼキュータスレッドが別のレプリケートデータベースのプロセスをブロックしているかどうかをチェックする **rs_dsi_check_thread_lock** ファンクション文字列を作成します。

```
create function string rs_dsi_check_thread_lock
for sqlserver_derived_class
output language
```

```
'select count(*) as seq from master..sysprocesses
where blocked = @@spid and suid = suser_id()'
```

使用法

- Replication Server は、**rs_dsi_check_thread_lock** ファンクションを使用して、現在の DSI エグゼキュータスレッドが別のレプリケートデータベースのプロセスをブロックしているかどうかをチェックします。これは、**dsi_commit_control** が on に設定されているコネクションに複数の DSI スレッドが定義され、DSI エグゼキュータスレッドのコミット準備が完了しても、コミットされるべき “next” のスレッドではないのでコミットできず、**dsi_commit_check_locks_intrvl** に定義された時間が経過した場合にのみ実行されます。
- ファンクション文字列 **rs_dsi_check_thread_lock** クエリは、*seq* の単一整数値のカラム名を返すことが予想されます。戻り値が 0 より大きい場合、別のデータベースプロセスに必要なリソースをスレッドが保持しており、スレッドがトランザクションをロールバックして再試行する必要があることを示す。
- **rs_dsi_check_thread_lock** ファンクションには、ファンクション文字列クラススコープがあります。
- インストール中、Replication Server はシステム提供ファンクション文字列クラスの初期 **rs_dsi_check_thread_lock** ファンクション文字列を作成します。
- カスタムベースファンクション文字列を使用しており、**dsi_commit_control** を on に設定した並列 DSI を使用する場合は、**rs_dsi_check_thread_lock** ファンクション文字列のファンクション文字列を作成する必要があります。それ以外の場合は、このファンクションに対してファンクション文字列を作成する必要はありません。
- **rs_dsi_check_thread_lock** ファンクション文字列の作成またはカスタマイズは、そのクラスのプライマリサイトで配備された Replication Server で行ってください。

rs_dumpdb

コーディネートデータベースダンプを開始します。

例

- **例 1 –** 指定したダンプデバイスにデータベースをダンプする **rs_dumpdb** ファンクション文字列を作成し、*rs_lastcommit* システムテーブルを更新するプロシージャを実行します。このファンクション文字列は、レプリケートデータベースが 1 つしかないとき、またはこのファンクション文字列クラスを使用するすべてのデータベースでダンプデバイス名が同じときに最も有効です。

```
create function string rs_dumpdb
 for sqlserver_derived_class
```

```
output language
'dump database ?rs_destination_db!sys_raw?
to pubs2_dmpdb;
execute rs_update_lastcommit
?rs_origin!sys?,
?rs_origin_qid!sys?,
?rs_secondary_qid!sys?,
?rs_origin_commit_time!sys?'
```

- **例 2 –** この例は、最初の例とは違って、複数のサイトと運用環境に適しています。**dumpdb_proc** は、レプリケートサイトのバックアップデバイスを管理します。このプロシージャは、使用するバックアップデバイスを選択し、その後のダンプで前のバックアップが上書きされないように “used” とマーク付けします。

```
alter function string rs_dumpdb
 for sqlserver_derived_class
 output rpc
 'execute dumpdb_proc
 ?rs_dump_dbname!sys?,
 ?rs_dump_label!sys?,
 ?rs_dump_timestamp!sys?,
 ?rs_destination_db!sys?,
 ?rs_origin!sys?,
 ?rs_origin_qid!sys?,
 ?rs_secondary_qid!sys?,
 ?rs_origin_commit_time!sys?'
```

プロシージャは、*rs_origin*、*rs_origin_qid*、*rs_secondary_qid* を使用して、**rs_update_lastcommit** を実行します。ダンプが完了した後、*rs_lastcommit* システムテーブルを更新する前にサーバが何らかの処理に失敗した場合、Replication Server はレジューム時にバックアップを再起動します。

注意： ダンプと **rs_update_lastcommit** プロシージャがアトミックに実行される保証はありません。これは、Adaptive Server では **dump** コマンドを他のコマンドと一緒にトランザクションに指定できないためです。*rs_lastcommit* システムテーブルが正常に更新されなかったときは、さらにダンプが必要な場合があります。

次に示す **dumpdb_proc** ストアドプロシージャの例では、ダンプデバイスがハードコードされています。実際の運用では、テーブルを参照することをおすすめします。

```
create proc dumpdb_proc
 @dump_dbname varchar(30),
 @dump_label varchar(30),
 @dump_timestamp varbinary(16),
 @destination_dbname varchar(30),
 @origin int,
 @origin_qid binary(36),
 @secondary_qid binary(36),
 @origin_time datetime
```

```
as
print 'Received a dump database command from Replication Server:'
declare @message varchar(255)
select @message = 'dump database ' + @dump_dbname
+ '. Label= '' + @dump_label
+ ''. Dest.db = '' + @destination_dbname
+ ''''
print @message
if @destination_dbname = 'pubs2'
begin
print 'issuing ''dump database pubs2.'''
dump database pubs2 to pubs2_dmplog
update dmp_count set d_count = d_count + 1
exec pubs2.dbo.rs_update_lastcommit
@origin, @origin_qid, @secondary_qid,
@origin_time
end
else if @destination_dbname = 'pubs3'
begin
print 'issuing ''dump database pubs3.'''
dump database pubs3 to pubs3_dmplog
update dmp_count set d_count = d_count + 1
exec pubs3.dbo.rs_update_lastcommit
@origin, @origin_qid, @secondary_qid,
@origin_time
end
```

使用法

- Replication Server は、各レプリケート Replication Server に分配されるトランザクションのストリーム内の同じ場所に **rs_dumpdb** ファンクション呼び出しを置くことによって、データベースダンプを調整します。
- **rs_dumpdb** には、ファンクション文字列クラススコープがあります。

  注意： Replication Server は、システム提供ファンクション文字列クラスに対する **rs_dumpdb** ファンクション文字列の初期化や生成は行いません。ファンクション文字列を作成してから、Adaptive Server でコーディネートダンプを使用してください。
- **rs_dumpdb** ファンクション文字列の作成は、そのクラスのプライマリサイトである Replication Server で行ってください。
- 複数のレプリケートサイトで異なるダンプデバイスを使用する場合は、データベースダンプを実行する各レプリケートデータベースにストアドプロシージャを作成します。そのうえで、そのストアドプロシージャを実行する **rs_dumpdb** ファンクション文字列を記述してください。
- Replication Server が再起動したときにダンプが重複して行われないよう、**rs_dumpdb** ファンクション文字列の実行時には *rs_lastcommit* システムテーブルを更新してください。*rs_lastcommit* の詳細については、「**rs_commit**」を参照してください。

表 38 : rs_dumpdb ファンクション文字列のシステム変数

変数名	データ型	説明
rs_dump_dbname	*varchar(30)*	ダンプを作成するデータベースの名前。
rs_dump_label	*varchar(30)*	ダンプのラベル情報。Adaptive Server では、この変数にはダンプが開始された時間を示す *datetime* 値が格納される。
rs_dump_timestamp	*varbinary(16)*	ダンプの開始時に取られたタイムスタンプ。

参照：

- create function string class (334 ページ)
- rs_commit (540 ページ)
- rs_dumptran (550 ページ)
- rs_get_lastcommit (556 ページ)

rs_dumptran

コーディネートトランザクションダンプを開始します。

例

- **例 1 – dumptran_proc** という名前のストアドプロシージャを実行する **rs_dumptran** ファンクション文字列を作成します。このストアドプロシージャはダンプデバイスを管理し、次に *rs_origin*、*rs_origin_qid*、*-rs_secondary_qid*、*rs_origin_commit_time* パラメータを渡して **rs_update_lastcommit** ストアドプロシージャを実行します。

```
create function string rs_dumptran
 for sqlserver_derived_class
 output rpc
 'execute dumptran_proc
 ?rs_dump_dbname!sys?,
 ?rs_dump_label!sys?,
 ?rs_dump_timestamp!sys?,
 ?rs_dump_status!sys?,
 ?rs_destination_db!sys?,
 ?rs_origin!sys?,
 ?rs_origin_qid!sys?,
 ?rs_secondary_qid!sys?
 ?rs_origin_commit_time!sys?'
```

ダンプが完了した後、*rs_lastcommit* システムテーブルを更新する前にサーバがクラッシュした場合、Replication Server はバックアップを再起動します。

注意： ダンプと **rs_update_lastcommit** プロシージャがアトミックに実行される保証はありません。これは、Adaptive Server では **dump** コマンドを他のコマン

ドと一緒にトランザクションに指定できないためです。*rs_lastcommit* システムテーブルが正常に更新されなかったときは、さらにダンプが必要な場合があります。

次に示す **dumptran_proc** ストアドプロシージャの例では、ダンプデバイスがハードコードされています。実際の運用では、テーブルを参照することをおすすめします。

```
create proc dumptran_proc
@dump_dbname varchar(30),
@dump_label varchar(30),
@dump_timestamp varbinary(16),
@dump_status int,
@destination_dbname varchar(30),
@origin int,
@origin_qid binary(36),
@secondary_qid binary(36),
@origin_time datetime
as
print 'Received a dump transaction command from Replication
Server:'
declare @message varchar(255)
if @dump_status = 0
begin
select @message = 'dump transaction ' + @dump_dbname + '. Label=
'''
+ @dump_label + '''' + '. Dest.db = ''' + @destination_dbname +
''''
end
else if @dump_status = 1
begin
select @message = 'dump transaction standby '
+ @dump_dbname + '. Label= ''' +
@dump_label + '''' + '. Dest.db = ''' + @destination_dbname + ''''
end
print @message
if @destination_dbname = 'pubs2'
begin
print 'issuing ''dump transaction pubs2.'''
if @dump_status = 0
begin
dump transaction pubs2 to pubs2_dmplog
end
else if @dump_status = 1
begin
dump transaction pubs2 to pubs2_dmplog with standby_access
end
update dmp_count set d_count = d_count + 1
exec pubs2.dbo.rs_update_lastcommit
@origin, @origin_qid, @secondary_qid,
@origin_time
end
else if @destination_dbname = 'pubs3'
begin
print 'issuing ''dump transaction pubs3.'''
```

```
if @dump_status = 0
begin
dump transaction pubs3 to pubs3_dmplog
end
else if @dump_status = 1
begin
dump transaction pubs3 to pubs3_dmplog with standby_access
end
update dmp_count set d_count = d_count + 1
exec pubs3.dbo.rs_update_lastcommit
@origin, @origin_qid, @secondary_qid,
@origin_time
end
```

- **例 2 –** 最初の例で作成した **rs_dumptran** ファンクション文字列を変更し、リモートプロシージャコールとして実行されるようにします。

```
alter function string rs_dumptran
 for sqlserver_derived_class
 output rpc
 'execute dumptran_proc
 ?rs_dump_dbname!sys?,
 ?rs_dump_label!sys?,
 ?rs_dump_timestamp!sys?,
 ?rs_dump_status!sys?,
 ?rs_destination_db!sys?,
 ?rs_origin!sys?,
 ?rs_origin_qid!sys?,
 ?rs_secondary_qid!sys?,
 ?rs_origin_commit_time!sys?!'
```

使用法

- Replication Server は、すべてのレプリケート Replication Server に分配するトランザクションのストリーム内の同じ場所に **rs_dumptran** ファンクション呼び出しを挿入することによって、トランザクションダンプを調整します。
- **rs_dumptran** には、ファンクション文字列クラススコープがあります。

注意： Replication Server は、システム提供ファンクション文字列クラスに対する **rs_dumptran** ファンクション文字列の初期化や生成は行いません。ファンクション文字列を作成してから、Adaptive Server でコーディネートダンプを使用してください。

- **rs_dumptran** ファンクション文字列の作成は、そのクラスのプライマリサイトである Replication Server で行ってください。
- Replication Server が再起動したときにダンプが重複して行われないよう、**rs_dumptran** ファンクション文字列の実行時には *rs_lastcommit* システムテーブルを更新してください。*rs_lastcommit* の詳細については、「**rs_commit**」を参照してください。
- 複数のレプリケートサイトで異なるダンプデバイスを使用する場合は、トランザクションダンプを実行する各レプリケートデータベースにストアドプロシー

ジャを作成します。そのうえで、そのストアドプロシージャを実行する **rs_dumptran** ファンクション文字列を記述してください。

表 39 : rs_dumptran ファンクション文字列のシステム変数

変数名	データ型	説明
rs_destination_db	*varchar(30)*	トランザクションが送信されたデータベースの名前。
rs_dump_dbname	*varchar(30)*	ダンプを作成するデータベースの名前。
rs_dump_label	*varchar(30)*	ダンプのラベル情報。Adaptive Server では、この変数にはダンプが開始された時間を示す *datetime* 値が格納される。
rs_dump_status	*int(4)*	ダンプステータスインジケータ： • 0 - ダンプトランザクションコマンドに **with standby_access** パラメータが含まれないことを示す。 • 1 - ダンプトランザクションコマンドに **with standby_access** パラメータが含まれることを示す。
rs_dump_timestamp	*varbinary(16)*	オリジンでダンプが開始されたときに取られた、Adaptive Server データベースのタイムスタンプ。この変数は、情報を提供するためだけに使用される。
rs_origin	*int(4)*	トランザクションが開始されるデータベースの ID。
rs_origin_commit_time	*datetime*	オリジンでトランザクションがコミットされた時間。 **注意：** ASE がまだユーザデータベースのリカバリを処理している間に **select getdate()** を実行した場合、**select getdate()** の戻り値が、**rs_origin_begin_time** の値と異なる可能性があります。
rs_origin_qid	*varbinary(36)*	トランザクションにある最初のコマンドのオリジンキュー ID。
rs_secondary_qid	*varbinary(36)*	サブスクリプションマテリアライゼーションキューまたはマテリアライゼーション解除キューにあるトランザクションのキュー ID。

参照：

- create function string (318 ページ)
- rs_commit (540 ページ)
- rs_dumpdb (547 ページ)
- rs_get_lastcommit (556 ページ)

rs_get_charset

データサーバで使用している文字セットを返します。このファンクションを使用すると、文字セットが予期していたものと違った場合に Replication Server で警告メッセージを出力できます。

例

- **例 1 – sp_serverinfo** システムプロシージャを呼び出してデータサーバが使用している文字セットを返す出力言語で、**rs_get_charset** ファンクション文字列を作成します。

```
create function string rs_get_charset
 for rs_sqlserver2_function_class
 output language
 'sp_serverinfo server_csname'
```

使用法

- **rs_get_charset** は、データサーバが使用している文字セットの名前を取得します。Replication Server は、データサーバへ接続するたびに、このファンクションを実行します。
- **rs_get_charset** ファンクションには、ファンクション文字列クラススコープがあります。
- インストール中、Replication Server はシステム提供ファンクション文字列クラスの初期 **rs_get_charset** ファンクション文字列を作成します。
- ユーザが作成した基本ファンクション文字列クラスを使用する場合は、**rs_get_charset** ファンクション文字列を自分で作成してください。
- **rs_get_charset** ファンクション文字列の作成またはカスタマイズは、そのクラスのプライマリサイトである Replication Server で行ってください。
- *rs_sqlserver_function_class* クラスと *rs_default_function_class* クラスに対するデフォルトの **rs_get_charset** ファンクション文字列は、Adaptive Server のストアドプロシージャ **sp_serverinfo** を引数 *server_csname* を使用して呼び出します。
- データサーバは、SAP がサポートしている有効な文字セットの名前の文字列を返されなければなりません。 有効な SAP の文字セットは、リリースディレクトリの `charsets/charset_name/charset.loc` に定義されています。*charset_name* はサポートされている文字セットの名前です。 たとえば、`charsets/iso_1/charset.loc` ファイルには iso_1 文字セットが定義されています。

参照：

- create function string (318 ページ)
- rs_get_sortorder (557 ページ)

rs_get_errormode

ネイティブエラーがレプリケートサーバから直接返されているかどうかを判断するネイティブエラー設定を返します。

例

- **例 1 –** ネイティブエラーを返す **oth_sql_class** ファンクション文字列クラスの rs_get_errormode ファンクション文字列を作成します。

```
create function string rs_get_errormode
for oth_sql_class
output language 'select yes'
```

- **例 2 –** ネイティブエラーを返さない **oth_sql_class** ファンクション文字列クラスの rs_get_errormode ファンクション文字列を作成します。

```
create function string rs_get_errormode
for oth_sql_class
output language 'select no'
```

使用法

- **rs_get_errormode** ファンクションには、ファンクション文字列クラススコープがあります。
- インストール中、Replication Server はシステム提供ファンクション文字列クラスの初期 **rs_get_errormode** ファンクション文字列を作成します。
- ユーザが作成した基本ファンクション文字列クラスを使用する場合は、rs_get_errormode ファンクション文字列を自分で作成してください。
- rs_get_errormode ファンクション文字列の作成またはカスタマイズは、そのクラスのプライマリサイトである Replication Server で行ってください。
- ファンクション **rs_get_errormode** に予想される結果は `yes` または `no` です。

rs_get_lastcommit

rs_lastcommit システムテーブルからローを返します。

例

- **例 1 – rs_get_lastcommit** という名前のストアドプロシージャを実行する **rs_get_lastcommit** ファンクション文字列を作成します。ストアドプロシージャの内容を次に示します。

```
create procedure rs_get_lastcommit
 as
 select origin, origin_qid, secondary_qid
   from rs_lastcommit
```

```
create function string rs_get_lastcommit
 for sqlserver_derived_class
 output language
 'execute rs_get_lastcommit'
```

使用法

- Replication Server は、データベースに対する DSI プロセスを起動するときに **rs_get_lastcommit** を実行します。このファンクションは、*rs_lastcommit* システムテーブル内のすべてのローを返します。Replication Server はこの情報を使用して、各プライマリデータの送信元から最後にコミットされたトランザクションを検索します。
- *rs_lastcommit* システムテーブルは、Replication Server がデータベースでトランザクションをコミットするたびに更新されます。
- **rs_get_lastcommit** には、ファンクション文字列クラススコープがあります。
- インストール中、Replication Server はシステム提供ファンクション文字列クラスの初期 **rs_get_lastcommit** ファンクション文字列を作成します。
- ユーザが作成した基本ファンクション文字列クラスを使用する場合は、**rs_get_lastcommit** ファンクション文字列を自分で作成してください。
- **rs_get_lastcommit** ファンクション文字列の作成またはカスタマイズは、そのクラスのプライマリサイトである Replication Server で行ってください。
- *rs_sqlserver_function_class* クラスと *rs_default_function_class* クラスに対するデフォルトの **rs_get_lastcommit** ファンクション文字列は、**rs_update_lastcommit** という名前のストアドプロシージャを **rs_commit** ファンクション文字列で実行して、*rs_lastcommit* テーブルを更新します。
- データベースに複写されるデータを持つ各プライマリデータベースに対して、**rs_get_lastcommit** は、正しい順番でカラムを返す必要があります。

表 40 : rs_get_lastcommit で返されるカラム

カラム名	データ型	説明
origin	*int*	各ローに該当するプライマリデータベースの ID 番号。
origin_qid	*binary(36)*	オリジンデータベースのステーブルキュー内にある最後にコミットされたトランザクションを示す。
secondary_qid	*binary(36)*	サブスクリプションマテリアライゼーションキューがオリジンデータベースに存在する場合、このカラムには、そのキューにあるレプリケートデータベースにコミットされた最新のトランザクションが含まれる。

参照：

- create function string (318 ページ)
- rs_commit (540 ページ)

rs_get_sortorder

データサーバが使用しているソート順を取得します。ソート順が Replication Server のソート順と一致しなかった場合、または予期したソート順ではなかった場合、警告メッセージを返します。

例

- **例 1 – sp_serverinfo** システムプロシージャを呼び出してデータサーバが使用しているソート順を返す出力言語で、**rs_get_sortorder** ファンクション文字列を作成します。

```
create function string rs_get_sortorder
 for rs_sqlserver2_function_class
 output language
 'sp_serverinfo server_soname'
```

使用法

- **rs_get_sortorder** ファンクションは、データサーバで使用されているソート順の名前を取得します。Replication Server は、データサーバへ接続するたびに、このファンクションを実行します。取得されたソート順が Replication Server のソート順と一致しなかった場合、Replication Server のエラーログに警告メッセージが出力されます。ソート順が一致した場合、警告メッセージは出力されません。
- **rs_get_sortorder** ファンクションには、ファンクション文字列クラススコープがあります。

- インストール中、Replication Server はシステム提供ファンクション文字列クラスの初期 **rs_get_sortorder** ファンクション文字列を作成します。
- ユーザが作成した基本ファンクション文字列クラスを使用する場合は、**rs_get_sortorder** ファンクション文字列を自分で作成してください。
- **rs_get_sortorder** ファンクション文字列の作成またはカスタマイズは、そのクラスのプライマリサイトである Replication Server で行ってください。
- *rs_sqlserver_function_class* クラスと *rs_default_function_class* クラスに対するデフォルトの **rs_get_sortorder** ファンクション文字列は、Adaptive Server のストアドプロシージャ **sp_serverinfo** を引数 *server_soname* を使用して呼び出します。
- **rs_get_sortorder** ファンクション文字列は、SAP がサポートする有効なソート順の名前の文字列を返す必要があります。 文字セットごとの SAP の有効なソート順は、リリースディレクトリの `charsets/charset_name/sortorder.srt` に定義されています。*charset_name* はサポートされる文字セットの名前、*sortorder* はその文字セットに対してサポートされるソート順の名前です。 たとえば、`charsets/iso_1/nocase.srt` ファイルには iso_1 文字セットに対する “nocase” (大文字と小文字を区別しない) ソート順が定義されています。

参照：

- create function string (318 ページ)
- rs_get_charset (554 ページ)

rs_get_textptr

text、*unitext*、または *image* カラムの記述を取得します。

例

- **例 1 –** *blurbs* テーブル内の *repcopy* カラムに対する **rs_get_textptr** ファンクション文字列を作成します。ファンクション文字列名の **copy** は、複写定義の *text*、*unitext*、または *image* カラムの名前です。

```
create function string
 blurbs_rep.rs_get_textptr;copy
 for sqlserver2_function_class
 output language
 'select repcopy from blurbs
 where au_id = ?au_id!new?'
```

使用法

- Replication Server は、Client-Library 関数 **ct_send_data** でデータを送信する前に、**rs_get_textptr** を呼び出して、*text*、*unitext*、または *image* カラムの記述を取得します。
- **rs_get_textptr** には、複写定義スコープがあります。
- 複写定義を作成すると、Replication Server は、複写定義内の複写される *text*、*unitext*、または *image* カラムごとに、*rs_sqlserver_function_class* クラスと *rs_default_function_class* クラスに対して **rs_get_textptr** ファンクション文字列を生成します。
- ユーザが作成した基本ファンクション文字列クラスを作成する場合は、複写定義に含まれる各 *text*、*unitext*、または *image* カラムに対して **rs_get_textptr** ファンクション文字列を作成してください。
- **rs_get_textptr** ファンクション文字列の作成またはカスタマイズは、複写定義を作成した Replication Server で行ってください。
- **rs_get_textptr** は、指定されたローの *text*、*unitext*、または *image* カラムについての *text* または *unitext* 型のカラム記述を返す必要があります。この *text* または *unitext* 型のカラム記述は、「I/O 記述子構造」を返す場合の Open Server の要件を満たす必要があります。この構造の詳細については、『Open Server Server-Library/C リファレンスマニュアル』を参照してください。
- ExpressConnect for Oracle および ExpressConnect for HANA データベースでは、LOB データの管理に LOB ポインタが使用されません。 その結果、LOB ポインタの管理に使用される Replication Server システムファンクションは、ExpressConnect for Oracle および ExpressConnect for HANA データベースで使用できません。 このようなファンクション (**rs_get_textptr**、**rs_textptr_init**、**rs_writetext** など) は ExpressConnect で認識できますが、ファンクションを使用しても Replication Server から無視されます。

参照：

- rs_datarow_for_writetext (541 ページ)
- rs_textptr_init (582 ページ)
- rs_writetext (592 ページ)

rs_get_thread_seq

rs_threads システムテーブル内の指定されたエントリのシーケンス番号を返します。

構文

```
rs_get_thread_seq @rs_id
```

パラメータ

- **rs_id –** *int* データ型の番号です。これはチェックするエントリの ID で、*rs_threads* システムテーブルにある *id* カラムの値と一致しています。

例

- **例 1 –** *rs_threads* テーブルに **select** 文を実行する、**rs_get_thread_seq** ファンクション文字列を作成します。

```
create function string rs_get_thread_seq
 for sqlserver_derived_class
 output language
 'select seq from rs_threads
   where id = ?rs_id!param?'
```

使用法

- Replication Server は、前のトランザクションが完了しているかどうか調べるために、**rs_get_thread_seq** を実行します。これは、1 つのコネクションに複数の DSI スレッドが定義されている場合にだけ実行されます。このファンクションは、指定された ID に対するシーケンス番号が格納された 1 つのカラム (*seq*) を含む 1 つのローを返します。
- このファンクションを起動するスレッドは、指定されたエントリを最後に変更したトランザクションが完了するまでブロックされます。
- **rs_get_thread_seq** には、ファンクション文字列クラススコープがあります。
- インストール中、Replication Server はシステム提供ファンクション文字列クラスの初期 **rs_get_thread_seq** ファンクション文字列を作成します。
- ユーザが作成した基本ファンクション文字列クラス、および並列 DSI 機能を使用する場合は、**rs_get_thread_seq** ファンクションに対してファンクション文字列を作成してください。並列 DSI を使用しない場合は、このファンクションに対してファンクション文字列を作成する必要はありません。

- **rs_get_thread_seq** ファンクション文字列の作成またはカスタマイズは、そのクラスのプライマリサイトである Replication Server で行ってください。

参照：
- configure connection (238 ページ)
- rs_initialize_threads (562 ページ)
- rs_set_isolation_level (577 ページ)
- rs_update_threads (589 ページ)

rs_get_thread_seq_noholdlock

noholdlock オプションを使用して、*rs_threads* システムテーブル内の指定されたエントリのシーケンス番号を返します。

構文

```
rs_get_thread_seq_noholdlock @rs_id
```

パラメータ

- **rs_id –** *int* データ型の番号です。これはチェックするエントリの ID で、*rs_threads* システムテーブルにある *id* カラムの値と一致しています。

例

- **例 1 –** *rs_threads* テーブルに対して **select** 文を実行する、**rs_get_thread_seq_noholdlock** ファンクション文字列を作成します。

```
create function string
   rs_get_thread_seq_noholdlock
 for sqlserver_derived_class
 output language
 'select seq from rs_threads noholdlock
   where id = ?rs_id!param?'
```

使用法

- **rs_get_thread_seq_noholdlock** は **rs_get_thread_seq** と同じですが、**dsi_isolation_level** が 3 のときに使用される点が異なります。これは、1 つのコネクションに複数の DSI スレッドが定義されている場合にだけ実行されます。ローの選択は **noholdlock** オプションを指定して行われます。このファンクションは、指定した ID の現在のシーケンス番号が格納された 1 つのカラム (*seq*) を含む 1 つのローを返します。

- **rs_get_thread_seq_noholdlock** ファンクションには、ファンクション文字列クラススコープがあります。
- インストール中、Replication Server はシステム提供ファンクション文字列クラスの初期 **rs_get_thread_seq_noholdlock** ファンクション文字列を作成します。
- ユーザが作成した基本ファンクション文字列クラスを使用し、トランザクションの独立性レベル 3 で並列 DSI 機能を使用する場合、**rs_get_thread_seq_noholdlock** ファンクションに対してファンクション文字列を作成してください。
- **rs_get_thread_seq_noholdlock** ファンクション文字列の作成またはカスタマイズは、そのクラスのプライマリサイトである Replication Server で行ってください。

参照：
- alter connection (134 ページ)
- rs_get_thread_seq (560 ページ)
- rs_initialize_threads (562 ページ)
- rs_set_isolation_level (577 ページ)
- rs_update_threads (589 ページ)

rs_initialize_threads

rs_threads システムテーブルの各エントリのシーケンスを 0 に設定します。

構文

```
rs_initialize_threads @rs_id
```

パラメータ

- **@rs_id –** 1 から *dsi_num_threads* までの番号で、Replication Server が 0 に設定するエントリの ID を指定します。

例

- **例 1 – rs_initialize_threads** という名前のストアドプロシージャを実行する **rs_initialize_threads** ファンクション文字列を作成します。ストアドプロシージャの内容を次に示します。

```
create procedure rs_initialize_threads
 @rs_id int
 as
 delete from rs_threads where id = @rs_id
```

```
 insert into rs_threads values
 (@rs_id, 0,"", "", "", "")
```

```
create function string rs_initialize_threads
 for sqlserver_derived_class
 output language
 'execute rs_initialize_threads
   @rs_id = ?rs_id!param?'
```

使用法

- コネクションが初期化されると、**rs_initialize_threads** ファンクションを実行します。これは、1 つのコネクションに複数の DSI スレッドが定義されている場合にだけ実行されます。このファンクションにより、*rs_threads* システムテーブルの各エントリのシーケンス番号が 0 に設定されます。
- **rs_initialize_threads** には、ファンクション文字列クラススコープがあります。
- インストール中、Replication Server はシステム提供ファンクション文字列クラスの初期 **rs_initialize_threads** ファンクション文字列を作成します。
- ユーザが作成した基本ファンクション文字列クラス、および並列 DSI 機能を使用する場合は、**rs_initialize_threads** に対してファンクション文字列を作成してください。
- **rs_initialize_threads** ファンクション文字列の作成またはカスタマイズは、そのクラスのプライマリサイトである Replication Server で行ってください。

参照：

- create connection (287 ページ)
- rs_get_thread_seq (560 ページ)
- rs_get_thread_seq_noholdlock (561 ページ)
- rs_set_isolation_level (577 ページ)
- rs_update_threads (589 ページ)

rs_insert

レプリケートデータベースのテーブルに、ローを 1 つ挿入します。

例

- **例 1 –** *publishers* テーブルに対する **rs_insert** ファンクション文字列を置き換えます。

```
alter function string publishers.rs_insert
 for sqlserver_derived_class
 output language
 'insert into publishers (pub_id, pub_name, city,
```

```
  state)
values (?pub_id!new?, ?pub_name!new?,
     ?city!new?, ?state!new?)'
```

使用法

- **rs_insert** には、複写定義スコープがあります。
- 複写定義を作成すると、Replication Server は、システム提供ファンクション文字列クラスに対して **rs_insert** ファンクション文字列を生成します。
- ユーザが作成した基本ファンクション文字列クラスを使用する場合、複写定義を作成するたびに **rs_insert** ファンクション文字列を作成してください。
- **rs_insert** ファンクション文字列の作成またはカスタマイズは、複写定義を作成した Replication Server で行ってください。
- 各複写定義の *rs_sqlserver_function_class* クラスと *rs_default_function_class* クラスの **rs_insert** に対してデフォルトで作成されたファンクション文字列では、Transact-SQL の **insert** コマンドの構文が使用されます。
- Replication Server は、**rs_insert** を使用して *text*、*unitext*、または *image* データをレプリケートデータベースに送信することはできませんが、*text_status* 変更子を使用して *text*、*unitext*、または *image* データのステータスを知らせることはできます。*text_status* 変更子については、「writetext」を参照してください。*text*、*unitext*、または *image* データは **rs_get_textptr**、**rs_textptr_init**、**rs_writetext** を使用してレプリケートデータベースに送信されます。

参照：

- create function string (318 ページ)
- create replication definition (346 ページ)
- rs_datarow_for_writetext (541 ページ)
- rs_delete (545 ページ)
- rs_get_textptr (558 ページ)
- rs_select (571 ページ)
- rs_select_with_lock (573 ページ)
- rs_textptr_init (582 ページ)
- rs_update (588 ページ)

rs_marker

パラメータを独立したコマンドとして Replication Server に渡します。

構文

```
rs_marker @rs_api
```

パラメータ

- **rs_api –** サブスクリプションマテリアライゼーションで使用されるデータを含む、*varchar(255)* の文字列です。

例

- **例 1 –**

```
create function string rs_marker
 for sqlserver_derived_class
 output language
 'execute rs_marker
     @rs_api = ?rs_api!param?'
```

使用法

- **rs_marker** を使用すると、Replication Server はトランザクションログにデータを挿入でき、RepAgent のスレッドでそのデータを検索できるようになります。
- **rs_marker** ファンクションには、ファンクション文字列クラススコープがあります。
- インストール中、Replication Server はシステム提供ファンクション文字列クラスの初期 **rs_marker** ファンクション文字列を作成します。
- ユーザが作成した基本ファンクション文字列クラスを使用する場合、**rs_marker** ファンクションに対してファンクション文字列を自分で作成してください。
- **rs_marker** ファンクション文字列の作成またはカスタマイズは、そのクラスのプライマリサイトである Replication Server で行ってください。
- Replication Server は、サブスクリプションマテリアライゼーション中に **rs_marker** を使用して、プライマリデータベースログを介してプライマリ Replication Server に **activate subscription** コマンドと **validate subscription** コマンドを渡します。

- プライマリデータベースの RepAgent は、**rs_marker** ファンクションの実行を認識し、*@rs_api* パラメータをコマンドとしてプライマリ Replication Server に渡す必要があります。
- Adaptive Server データベースの場合、データベースが Replication Server に対して設定されたときに、**rs_marker** という Adaptive Server 複写ストアドプロシージャが作成されます。このストアドプロシージャは、**sp_setrepproc** システムプロシージャによって、“replicated” とマーク付けされます。
- Adaptive Server の RepAgent がトランザクションログで **rs_marker** の実行を検出すると、*@rs_api* パラメータがコマンドとしてプライマリ Replication Server へ送信されます。

注意： バルクサブスクリプションを作成したとき以外は、**rs_marker** ファンクション文字列を作成したり **rs_marker** ストアドプロシージャを起動したりしないでください。詳細については、『Replication Server 管理ガイド 第 1 巻』を参照してください。

参照：

- activate subscription (48 ページ)
- create subscription (376 ページ)
- sp_setrepproc (673 ページ)
- validate subscription (525 ページ)

rs_non_blocking_commit

データサーバに対し、トランザクションがディスクに書き込まれるまで待機せず、ただちに COMMIT 文の肯定応答を送信するよう要求します。

使用法

- **rs_non_blocking_commit** には、ファンクション文字列クラススコープがあります。
- **rs_non_blocking_commit** は、**dsi_non_blocking_commit** の値が 1 ～ 60 の場合、DSI がレプリケートデータサーバに接続するたびに実行されません。**dsi_non_blocking_commit** の値が 0 の場合は、**rs_non_blocking_commit** は実行されません。
- **rs_non_blocking_commit** ファンクションは、Adaptive Server 15.0 以降では “**set delayed_commit on**” ファンクション文字列に、Oracle 10g v2 以降では対応する “**alter session set commit_write = nowait;**” ファンクション文字列にマップしま

す。その他すべての SAP 以外のデータベースでは、**rs_non_blocking_commit** は null にマップします。

- 非ブロッキングコミットを有効にした Replication Server では、Oracle 10g v2 以降への複写がサポートされます。これは、Oracle 10g v2 が、遅延コミットに似た機能をサポートしているためです。
  Replication Server 15.2 の異機種データ型サポート (HDS) スクリプトには、非ブロッキングコミット機能をサポートする新しいファンクション文字列があります。 Enterprise Connect Data Access for Oracle では、これらのファンクション文字列がサポートされます。『Replication Server Options 15.1 概要ガイド』を参照してください。

参照：

- rs_non_blocking_commit_flush (567 ページ)

rs_non_blocking_commit_flush

insert コマンド、**delete** コマンド、または **update** コマンドをデータサーバに送信し、**rs_ non_blocking_commit** で設定されたコネクション経由で送信されたトランザクションがディスクに保存されるようにします。

例

- **例 1 –** Adaptive Server 用に **rs_non_blocking_commit_flush** ファンクション文字列のインスタンスを作成します。

```
create function string rs_non_blocking_commit_flush
for sqlserver_derived_class
output language
'set delayed_commit off; begin tran; update rs_lastcommit set
origin_time = getdate() where origin = 0; commit tran;
set delayed_commit on'
```

- **例 2 –** Oracle 用に **rs_non_blocking_commit_flush** ファンクション文字列のインスタンスを作成します。

```
create function string rs_non_blocking_commit_flush
for oracle_derived_class
output language
'alter session set commit_write = immediate; begin tran;
update rs_lastcommit set origin_time = getdate() where
origin = 0; commit tran; alter session set commit_write = nowait'
```

使用法

- **rs_non_blocking_commit_flush** には、ファンクション文字列クラススコープがあります。

- **rs_non_blocking_commit_flush** は **dsi_non_blocking_commit** での間隔の指定に従って 1 ～ 60 分間隔で実行されます。**rs_non_blocking_commit_flush** は、**dsi_non_blocking_commit** がゼロの場合、実行されません。
- Adaptive Server 15.0 以降と Oracle 10g v2 以降では、**rs_non_blocking_commit_flush** は、対応するファンクション文字列にマップします。 その他すべての SAP 以外のデータベースでは、**rs_non_blocking_commit_flush** は null にマップします。
- 非ブロッキングコミットを有効にした Replication Server では、Oracle 10g v2 以降への複写がサポートされます。これは、Oracle 10g v2 が、遅延コミットに似た機能をサポートしているためです。
  Replication Server 15.2 の異機種データ型サポート (HDS) スクリプトには、非ブロッキングコミット機能をサポートする新しいファンクション文字列があります。 Enterprise Connect Data Access for Oracle では、これらのファンクション文字列がサポートされます。『Replication Server Options 15.1 概要ガイド』を参照してください。

参照：

- rs_non_blocking_commit (566 ページ)

rs_raw_object_serialization

直列化された形式の Java カラムを Replication Server で処理できるようにします。

使用法

- **rs_raw_object_serialization** を使用すると、直列化されたデータを直接レプリケートデータベースに挿入できます。
- **rs_raw_object_serialization** には、ファンクション文字列クラススコープがあります。
- インストール中、Replication Server はシステム提供ファンクション文字列クラス *rs_sqlserver_function_class* と *rs_default_function_class* の初期 **rs_raw_object_serialization** ファンクション文字列を作成します。
- Replication Server は、あるコネクションに対して最初に Java カラムがマテリアライズされるか複写されたときに、**rs_raw_object_serialization** を使用して、Adaptive Server にデフォルトのコマンドセット **rs_raw_object_serialization** を渡します。

rs_repl_on

データベースコネクションに対して Adaptive Server の複写をオンに設定します。

例

- **例 1 – rs_repl_on** ファンクション文字列のインスタンスを作成します。

```
create function string rs_repl_on
 for sqlserver_derived_class
 output language
 'set replication on'
```

使用法

- **rs_repl_on** は、データベースへの DSI コネクションに対して実行されます。
- **rs_repl_on** には、ファンクション文字列クラススコープがあります。
- インストール中、Replication Server はシステム提供ファンクション文字列クラスの初期 **rs_repl_on** ファンクション文字列を作成します。
- ユーザが作成した基本ファンクション文字列クラスを、デフォルト以外の方法で使用する場合には、**rs_repl_on** ファンクション文字列を作成してください。
- **rs_repl_on** ファンクション文字列の作成またはカスタマイズは、そのクラスのプライマリサイトである Replication Server で行ってください。

参照：

- alter connection (134 ページ)
- rs_repl_off (569 ページ)

rs_repl_off

Adaptive Server データベースでメンテナンスユーザによって実行されるトランザクションを複写するかどうかを指定します。

例

- **例 1 – rs_repl_off** ファンクション文字列のインスタンスを作成します。

```
create function string rs_repl_off
 for sqlserver_derived_class
 output language
 'set replication off'
```

使用法

- **rs_repl_off** は、スタンバイデータベースへの DSI コネクションに対して実行されます。
- **rs_repl_off** には、ファンクション文字列クラススコープがあります。
- インストール中、Replication Server はシステム提供ファンクション文字列クラスの初期 **rs_repl_off** ファンクション文字列を作成します。
- ユーザが作成した基本ファンクション文字列クラスを、デフォルト以外の方法で使用する場合には、**rs_repl_off** ファンクション文字列を作成してください。
- **rs_repl_off** ファンクション文字列の作成またはカスタマイズは、そのクラスのプライマリサイトである Replication Server で行ってください。
- スタンバイデータベースへのコネクションでは、システムが提供する *rs_default_function_class* クラスが常に使用されますが、このクラスは修正できません。したがって、ウォームスタンバイを使用していない場合は **rs_repl_off** に対してファンクション文字列を作成する必要はありません。
- **alter connection** または **configure connection** を使用して **dsi_replication** 設定パラメータを設定すれば、スタンバイデータベースへの接続時に **rs_repl_off** ファンクションを実行するかどうかを指定できます。**rs_repl_off** を実行するには **dsi_replication** を “off” に設定してください。
- ウォームスタンバイアプリケーションでは、Replication Server は **dsi_replication** を、アクティブデータベースに対して “on”、スタンバイデータベースに対して “off” に設定します。

参照：

- create connection (287 ページ)
- create function string (318 ページ)

rs_rollback

トランザクションをロールバックします。このファンクションは、今後のリリースで使用される予定です。

例

- **例 1 –** この例は、*rs_sqlserver_function_class* クラスと *rs_default_function_class* クラスに対するデフォルトの **rs_rollback** ファンクション文字列を示します。

```
create function string rs_rollback
 for sqlserver_derived_class
 output language
 'rollback transaction'
```

使用法

- プライマリデータベースのトランザクションログから検索されるロールバックトランザクションは、レプリケート Replication Server に分配されません。このため、このファンクションは絶対に実行しないでください。
- **rs_rollback** ファンクションには、ファンクション文字列クラススコープがあります。
- インストール中、Replication Server はシステム提供ファンクション文字列クラスの初期 **rs_rollback** ファンクション文字列を作成します。

参照：

- alter function string (188 ページ)
- create function string (318 ページ)
- rs_begin (538 ページ)
- rs_commit (540 ページ)

rs_select

サブスクリプションマテリアライゼーションでは複写テーブルのプライマリコピーから、サブスクリプションマテリアライゼーション解除ではテーブルのレプリケートコピーから、それぞれローを選択します。

例

- **例 1 – rs_select** ファンクション文字列のインスタンスを作成します。サブスクリプションの **where** 句が *au_lname* カラムの特定の値を指定した場合に、Replication Server はこのファンクション文字列を使用します。

```
create function string
   authors.rs_select;name_select
for flat_file_class
scan 'select * from authors
  where au_lname = ?l_name!user?'
output rpc
'execute name_sel ?l_name!user?, "authors"'
```

使用法

- **create subscription** コマンドに **without holdlock** が指定されていると、Replication Server は、**rs_select** を実行して、プライマリ Replication Server からサブスクリプションマテリアライゼーションのローを検索します。**without holdlock** は、ノンアトミックマテリアライゼーションで使用されます。このオペレーションに

使用されるファンクション文字列は、プライマリデータベースに割り当てられたクラスのものです。

- アトミックマテリアライゼーション中にデータを検索するには、レプリケートデータベースのコネクションに関連付けられたクラスではなく、プライマリデータベースのコネクションに関連付けられたファンクション文字列クラスとエラークラスを使用してください。
- **incrementally with purge** 句を使用してテーブル複写定義のサブスクリプションを削除する場合も、Replication Server は **rs_select** ファンクションを実行して、サブスクリプションマテリアライゼーション解除のローを識別します。このオペレーションに使用されるファンクション文字列は、レプリケートデータベースに割り当てられたクラスのものです。
- **create subscription** に **without holdlock** が指定されていない場合、Replication Server は **rs_select** ではなく **rs_select_with_lock** ファンクションを実行します。
- **rs_select** には、複写定義スコープがあります。
- 複写定義を作成すると、Replication Server は、システム提供ファンクション文字列クラスに対して、**rs_select** ファンクション文字列を生成します。
- ユーザが作成した基本ファンクション文字列クラスを使用する場合、複写定義を作成するたびに、各サブスクリプションの **where** 句と一致する **rs_select** ファンクション文字列を作成してください。
- **rs_select** ファンクション文字列の作成またはカスタマイズは、複写定義を作成した Replication Server で行ってください。
- 各複写定義の *rs_sqlserver_function_class* クラスと *rs_default_function_class* クラスの **rs_select** に対してデフォルトで作成されたファンクション文字列では、Transact-SQL の **select** コマンドの構文が使用されます。
- **rs_select** のファンクション文字列には、入力テンプレートと出力テンプレートがあります。入力テンプレートは、Replication Server で **create subscription** コマンドの **where** 句に一致する **where** 句を持つ SQL **select** コマンドです。
- **select** オペレーションの **where** 句とファンクション文字列の入力テンプレートが一致しない場合、Replication Server は入力テンプレートのないファンクション文字列が存在すればそれを使用します。
- 一致する入力テンプレートのあるファンクション文字列、または入力テンプレートのないファンクション文字列が見つからない場合、**rs_select** ファンクションの呼び出しは失敗します。

参照：

- alter function string (188 ページ)
- create function string (318 ページ)
- create subscription (376 ページ)
- rs_delete (545 ページ)
- rs_insert (563 ページ)

- rs_select_with_lock (573 ページ)
- rs_update (588 ページ)

rs_select_with_lock

順序の一貫性を保つために、holdlock を使って複写テーブルのプライマリコピーからサブスクリプションマテリアライゼーション用のローを選択します。

例

- **例 1 – rs_select_with_lock** ファンクション文字列のインスタンスを作成します。サブスクリプションの **where** 句に *au_lname* カラムの値が指定されている場合に、Replication Server はこのファンクション文字列を使用します。

```
create function string
   authors.rs_select_with_lock;name_select
for flat_file_class
scan 'select * from authors
  where au_lname = ?l_name!user?'
output rpc
'execute name_sel_lock ?l_name!user?, "authors"'
```

使用法

- **create subscription** に **without holdlock** 句が指定されていると、Replication Server は **rs_select_with_lock** ファンクションを実行して、プライマリ Replication Server から最初のサブスクリプションローを検索します。**without holdlock** 句は、アトミックマテリアライゼーションでは使用されません。このオペレーションに使用されるファンクション文字列は、プライマリデータベースに割り当てられたクラスのものです。
- **with purge** 句を使用してテーブル複写定義のサブスクリプションを削除する場合も、Replication Server は **rs_select_with_lock** ファンクションを実行して、サブスクリプションマテリアライゼーション解除のローを識別します。このオペレーションに使用されるファンクション文字列は、レプリケートデータベースに割り当てられたクラスのものです。
- **without holdlock** 句が **create subscription** に指定されている場合は、Replication Server は **rs_select** ファンクションを実行します (**rs_select_with_lock** ではなく)。
- **rs_select_with_lock** には、複写定義スコープがあります。
- 複写定義を作成すると、Replication Server は、システム提供ファンクション文字列クラスに対して、**rs_select_with_lock** ファンクション文字列を生成します。

- ユーザが作成した基本ファンクション文字列クラスを使用する場合、複写定義を作成するたびに、各サブスクリプションの **where** 句と一致する **rs_select_with_lock** ファンクション文字列を作成してください。
- **rs_select_with_lock** ファンクション文字列の作成またはカスタマイズは、複写定義を作成した Replication Server で行ってください。
- 各複写定義の *rs_sqlserver_function_class* クラスと *rs_default_function_class* クラスの **rs_select_with_lock** に対してデフォルトで作成されたファンクション文字列では、Transact-SQL の **select...holdlock** コマンドの構文が使用されます。
- **rs_select_with_lock** のファンクション文字列には、入力テンプレートと出力テンプレートがあります。入力テンプレートは、Replication Server で **create subscription** コマンドの **where** 句に一致する **where** 句を持つ SQL **select** コマンドです。
- **select** オペレーションの **where** 句とファンクション文字列の入力テンプレートが一致しない場合、Replication Server は入力テンプレートのないファンクション文字列が存在すればそれを使用します。
- 一致する入力テンプレートのあるファンクション文字列、または入力テンプレートのないファンクション文字列が見つからない場合、**rs_select_with_lock** ファンクションの呼び出しは失敗します。

参照：

- alter function string (188 ページ)
- create function string (318 ページ)
- create subscription (376 ページ)
- rs_delete (545 ページ)
- rs_insert (563 ページ)
- rs_select (571 ページ)
- rs_update (588 ページ)

rs_session_setting

SAP IQ レプリケートデータベースへの接続中に、SAP IQ のパラメータとデータベースオプションを設定します。

例

- **例 1 – rs_session_setting** ファンクション文字列を **my_iq_fclass** ファンクション文字列クラスに作成し、**LOAD_MEMORY_MB**、**MINIMIZE_STORAGE**、**JOIN_PREFERENCE** SAP IQ データベースオプションなど、設定する SAP IQ パラメータを含めます。

```
create function string rs_session_setting
for my_iq_fclass
output language
'set temporary option Load_Memory_MB=''200''
set temporary option Minimize_Storage=''on''
set temporary option join_preference=5'
go
```

使用法

- SAP IQ データベースオプションの値は、**TEMPORARY** キーワードで設定されるため、現在の SAP IQ コネクションにのみ適用します。 SAP IQ データベースへのコネクションを再起動すると、**TEMPORARY** キーワードなしに、デフォルト値または以前に設定した値に戻ります。 『SAP IQ リファレンス: 文とオプション』の「データベースオプション」で「データベースオプションの概要」の「オプションの設定」を参照してください。
- **rs_session_settingt** には、ファンクション文字列クラススコープがあります。
- インストール中、Replication Server は、システム提供ファンクション文字列クラスの初期 **rs_session_setting** ファンクション文字列を作成します。
- **rs_session_setting** ファンクション文字列の作成またはカスタマイズは、そのクラスのプライマリサイトである Replication Server で行ってください。
- 関数 **rs_session_setting** のファンクション文字列にデフォルト生成されるファンクション文字列：
  - **rs_sqlserver_function_class** クラスと **rs_default_function_class** クラスは空の文字列です。
  - **rs_iq_function_class** クラスは、次のとおりです。

    ```
    {set temporary option Load_Memory_MB=''200''
    set temporary option Minimize_Storage=''on''
    set temporary option join_preference=5}
    ```
- **LOAD_MEMORY_MB** データベースオプションは Sybase IQ 15.2 以降では使用されなくなりました。「Sybase IQ 15.2」の『新機能の概要』で「動作の変更点」の「データベースオプションの変更点」を参照してください。

rs_set_ciphertext

Adaptive Server テーブルへの暗号化カラムの複写を有効にします。

例

- **例 1 – "set ciphertext on"** をサポートしない Adaptive Server 以外のデータベースの場合は、**rs_set_ciphertext** を変更します。

```
alter function string rs_set_ciphertext
    for some_function_string_class
```

```
    output language
    ''
```

使用法

- **rs_set_ciphertext** は、すべてのユーザデータベースコネクションに対して **rs_usedb** の後に呼び出されます。Replication Server は Replication Server コネクションおよび RSSD コネクションに対してこのファンクション文字列を呼び出しません。
- **rs_set_ciphertext** は、**rs_default_function_class** および **rs_sqlserver_function_class** に関して "**set ciphertext on**" を発行します。その他のすべてのファンクションクラスでは、**rs_set_ciphertext** は null (空の文字列) に設定されます。
- 障害が発生した場合は、Replication Server は動作を継続して、ユーザにはレポートしません。これは、“**set ciphertext on**” をサポートしない古いバージョンの Adaptive Server との下位互換性のためです。
- 暗号化カラムは、暗号化形式の *varbinary* で Replication Server に到達します。マテリアライゼーションとマテリアライゼーション解除では、Replication Server は、データベースコネクションに対して “**set ciphertext on**” を発行するか、Adaptive Server の **ciphertext()** ファンクションを呼び出す必要があります。
- Replication Server は、複写する暗号化カラムがあるかどうか、またはターゲットデータベースが暗号化テキストプロパティを受け入れるかどうかにかかわらず、常に暗号化テキストプロパティをオンに設定します。
- 暗号化カラムをサーチャブルとして指定しないでください。Replication Server は *varbinary* カラムが暗号化テキストまたはプレーンバイナリのいずれであるのかを認識しないため、暗号化カラムがサーチャブルカラムになるのを回避することができません。
- 暗号化カラムを *varbinary* データ型以外にマップしないでください。Replication Server はカラムが暗号化されているかどうかを認識しないため、暗号化テキストが他のデータ型に変換されるのを回避することができません。
- Replication Server は、*text*、*unitext*、*image* カラムを暗号化できません。

参照：

- alter connection (134 ページ)
- alter function string (188 ページ)
- create database replication definition (300 ページ)
- create replication definition (346 ページ)

rs_set_dml_on_computed

レプリケート Adaptive Server データベースへのマテリアライズされた計算カラムの複写を通常カラムとして有効にします。

使用法

- rs_set_dml_on_computed は Adaptive Server レプリケートデータベースのコマンド **set dml_on_computed "on"** にマップします。 SAP 以外のデータベースでは、このファンクションは null にマップします。
- **rs_set_dml_on_computed** には、ファンクション文字列クラススコープがあります。
- rs_set_dml_on_computed は、コネクションが確立されるときに、常に **use database** コマンドの後の DSI で適用されます。
- **set dml_on_computed "on"** は、Adaptive Server バージョン 12.5.*x* 以前のデータベースではサポートされていません。障害が発生した場合は、Replication Server は動作を継続して、ユーザにはレポートしません。

参照：
- create replication definition (346 ページ)

rs_set_isolation_level

レプリケートデータサーバにトランザクションの独立性レベルを渡します。

例

- **例 1 – rs_set_isolation_level** ファンクション文字列のインスタンスを作成します。

```
create function string rs_set_isolation_level
for sqlserver_derived_class
output language
'set transaction isolation level?rs_isolation_level!sys_raw?'
```

使用法

- **rs_set_isolation_level** ファンクションは、レプリケートデータサーバにトランザクションの独立性レベルを渡し、値が **dsi_isolation_level** に設定されている場合には、DSI がレプリケートデータサーバに接続するたびに実行されます。

dsi_isolation_level の値が **default** であるか、設定されていない場合、Replication Server は **rs_set_isolation_level** を実行しません。

- **alter connection** または **create connection** は、**dsi_isolation_level** パラメータを変数 *rs_isolation_level* の値に設定して使用します。 サポートされる Adaptive Server の値は 0、1、2、および 3 です。 Replication Server は、ほかのデータサーバによってサポートされる、ほかのすべての独立性レベル値をサポートします。 **dsi_isolation_level** の値が **default** であるか、設定されていない場合、Replication Server はターゲットデータサーバのトランザクション独立性レベルを使用します。 独立性レベルの Replication Server によるサポートについては、『管理ガイド 第 2 巻』の「独立性レベルの選択」を参照してください。
- Replication Server は、**rs_usedb** ファンクション文字列コマンドの実行直後に **rs_set_isolation_level** を実行します。
- **rs_set_isolation_level** ファンクションには、ファンクション文字列クラススコープがあります。
- インストール中、Replication Server は Adaptive Server とデフォルトのファンクション文字列クラスの初期 **rs_set_isolation_level** ファンクション文字列を作成します。
- デフォルト以外のファンクション文字列クラスを使用して並列 DSI 機能を使用する場合、**rs_set_isolation_level** ファンクションに対してファンクション文字列を作成してください。変更したファンクション文字列には、変数 *rs_isolation_level* が含まれている必要があります。
- **rs_set_isolation_level** ファンクション文字列の作成またはカスタマイズは、そのクラスのプライマリサイトである Replication Server で行ってください。

参照：

- create connection (287 ページ)
- rs_get_thread_seq (560 ページ)
- rs_initialize_threads (562 ページ)
- rs_update_threads (589 ページ)

rs_set_quoted_identifier

引用符付き識別子を受け入れるためのデータサーバコネクションを設定します。

注意： Adaptive Server、SQL Anywhere、Microsoft SQL Server、Universal Database (UDB)、Oracle などのデータサーバでは、サポートされる長さ、特殊文字、および予約語に関して、引用符付き識別子は異なる方法で処理されます。 異機種環境では、複写されている引用符付き識別子がプライマリデータサーバとレプリケートデータサーバの両方で有効であることを確認してください。

使用法

- **rs_set_quoted_identifier** がデフォルトのファンクション文字列クラスに追加され、これにはファンクション文字列クラススコープがあります。
- **dsi_quoted_identifier** を on にすると、Replication Server は **rs_set_quoted_identifier** をレプリケートデータサーバに送信し、引用符付き識別子を予期するようにデータサーバに通知します。 レプリケートデータサーバが Adaptive Server、SQL Anywhere、または Microsoft SQL Server の場合、**rs_set_quoted_identifier** が **set quoted_identifiers on** コマンドに設定されます。 それ以外の場合は、**rs_set_quoted_identifier** が "" に設定されます。

参照：

- create connection (287 ページ)
- create replication definition (346 ページ)
- alter connection (134 ページ)
- alter replication definition (199 ページ)

rs_set_timestamp_insert

Adaptive Server テーブルへの timestamp カラムの複写を有効にします。

例

- **例 1 – set timestamp_insert on** をサポートしない Adaptive Server 以外のデータベースの場合は、**rs_set_timestamp_insert** を変更します。

```
alter function string rs_set_timestamp_insert
    for some_function_string_class
    output language
    ''
```

使用法

- **rs_set_timestamp_insert** は、すべてのユーザデータベース接続に対して **rs_usedb** の後に呼び出されます。 RSSD コネクションの場合、このファンクション文字列は呼び出されません。
- **rs_set_timestamp_insert** には、ファンクション文字列クラススコープがあります。
- **rs_set_timestamp_insert** は Adaptive Server レプリケートデータベースの **set timestamp_insert on** にマップします。Adaptive Server 以外のすべてのデータベースでは、**rs_set_timestamp_insert** は null にマップします。

- Adaptive Server 15.0.1 以前のデータベースでは、**set timestamp_insert on** はサポートされません。
- **rs_set_timestamp_insert** を実行できない場合は、Replication Server は動作を継続して、ユーザにはレポートしません。

参照：
- alter function string (188 ページ)
- create replication definition (346 ページ)

rs_setproxy

データサーバでログイン名を変更します。

使用法

- **rs_setproxy** には、ファンクション文字列クラススコープがあります。
- インストール中、Replication Server は、**rs_sqlserver_function_class** ファンクション文字列クラスに対して **rs_setproxy** ファンクション文字列を作成します。デフォルトは次のとおりです。
  set session authorization "?rs_destination_user!sys"
  生成される文字列では Adaptive Server の **set proxy** コマンドの構文が使用されます。デフォルトのファンクション文字列を置き換えるには、**alter function string** コマンドを使用してください。
- データサーバでネットワークセキュリティサービスがサポートされていない場合や、対応する **set proxy** コマンドがない場合には、**unified_login** を "not required" にするか、空の **rs_setproxy** ファンクション文字列を作成してください。
- ファンクション文字列の変数変更子 *sys* には、データサーバのログイン名が入っています。このログイン名は通常、メンテナンスユーザまたはサブスクリプションユーザのログイン名です。

参照：
- alter function string (188 ページ)
- create function string (318 ページ)

rs_sqldml

SQLDML を Replication Server に配信する複写ファンクションです。

例

- **例 1 –** SQLDML を **rs_sqldml** というストアドプロシージャとして Replication Server に送信します。

```
create proc rs_sqldml
@rs_operator char(1),
@rs_status int,
@rs_insert_column varchar(16384),
@rs_from varchar(16384),
@rs_where varchar(16384),
@rs_set varchar(16384),
@rs_select varchar(16384),
@rs_owner varchar(255),
@rs_object varchar(255),
   @rs_rowcount int
```

構文の説明は次のとおりです。

- *rs_operator* - 次のいずれか
  - U - **update**
  - D - **delete**
  - I - **insert select**
  - S - **select into**
- *rs_object* - 操作対象のテーブル名
- *rs_owner* - 操作対象のテーブルの所有者(テーブルの所有者ステータスが off の場合、所有者名は null)
- *rs_category* - 次の SQLDML カテゴリ:
  - C1 - すべての複写データベースに適用でき、同一の結果セットが生成される文
  - C2 - ウォームスタンバイデータベースまたは MSA データベースにのみ適用でき、同一の結果セットが生成される文
- *rs_status* - SQLDML ステータス
- *rs_set* - **set** 句 (**UPDATE** 文)
- *rs_where* - **where** 句
- *rs_select* - **select** 句 (**INSERT SELECT** 文または **SELECT INTO** 文)
- *rs_from* - **from** 句 (**INSERT SELECT** 文または **SELECT INTO** 文)
- *rs_insert_column* - **INSERT SELECT** 文のカラムリスト

- *rs_rowcount* - 影響を受け、**rs_sqldml** の終了時にのみ使用可能なローの数

使用法

- **rs_sqldml** は、複写ファンクションとして Replication Server に送信されます。SQLDML に **responding** 句がない場合、パラメータは null に設定されます。
- **SELECT INTO** はユーザ定義トランザクション内で実行でき、システムトランザクションとして複写されます。
- RepAgent は、**rs_sqldml** およびその影響を受けるローのログレコードを Replication Server に送信します。Replication Server は、SQLDML または影響を受けるローをターゲットに適用するかどうかを決定します。
- Adaptive Server は、SQLDML の先頭を示す **execbegin rs_sqldml**、SQLDML の最後を示す **execend rs_sqldm** を記録します。 SQLDML は、**execbegin** コマンド内にパックされます。 *@rs_rowcount* は、execend コマンド内にパックされます。
- SQLDML 複写スレッショルドローより変更の少ない SQLDML を記録しないように、Adaptive Server は **execbegin** に対して遅延ロギングを実行します。SQLDML による変更がスレッショルドローを超えるまで、execbegin は記録されません。 RepAgent は SQLDML の最初のログレコードを通知します。
- SQLDML の遅延ロギングは必須ではありません。 たとえば、Adaptive Server 以外の複写エージェントは遅延ロギングを実行しない場合があります。

rs_textptr_init

text、*unitext*、または *image* カラムにテキストポインタを割り付けます。

例

- **例 1 –** *blurbs* テーブル内の *copy* カラムに対する **rs_textptr_init** ファンクション文字列を作成します。

```
create function string blurbs_rep.rs_textptr_init;copy
for sqlserver2_function_class
output language
'update blurbs set copy = NULL
 where au_id = ?au_id!new?'
```

使用法

- Replication Server は **rs_textptr_init** ファンクションを実行します。これは、プライマリデータベースで修正が行われたことを示す、*text*、*unitext*、または *image* カラムに対するテキストポインタの割り付けを発生させたローが到着した場合です。また、Replication Server がレプリケートデータベースで **writetext** オペ

レーションを実行する必要があり、テキストポインタがまだ割り付けられていない場合にもこのファンクションが実行されます。
- **rs_textptr_init** ファンクションには、複写定義スコープがあります。
- 複写定義を作成すると、Replication Server は、複写定義内の複写される *text*、*unitext*、または *image* カラムごとに、*rs_sqlserver_function_class* クラスと *rs_default_function_class* クラスに対する **rs_textptr_init** ファンクション文字列を生成します。
- ユーザが作成した基本ファンクション文字列クラスを作成する場合は、複写定義に含まれる各 *text*、*unitext*、または *image* カラムに対して **rs_textptr_init** ファンクション文字列を作成してください。
- **rs_textptr_init** ファンクション文字列の作成またはカスタマイズは、複写定義を作成した Replication Server で行ってください。
- ExpressConnect for Oracle および ExpressConnect for HANA データベースでは、LOB データの管理に LOB ポインタが使用されません。 その結果、LOB ポインタの管理に使用される Replication Server システムファンクションは、ExpressConnect for Oracle および ExpressConnect for HANA データベースで使用できません。 このようなファンクション (**rs_get_textptr**、**rs_textptr_init**、**rs_writetext** など) は ExpressConnect で認識できますが、ファンクションを使用しても Replication Server から無視されます。

参照：
- rs_get_textptr (558 ページ)
- rs_datarow_for_writetext (541 ページ)
- rs_writetext (592 ページ)

rs_ticket_report

rs_ticket_history テーブルにチケットを挿入します。

例

- **例 1 –** カスタマイズした **rs_ticket_report** のサンプルを次に示します。

```
alter function string rs_ticket_report
for rs_sqlserver_function_class
output language
'insert rs_ticket_history(h1,h2,h3,h4,
pdb,prs,rrs,rdb,pdb_t,exec_t, dist_t,rsi_t,
dsi_t,exec_b,rsi_b,dsi_tnx,dsi_cmd,ticket)
values(?h1!param?, ?h2!param?, ?h3!param?,
?h4!param?, ?rs_origin_db!sys?, ?prs!param?,
?rrs!param?, ?rs_destination_db!sys?,
?pdb!param?, ?exec!param?, ?dist!param?,
```

```
?rsi!param?, ?dsi!param?, ?b!param?,
?rsi_b!param?, ?dsi_t!param?, ?dsi_c!param?,
?rs_ticket_param!param?)'
```

使用法

- **rs_ticket_report** には、ファンクション文字列クラススコープがあります。
- **rs_ticket_report** は、**rs_ticket** 情報を *rs_ticket_history* テーブルに書き込みます。ただし、必要に応じて、**rs_ticket** 情報を使用するように **rs_ticket_report** をカスタマイズできます。
  rs_ticket_history パラメータについては、「**rs_ticket_history**」を参照してください。
- **rs_ticket_report** を無効にするには、コネクション設定パラメータ **dsi_rs_ticket_report** を off に設定します。

参照：

- rs_ticket (737 ページ)
- rs_ticket_history (839 ページ)

rs_triggers_reset

Adaptive Server および Oracle のトリガをオフにします。

例

- **例 1 –** ユーザが作成した Adaptive Server の基本ファンクション文字列クラスに対して、**rs_triggers_reset** ファンクション文字列のインスタンスを作成します。

```
create function string rs_triggers_reset
 for sqlserver2_function_class
 output language
 'set triggers off'
```

- **例 2 –** ユーザが作成した Oracle の基本ファンクション文字列クラスに対して、**rs_triggers_reset** ファンクション文字列のインスタンスを作成します。

```
create function string rs_triggers_reset
 for oracle_function_class
 output language
 'BEGIN rs_trigger_control.enable();; END;;'
```

注意： Adaptive Server には **set triggers off** コマンドがありますが、Oracle ではセッションレベルのトリガ制御をパブリッシュしません。そのため、**create connection using profile** を使用して RS_TRIGGER_CONTROL パッケージをレプリケート Oracle データベースにインストールしてから、**rs_triggers_reset** ファ

ンクション文字列を使用できるようになります。『Replication Server 異機種間複写ガイド』の「Oracle レプリケートデータサーバの問題」を参照してください。

使用法

- デフォルトでは、**rs_triggers_reset** ファンクションはスタンバイデータベースへの DSI コネクションに対して実行され、それ以外の DSI コネクションに対しては実行されません。
- **rs_triggers_reset** には、ファンクション文字列クラススコープがあります。
- インストール中、Replication Server はシステム提供ファンクション文字列クラスの初期 **rs_triggers_reset** ファンクション文字列を作成します。
- スタンバイデータベースへのコネクションでは、システムが提供する *rs_default_function_class* クラスが常に使用されますが、このクラスは修正できません。その他のデータベースコネクションでは、次の両方の条件を満たす場合にのみ **rs_triggers_reset** ファンクションのファンクション文字列を作成してください。
  - ユーザが作成した基本ファンクション文字列クラスをデータベースコネクションが使用する。
  - コネクションに対して **dsi_keep_triggers** 設定パラメータを “off” に設定する。
- **rs_triggers_reset** ファンクション文字列の作成は、そのクラスのプライマリサイトである Replication Server で行ってください。
- データベースコネクションに対する **dsi_keep_triggers** を “off” に設定すると、コネクションの確立時に **rs_triggers_reset** が実行されます。**dsi_keep_triggers** のデフォルト値は、スタンバイデータベースでは “off”、レプリケートデータベースでは “on” です。デフォルトの設定を変更するには、**alter connection** コマンドまたは **configure connection** コマンドを使用してください。

参照：

- create connection (287 ページ)
- create function string (318 ページ)

rs_truncate

レプリケートデータベースのテーブルまたはテーブルパーティションをトランケートします。

例

- **例 1 –** 既存の **rs_truncate** ファンクション文字列 (*authors* テーブルに対する) を、Transact-SQL の **truncate table** コマンド (削除をログに記録しない) を実行するも

のから、**delete** コマンド (すべての削除をログに記録する) を実行するものへ置き換えます。

```
alter function string authors.rs_truncate
 for sqlserver_derived_class
 output language
 'delete authors'
```

次のどちらかの場合には、**rs_truncate** ファンクション文字列 (*authors* テーブルに対する) をカスタマイズする必要があります。

- レプリケートデータベースが Transact-SQL の **truncate table** コマンドをサポートしていない場合。
- レプリケートデータベースで削除をログに記録する場合。

- **例 2 –** *publisher* テーブルに対する既存の **rs_truncate** ファンクション文字列を置き換えて、**truncate table partition** を **delete** コマンドとして複写します。

```
alter function string publisher.rs_truncate
for rs_sqlserver_function_class
output language
'begin transaction
if (?1!param? = '''') /* No parameter */
delete publisher
if (?1!param? = ''A'')
delete publisher where c1 < 1000
if (?1!param? = ''B'')
delete publisher where c1 >= 1000
commit transaction'
```

- **例 3 –** レプリケートでテーブルパーティションをトランケートしないように、パラメータがあっても、ファンクション文字列が何もしないように変更します。

```
alter function string publisher.rs_truncate
for rs_sqlserver_function_class
output language
'if(?1!param? = '''') delete publisher'
```

使用法

- **rs_truncate** には、複写定義スコープがあります。Replication Server はこれを実行して、1 つのテーブルまたは 1 つ以上のテーブルパーティションをトランケートします。
- 複写定義を作成すると、Replication Server は、システム提供ファンクション文字列クラスに対して **rs_truncate** ファンクション文字列を生成します。
- ユーザが作成した基本ファンクション文字列クラスを使用する場合、複写定義を作成するたびに **rs_truncate** ファンクション文字列を作成してください。

- **rs_truncate** ファンクション文字列の作成またはカスタマイズは、複写定義を作成した Replication Server で行ってください。
- 各複写定義の *rs_sqlserver_function_class* クラスと *rs_default_function_class* クラスの **rs_truncate** に対してデフォルトで作成されたファンクション文字列では、Transact-SQL の **truncate table** コマンドの構文が使用されます。このファンクションは、個々のローの削除はログに記録せずに、テーブル内のすべてのローを削除します。
- Replication Server は、プライマリサイトで実行されたコマンドと同じコマンドを再構築します。このコマンドでは、レプリケートサイトが同じパーティション名を持っている必要があります。持っていない場合、DSI が停止します。
- パーティション名はパラメータとして **rs_truncate** ファンクションに渡されます。**rs_truncate** ファンクション文字列は位置に基づくファンクション文字列パラメータを受け入れます。位置に基づく変数を次に示します。

  ```
  ?n!param?
  ```

  ファンクション文字列変数 **?1!param?** は **rs_truncate** ファンクションの最初のパラメータに対応します。
- 位置に基づくファンクション文字列変数を含んでいる場合、ファンクション文字列には最小バージョン 1500 があります。ファンクション文字列 1500 を含んでいる場合、複写定義には少なくとも最小バージョン 1500 があります。

表 41 : ファンクション文字列変数の変更子

変更子	説明
new、*new_raw*	挿入または更新するローのカラムの新しい値への参照。
old、*old_raw*	更新または削除するローのカラムの既存値への参照。
user、*user_raw*	**rs_select** ファンクション文字列または **rs_select_with_lock** ファンクション文字列の入力テンプレートに定義されている変数への参照。
sys、*sys_raw*	システム定義変数への参照。
param、*param_raw*	ファンクションパラメータへの参照。
text_status	ファンクションパラメータへの参照またはファンクションパラメータ。パラメータがファンクション複写定義またはユーザ定義ファンクション (**create function**) によって定義されていない場合、パラメータ名の代わりに、LTL コマンドでファンクション内のパラメータの位置を示す 1 ～ 99 (先行ゼロは削除) までの番号が必要。

参照：

- alter function string (188 ページ)
- rs_datarow_for_writetext (541 ページ)

- rs_get_textptr (558 ページ)
- rs_insert (563 ページ)
- rs_delete (545 ページ)
- rs_textptr_init (582 ページ)
- rs_writetext (592 ページ)
- set (444 ページ)

rs_update

レプリケートデータベースのテーブル内のローを 1 つ更新します。

例

- **例 1 –** 既存の **rs_update** ファンクション文字列 (*authors* テーブルに対する) を、システム提供ファンクション文字列クラスに対して Replication Server が生成したデフォルトのファンクション文字列と同様のものに置き換えます。

```
alter function string authors.rs_update
 for sqlserver_derived_class
 output language
 'update authors set au_id = ?au_id!new?,
 au_lname = ?au_lname!new?,
 au_fname = ?au_fname!new?,
 phone = ?phone!new?,
 address = ?address!new?,
 city = ?city!new?,
 state = ?state!new?,
 country = ?country!new?,
 postalcode = ?postalcode!new?
 where au_id = ?au_id!old?'
```

使用法

- Replication Server は **rs_update** を実行して、テーブル内の 1 つのローを更新します。ローは、テーブルの複写定義で定義されているプライマリキーカラムによって識別されます。
- **rs_update** ファンクションには、複写定義スコープがあります。
- 複写定義を作成すると、Replication Server は、システム提供ファンクション文字列クラスに対して **rs_update** ファンクション文字列を生成します。
- ユーザが作成した基本ファンクション文字列クラスを使用する場合、複写定義を作成するたびに **rs_update** ファンクション文字列を作成してください。
- **rs_update** ファンクション文字列の作成またはカスタマイズは、複写定義を作成した Replication Server で行ってください。

- 各複写定義の *rs_sqlserver_function_class* クラスと *rs_default_function_class* クラスの **rs_update** に対してデフォルトで作成されたファンクション文字列では、Transact-SQL の **update** コマンドの構文が使用されます。このファンクションはローのすべてのカラムを置き換えます。ローは、プライマリキーカラムの更新前の値、つまり更新前のイメージを指定する **where** 句で識別されます。
- **set autocorrection** が **on** の場合、Replication Server は **rs_update** は使用しません。代わりに、**rs_delete** を呼び出して既存のローを削除し、**rs_insert** を呼び出してローを挿入します。
- Replication Server は、*text*、*unitext*、または *image* データの送信に **rs_update** を使用することはできませんが、*text*、*unitext*、または *image* データのステータスを知らせるために *text_status* 変更子を使用することはできます。*text_status* 変更子については、rs_datarow_for_writetext を参照してください。データ型が *text*、*unitext*、または *image* のデータは、**rs_get_textptr**、**rs_textptr_init**、**rs_datarow_for_writetext**、**rs_writetext** の各ファンクションでレプリケートデータベースに送信できます。

参照：

- alter function string (188 ページ)
- rs_datarow_for_writetext (541 ページ)
- rs_get_textptr (558 ページ)
- rs_insert (563 ページ)
- rs_delete (545 ページ)
- rs_textptr_init (582 ページ)
- rs_writetext (592 ページ)
- set (444 ページ)

rs_update_threads

rs_threads システムテーブル内の指定したエントリのシーケンス番号を更新する。

構文

```
rs_update_threads @rs_id, @rs_seq
```

パラメータ

- **rs_id** – *int* データ型の番号です。更新するエントリの ID を指定します。
- **rs_seq** – *int* データ型の番号です。エントリの新しいシーケンス番号を指定します。

例

- **例 1 – rs_update_threads** という名前のストアドプロシージャを実行する **rs_update_threads** ファンクション文字列を作成します。ストアドプロシージャの内容を次に示します。

```
create function string rs_update_threads
for sqlserver_derived_class
output language
'execute rs_update_threads
@rs_seq = ?rs_seq!param?,
@rs_id = ?rs_id!param?'
```

```
create procedure rs_update_threads
@rs_id int,
@rs_seq int
as
update rs_threads set seq = @rs_seq
where id = @rs_id
```

使用法

- **rs_update_threads** ファンクションは、コネクションに複数の DSI スレッドが定義されている場合に、各トランザクションの開始時に実行されます。これは、1 つのコネクションに複数の DSI スレッドが定義されている場合にだけ実行されます。
- **rs_update_threads** ファンクションには、ファンクション文字列クラススコープがあります。
- インストール中、Replication Server はシステム提供ファンクション文字列クラスの初期 **rs_update_threads** ファンクション文字列を作成します。
- ユーザが作成した基本ファンクション文字列クラスおよび並列 DSI 機能を使用する場合は、**rs_update_threads** に対してファンクション文字列を作成してください。
- **rs_update_threads** ファンクション文字列の作成またはカスタマイズは、そのクラスのプライマリサイトである Replication Server で行ってください。

参照：

- create connection (287 ページ)
- rs_get_thread_seq (560 ページ)
- rs_initialize_threads (562 ページ)
- rs_set_isolation_level (577 ページ)

rs_usedb

データサーバのデータベースコンテキストを変更します。

例

- **例 1 –** 既存の **rs_usedb** ファンクション文字列を、システム提供ファンクション文字列クラスに対して Replication Server が生成したデフォルトのファンクション文字列と同様のものに変更します。

```
alter function string rs_usedb
 for sqlserver_derived_class
 output language
 'use ?rs_destination_db!sys_raw?'
```

- **例 2 –** 複数のデータベースをサポートしていないデータサーバ用の出力テンプレートに対して、空の文字列を使って **rs_usedb** ファンクション文字列を作成します。

```
create function string rs_usedb
 for TOKYO_DS
 output language ''
```

使用法

- Replication Server の DSI は、データサーバに最初に接続したときにこのファンクションを実行します。
- **rs_usedb** には、ファンクション文字列クラススコープがあります。
- インストール中、Replication Server はシステム提供ファンクション文字列クラスの初期 **rs_usedb** ファンクション文字列を作成します。
- ユーザが作成した基本ファンクション文字列クラスを使用する場合、**rs_usedb** ファンクションに対してファンクション文字列を自分で作成してください。
- **rs_usedb** ファンクション文字列の作成またはカスタマイズは、そのクラスのプライマリサイトである Replication Server で行ってください。
- *rs_sqlserver_function_class* クラスと *rs_default_function_class* クラスに対してデフォルトで作成された **rs_usedb** ファンクションのファンクション文字列では、Transact-SQL の **use** コマンドの構文が使用されます。
- データサーバが複数のデータベースまたはデータベースコンテキストをサポートしていない場合、出力テンプレートは空の文字列 ('') となることがあります。

参照：

- alter function string (188 ページ)
- create function string (318 ページ)

rs_writetext

レプリケートデータベースの *text*、*unitext*、または *image* データを修正します。

例

- **例 1 –** RPC メソッドを使用して、*blurbs* テーブルの *copy* カラムを更新する **rs_writetext** ファンクション文字列を作成します。

```
create function string
   blurbs_rep.rs_writetext;copy
 for gw_function_class
 output rpc
 'execute update_blurbs_copy
   @copy_chunk = ?copy!new?,
   @au_id = ?au_id!new?,
   @last_chunk = ?rs_last_text_chunk!sys?,
   @writetext_log = ?rs_writetext_log!sys?'
```

- **例 2 – writetext** メソッドを使用して、*copy* カラムを更新する **rs_writetext** ファンクション文字列を作成します。Replication Server は、*copy* カラムに対して実行される **rs_get_textptr** ファンクションが返す I/O 記述子を使用して、*copy* カラムを修正します。

```
create function string
   blurbs_rep.rs_writetext;copy
 for rs_sqlserver2_function_class
 output writetext
 use primary log
```

  たとえば、**rs_get_textptr** に対するファンクション文字列がある場合、**rs_writetext** ファンクションは次のように *repcopy* カラム (*blurbs* テーブル) を修正します。

```
create function string
   blurbs_rep.rs_get_textptr;copy
 for sqlserver2_function_class
 output language
 'select repcopy from blurbs
 where au_id = ?au_id!new?'
```

- **例 3 – rs_writetext** ファンクション文字列を作成します。このファンクション文字列は、**none** メソッドを使用して *copy* カラムが更新されないよう指定します。

```
create function string
   blurbs_rep.rs_writetext;copy
 for rs_sqlserver2_function_class
 output none
```

使用法

- **rs_writetext** には、複写定義スコープがあります。
- 複写定義を作成すると、Replication Server は、複写定義内の複写される *text*、*unitext*、または *image* カラムごとに、*rs_sqlserver_function_class* クラスと *rs_default_function_class* クラスに対して **rs_writetext** ファンクション文字列を生成します。
- ユーザが作成したファンクション文字列クラスを作成する場合は、複写定義に含まれる各 *text*、*unitext*、または *image* カラムに対して **rs_writetext** ファンクション文字列を作成してください。
- **rs_writetext** ファンクション文字列の作成またはカスタマイズは、複写定義を作成した Replication Server で行ってください。
- Replication Server は、**rs_writetext** ファンクション文字列を作成するための 3 つの出力フォーマット (RPC、**writetext**、**none**) をサポートしています。
- ExpressConnect for Oracle および ExpressConnect for HANA データベースでは、LOB データの管理に LOB ポインタが使用されません。 その結果、LOB ポインタの管理に使用される Replication Server システムファンクションは、ExpressConnect for Oracle および ExpressConnect for HANA データベースで使用できません。 このようなファンクション (**rs_get_textptr**、**rs_textptr_init**、**rs_writetext** など) は ExpressConnect で認識できますが、ファンクションを使用しても Replication Server から無視されます。

RPC メソッドの使用

RPC メソッドを使用して作成した **rs_writetext** ファンクション文字列では、Replication Server はリモートプロシージャコールを繰り返し実行し、1 回の実行ごとに最高 255 バイトの *text*、*unitext*、または *image* 値を提供します。

データは、*text* または *unitext* の場合は *varchar* パラメータで、*image* の場合は *varbinary* パラメータで RPC に渡されます。*text* または *unitext* カラムに対してデータのチャンクが文字境界で分割されるようにするため、たとえば 1 バイトの文字セットが使用されている場合には、255 バイトのチャンクでデータが送信されます。

RPC が実行されるたびに、*rs_last_text_chunk* システム変数 (*int*) が、後続のデータがある場合は 0 に、現在の *text* カラムに対する最後の RPC の場合は 1 に設定されます。

- もう 1 つの *int* システム変数である *rs_writetext_log* は、プライマリデータベースで **writetext** ロギングオプションが使用された場合は 1 に、使用されなかった場合には 0 に設定されます。

- データローにある他のカラムの値は、*new* または *old* 変更子を使ってアクセスできます。プライマリデータベースで Transact-SQL の **insert** コマンドを使用した場合は、*new* 変更子を使用してください。
- *text_status* 変更子を使用すると、*text*、*unitext*、または *image* カラムのステータスを取得できます。*text_status* 変更子については、**rs_datarow_for_writetext** を参照してください。

writetext メソッドの使用

writetext メソッドを使用して **rs_writetext** ファンクション文字列を作成する場合、次の表に示すオプションを使用して、レプリケートデータベースのロギング動作を指定できます。

表 42 : writetext ロギングオプション

ロギングオプション	説明
use primary log	ロギングオプションがプライマリデータベースのトランザクションログに指定されている場合は、レプリケートデータベースのトランザクションログにデータのログを取る。プライマリデータベースのトランザクションログにロギングオプションが指定されていない場合は、ログを取らない。
with log	レプリケートデータベースのトランザクションログにデータのログを取る。
no log	レプリケートデータベースのトランザクションログにデータのログを取らない。

rs_sqlserver_function_class に対するデフォルトのファンクション文字列では、**use primary log** オプションが使用されます。

none メソッドの使用

rs_writetext ファンクション文字列の **none** 出力テンプレートオプションは、*text*、*unitext*、または *image* カラム値の更新に Client-Library 関数 **ct_send_data** を使用しないよう、Replication Server に指示します。このオプションは、異機種環境での *text*、*unitext*、または *image* カラムの使用に対応するためのものです。

詳細については、『Replication Server 管理ガイド 第 2 巻』を参照してください。

参照：

- rs_get_textptr (558 ページ)
- rs_textptr_init (582 ページ)
- rs_datarow_for_writetext (541 ページ)

SAP ASE コマンドとシステムプロシージャ

SAP ASE からの複写をサポートするために、SAP ASE コマンドとシステムプロシージャを使用して、RepAgent (Replication Agent for SAP ASE) の設定および管理を行います。

create replication filter

RepAgent が Adaptive Server データベースのテーブルに適用する条件を定義する、複写フィルタオブジェクトを作成します。

Adaptive Server RepAgent では、プライマリデータベースで複写フィルタを使用して、ログレコード内のデータがフィルタにバインドされたいずれかのパスの送信先へのデータ送信基準に一致するかを判定します。フィルタを複写パスにバインドすると、RepAgent はそのパスのフィルタ条件を満たすデータのみを複写します。

構文

```
create replication filter filter_name on table_name
as filter clause
```

パラメータ

- ***filter_name*** – フィルタオブジェクトの名前
- ***table_name*** – *filter_name* を適用するプライマリデータベースでのテーブル名
- ***filter clause*** – フィルタの検索条件 。 次を使用できます。
  - 比較演算子 - :

演算子	意味
=	等しい
>	より大きい
<	より小さい
>=	以上
<=	以下
!=	等しくない
<>	等しくない

演算子	意味
!>	より大きくない
!<	より小さくない

- 範囲 - **between** と **not between** を使用して、それぞれ指定した範囲内または範囲外の、列の値を選択します。
- リスト - **in** と **not in** を使用して、それぞれ指定した数値リスト内またはリスト外の、列の値を選択します。
- 文字の一致 - **like** および **not like** を使用して、それぞれ指定した値と一する、または一致しない、列の値を選択します。 一致または一致しない値で、ワイルドカード文字を使用できます。 『Adaptive Server Enterprise』の「リファレンスマニュアル: ビルディングブロック」の「式、識別子、およびワイルドカード文字」の「ワイルドカード文字によるパターン一致」を参照してください。
- 不定の値 - **is not null** および **is null** を使用して、それぞれデータを含むまたは含まない、列を選択します。
- 検索条件の組み合わせ - **and** および **or** を使用して、組み合わせた条件を満たす列の値を選択します。
- 組み込みの式 - `date` や `time` など、データベースで使用可能な式を使用して、列の値を選択します。 同じ入力を行う限り常に同じ値を戻す式のみ使用できます。 使用できるその他の組み込みの式は、次のとおりです。
  `cast:charindex、convert、dateadd、datediff、datename、datepart、isnull、substring、abs、acos、ascii、asin、atan、atn2、ceiling、char、cos、cot、datalength、octet_length、degrees、difference、radians、exp、floor、isnumeric、log、log10、lower、ltrim、pi、replicate、right、round、rtrim、sign、sin、space、sqrt、str、strtobin、stuff、tan、upper、power、patindex、reverse、char_length、character_length、hextoint、inttohex、daytonum、compare、day、left、len、month、str_replace、square、year、hextobigint、biginttohex、hash、hashbytes`

例

- **例 1 –** 比較演算子 - より大きい "**>**":

  `sales` テーブルに `advance_vs_revenue` という名前のフィルタを作成し、`advance` 列の値に 2 を掛けたものが `total_sales` 列の値と `price` 列の値をかけたものより大きいレコードのみを選択します。

```
create replication filter advance_vs_filter on sales
as advance * 2 > total_sales * price
```

- **例 2 –** 範囲 - **between**:

  sales テーブルに filter_in__sales という名前のフィルタを作成し、total_sales 列の値が 4095 から 12000 のレコードのみを選択します。

```
create replication filter filter_in_sales on sales
as total_sales between 4095 and 12000
```

- **例 3 –** 範囲 - **not between**:

  sales テーブルに filter_out_sales いう名前のフィルタを作成し、total_sales 列の値が 4095 から 12000 でないレコードのみを選択します。

```
create replication filter filter_out_sales on sales
as total_sales not between 4095 and 12000
```

- **例 4 –** リスト - **in**:

  sales テーブルに state_in という名前のフィルタを作成し、state 列の状態が CA、IN、または MD のいずれかであるレコードのみを選択します。

```
create replication filter state_in on sales
as state in ("CA", "IN", "MD")
```

- **例 5 –** リスト - **not in**:

  sales テーブルに state_out という名前のフィルタを作成し、state 列の状態が CA、IN、または MD のいずれかでないレコードのみを選択します。

```
create replication filter state_out on sales
as state not in ("CA", "IN", "MD")
```

- **例 6 –** 文字の一致 - **like**:

  books テーブルに author_lastname という名前のフィルタを作成し、au_lname 列の作成者の姓が "[CK]ars[eo]n" 検索条件 (姓の最初の文字が"C" または "K" で、"a"、"r"、および "s" がこの順序で続き、次の文字が "e" または "o" で、最後の文字が "n") と一致するレコードのみを選択します。

```
create replication author_lastname on books
as au_lname like "[CK]ars[eo]n"
```

  Carson と Karsen はこの条件と一致しますが、Larson と Karsin は一致しません。

- **例 7 –** 文字の一致 - **not like**:

  contacts テーブルに not_phone_num という名前のフィルタを作成し、phone 列の電話番号のうち、電話番号の冒頭に数字 "415" のないレコードのみを選択します。

```
create replication not_phone_num on contacts
as phone not like "415%"
```

- **例 8 –** 不定の値 - **null**:

  sales テーブルに advance_null という名前のフィルタを作成し、advance 列に null 値があるレコードのみを選択します。

```
create replication advance_null on sales
as advance is null
```

- **例 9 –** 既知の値 - **not null**:

  sales テーブルに advance_not_null という名前のフィルタを作成し、advance 列に null でない値があるレコードのみを選択します。

```
create replication advance_not_null on sales
as advance is not null
```

- **例 10 –** 検索条件の組み合わせ - **- <、or、between**:

  sales テーブルに advance_vs_totalsales という名前のフィルタを作成し、advance 列の値が 5000 より小さいか、total_sales 列の値が 2000 から 2500 のレコードのみを選択します。

```
create replication advance_vs_totalsales on sales
as advance < 5000 or total_sales between 2000 and 2500
```

- **例 11 –** 組み込みの式 - getdate():

  sales テーブルに older という名前のフィルタを作成し、date 列の値が現在日付より古いレコードのみを選択します。

```
create replication older on sales
as date > getdate()
```

- **例 12 –** ブール値 - **true**:

  sales テーブルに all_rows という名前のフィルタを作成し、テーブルのすべての値を選択します。

```
create replication all_rows on sales
as true
```

使用法

- 複写フィルタは、複写元のプライマリデータベース内のテーブルに対してのみ作成できます。
- フィルタは、そのテーブルを含むデータベース内のテーブルに対して作成する必要があります。

- Replication Server への接続でフィルタを適用するプライマリ Adaptive Server データベースで、**create replication filter** を実行します。
- データベースに作成する複写フィルタには、そのデータベース内でユニークな名前を指定する必要があります。
- 同じ名前の新しい複写フィルタを作成する前に、複写フィルタを削除する必要があります。
- 検索条件文字列は二重引用符 **" "** で囲む必要があります。
- 単一のバッチ内で **create replication filter** 文を他の文と結合することはできません。
- 複写フィルタにサブクエリ、**order by**、**group by** または **having** 句を含めることはできません。
- 複写フィルタに集合関数、ユーザ定義関数、システムテーブルにアクセスする関数、またはマテリアライズされていない列を含めることはできません。
- 次のデータ型を持つ列に対して複写フィルタを作成することはできません。`text`、`unitext`、`rawobject`、`image`、または `xtype` (java クラス)。
- 複数の複写パスを設定する場合は、『Replication Server 管理ガイド 第 2 巻』>「パフォーマンスチューニング」>「Multi-Path Replication」を参照してください。
- Replication Server はウォームスタンバイアプリケーションで **create replication filter** をレプリケートします。

パーミッション

create replication filter には、"create object" パーミッションが必要です。

dbcc dbrepair

オフライン複写データベースのセカンダリトランケーションポイントをクリアする Transact-SQL コマンドです。

構文

```
dbcc dbrepair(database_name, ltmignore)
```

パラメータ

- **database_name –** セカンダリトランケーションポイントをクリアするデータベースの名前です。
- **ltmignore –** 指定されたデータベースのセカンダリトランケーションポイントを非アクティブ化します。

使用法

- **dbcc dbrepair** は、オフラインデータベースのセカンダリトランケーションポイントをクリアします。オンラインデータベースのセカンダリトランケーションポイントをクリアするには、**dbcc settrunc** を **ignore** オプションとともに使用します。
- 複写データベースのトランザクションログを排出し、セカンダリトランケーションポイントをクリアしてから、アップグレードを開始することをおすすめします。 これら 2 つのタスクを実行していない場合は、アップグレード後に Adaptive Server でデータベースをオンラインにできません。
- アップグレードの前にトランザクションログの排出とセカンダリトランケーションポイントのクリアを行わなかった場合は、Adaptive Server でデータベースをオンラインにできるように、次の手順で **dbcc dbrepair** を使用します。

  dbcc dbrepair を実行する前に、次の手順に従います。

  1. オフラインデータベースで RepAgent スレッドを開始します。
  2. トランザクションログを排出します。

  トランザクションログを排出してから **dbcc dbrepair** を実行しないと、ログのすべてのトランザクションが失われます。

参照：

- dbcc settrunc (602 ページ)

dbcc gettrunc

Adaptive Server データベースに関する現在の RepAgent の情報を検索する Transact-SQL コマンドです。

構文

```
dbcc gettrunc
```

使用法

- **dbcc gettrunc** は、RepAgent が有効なデータベースに使用します。
- **dbcc gettrunc** は、次の表に示すカラムで構成されるローを 1 つ返します。

表 43 : dbcc gettrunc によって返されるカラム

カラム名 RepAgent	内容
secondary trunc page	データベースログでトランケートされていない最初のページ。
secondary trunc state	値は次のいずれかになる。 • 1 - Adaptive Server は、トランケーションページ以降のページでログをトランケートしません。 • 0 - Adaptive Server は、トランケーションページを無視する。
db rep stat	次の値で構成されるマスク。 • 0x01 - セカンダリトランケーションページは有効。 • 0x02 - データベースに明示的な複写テーブルが 1 つ以上含まれる。 • 0x04 - データベースに複写ストアドプロシージャが含まれる。 • 0x08 - スタンバイデータベースにすべてを複写する。 • 0x10 - スタンバイデータベースに L1 を複写する。 • 0x80 - 高可用性フェールオーバ後に Replication Agent が自動的に再起動する。 RepAgent のみ： • 0x20 - RepAgent は有効。 • 0x40 - RepAgent を自動起動する。
generation id	データベースの世代 ID。
database id	データベースの Adaptive Server ID 番号。
database name	データベース名。
ltl version	RepAgent：ログ転送言語 (LTL) のバージョン。

注意： 現在、Adaptive Server バージョン 12.0 以降に実装されているのはサポートレベル L1 のみであるため、スタンバイデータベースに L1 を複写しても、すべてを複写しても違いはありません。詳細については、**sp_reptostandby** を参照してください。

参照：

- admin get_generation (59 ページ)
- dbcc settrunc (602 ページ)

dbcc settrunc

Adaptive Server データベースのセカンダリトランケーションポイントの情報を修正する Transact-SQL コマンドです。

構文

```
dbcc settrunc('ltm', {'valid' | 'ignore'})
```

```
dbcc settrunc('ltm', 'gen_id', db_generation)
```

```
dbcc settrunc('ltm', {'begin' | 'end',)
```

パラメータ

- **valid –** Adaptive Server に対してセカンダリトランケーションポイントを尊重するように指示します。このオプションを指定すると、Adaptive Server は Replication Server に転送されていないトランザクションログレコードをトランケートしません。
- **ignore –** Adaptive Server に対してセカンダリトランケーションポイントを無視するように指示します。これによって、RepAgent が Replication Server にまだ転送していないログレコードを Adaptive Server がトランケートできるようになります。
- **gen_id –** Adaptive Server にログのデータベース世代番号を再設定するように指示します。
- **db_generation –** 新しいデータベース世代番号です。ダンプをリストアした後に番号を増やすことで、Replication Server が新しいトランザクションを重複したものとして拒否することを防ぎます。

  警告！ RepAgent を実行しているときには、**dbcc settrunc** は実行できません。
- **begin –** セカンダリトランケーションポイント (STP) をログの先頭に設定します。
- **end –** STP をログの終わりに設定します。

使用法

- **dbcc settrunc** は、RepAgent が有効なデータベースに使用します。
- 複写するプライマリデータを含む Adaptive Server データベースに対して、または複写ストアドプロシージャが格納されているデータベースに対して、セカンダリトランケーションポイントは **valid** でなければなりません。

- セカンダリトランケーションポイントが **valid** の場合、Adaptive Server は Replication Server が RepAgent から受信していないログレコードをトランケートしません。
- セカンダリトランケーションポイントが長期間変更されないと、ログが満杯になり、アプリケーションを継続できなくなることがあります。Replication Server および RepAgent を停止した後で、セカンダリトランケーションポイントを **ignore** に変更すれば、ログがトランケートされて、アプリケーションは作業を継続できます。この後、**rs_zeroltm** プロシージャを使用して、ロケータ値をゼロ (0) にリセットしてください。ただし、次の警告に注意してください。

  警告！ セカンダリトランケーションポイントを **ignore** に設定してログをトランケートすると、複写データに矛盾が生じます。その場合、サブスクリプションを再作成するか、**rs_subcmp** を実行してサブスクリプションを調整するか、またはデータベースとトランザクションダンプをロードして失ったトランザクションをリプレイしてください。リカバリ手順の詳細ついては、『Replication Server 管理ガイド 第 2 巻』を参照してください。コーディネートダンプをリストアした後は、データベース世代番号を増やしてください。現在の世代番号を調べるには、**admin get_generation** を使用します。

  このストアドプロシージャの実行については、**rs_zeroltm** を参照してください。
- Replication Server で新しいログレコードが拒否されないように、リストア後はデータベース世代番号を増やしてください。コーディネートダンプの再ロードについては、『Replication Server 管理ガイド 第 2 巻』を参照してください。
- プライマリ Replication Server がトランザクションを受け入れることができず、しかもプライマリデータベースのトランザクションログが満杯でトランケートが必要な場合、Adaptive Server のトランザクションを続行するために、セカンダリトランケーションポイントをオフにしてログをトランケートしなければならないことがあります。このような場合には、**dbcc settrunc('ltm', 'ignore')** を使用して、Replication Agent を停止し、データベースのセカンダリトランケーションポイントをオフに切り替えてください。

  dbcc settrunc を使用した後は、必ず **rs_zeroltm** ストアドプロシージャを使用して、データベースのロケータ値を 0 にリセットしてください。このようにしないと、*rs_locater* システムテーブルに格納されているログページが無効になることがあります。その場合、RepAgent の起動時に Adaptive Server でデータの破損が登録され、605 や 813 などのエラーが発生することがあります。
- セカンダリトランケーションポイントをオフに切り替えた後に実行されるトランザクションは、Replication Server に転送されません。したがって、プライマリデータベースとレプリケートデータベースが同期しなくなる場合があります。

  このため、ログをトランケートして Replication Server を正常に起動した後は、複写定義を変更し、サブスクリプションを削除して再作成し、レプリケート

データベース内のデータを再度マテリアライズする必要があります。データを再マテリアライズするまで、新しいカラムは null です。

Replication Server に転送されなかったトランザクションの数が比較的少ない場合には、**rs_subcmp** プログラムを使用して、プライマリデータベースとレプリケートデータベースを調整する方法を取ることもできます。

参照：

- admin get_generation (59 ページ)
- dbcc dbrepair (599 ページ)
- rs_subcmp (748 ページ)
- rs_zeroltm (739 ページ)
- sp_config_rep_agent (614 ページ)

drop replication filter

複写パスにバインドされていない複写フィルタを削除します。

構文

```
drop replication filter filter_name
```

パラメータ

- ***filter_name*** – フィルタオブジェクトの名前

例

- **例 1** – `advance_vs_revenue` という名前のフィルタを削除します。

  ```
  drop replication filter advance_vs_filter
  ```

使用法

- フィルタを削除する前に、すべての複写パスからフィルタをバインド解除する必要があります。 RepAgent では、複写パスにバインドされているフィルタは削除できません。
- フィルタに関連付けられているテーブルを削除すると、Adaptive Server でそのフィルタは削除されます。 ただし、フィルタを削除しても、関連付けられているテーブルには影響しません。
- テーブルにフィルタ付きの列がある場合、Adaptive Server の **alter table** コマンドを使用してテーブルを変更することはできません。 テーブルを変更しようと

すると、複写フィルタを削除して再作成するよう伝えるエラーメッセージが表示されます。
- フィルタを削除すると、Adaptive Server により、`sysattributes` テーブル内のそのフィルタに対応する ID を持つすべての行が削除されます。
- **alter table** を使用して関連付けられているフィルタのあるテーブルを変更しようとすると、テーブルに対する **alter table** の実行が阻止され、フィルタを削除して再作成するように伝えるエラーメッセージが表示されます。

パーミッション

drop replication filter には、"create object" パーミッションが必要です。

set replication

現在の **isql** セッションに対し、データ定義言語 (DDL) かデータ操作言語 (DML) コマンド、またはその両方のスタンバイデータベースへの複写を有効または無効にする Transact-SQL コマンドです。

構文

```
set replication [on | force_ddl | default | off]
```

パラメータ

- **on – sp_setreptable** でマーク付けされたテーブルに対して DML コマンドの複写を有効にします (**sp_reptostandby** が “none” に設定されている場合)。**sp_reptostandby** が “L1” または “all” に設定されている場合、スタンバイデータベースへの DML および DDL コマンドの複写を有効にします。デフォルトの設定です。
- **force_ddl** – 現在のセッションに対し、DDL コマンドの複写を常に有効にします。DML コマンドは、**sp_reptostandby** が “L1” または “all” に設定されている場合、すべてのユーザテーブルに対して複写されます。**sp_reptostandby** が “none” に設定されている場合、DML コマンドは **sp_setreptable** でマーク付けされたテーブルに対して複写されます。

  注意： Replication Server 12.0 以降、キーワード **force_ddl** (**set replication force_ddl** コマンドで使用) は予約語ではなくなりました。このことは、**set replication force_ddl** 自体の機能には影響しません。他のオブジェクト名で **force_ddl** を使用する場合に、二重引用符で囲む必要がなくなりました。
- **default – force_ddl** の設定を解除し、**set replication** のステータスを “on” (デフォルト) に戻します。

- **off –** 現在のセッションに対し、マーク付けされたテーブルとユーザストアドプロシージャの複写をオフに切り替えます。DML コマンドと DDL コマンドはどちらも、スタンバイデータベースまたはレプリケートデータベースにコピーされません。

使用法

- **set replication** には、Adaptive Server バージョン 11.5 以降のデータベースが必要です。

パーミッション

set replication には、“sa” または “dbo” パーミッション、および **replication_role** が必要です。

参照：

- sp_reptostandby (656 ページ)
- sp_setreptable (675 ページ)

set repmode

update、**delete**、**insert select**、または **select into** の複写を SQL 文としてセッションレベルで有効または無効にします。

構文

```
set repmode {“on” SQLDML_option | “never” | “off” | ‘threshold’,
‘value’}
```

```
SQLDML_option ::= { U | D | I | S }
```

パラメータ

- **SQLDML_option –** 次の DML オペレーションの任意の組み合わせです。
  - U - **update**
  - D - **delete**
  - I - **insert select**
  - S - **select into**

  set repmode を使用して定義した SQL 複写設定は、**sp_setrepdbmode** または **sp_setrepdefmode** を使用して定義した複写設定より優先されます。
- **on –** 指定した DML オペレーションの SQL 複写を有効にします。

- **off –** SQL 文のセッションレベルの複写設定を解除し、データベースレベルまたはテーブルレベルの設定に戻します。
- **never –** SQL 文を複写しないように指定します。

例

- **例 1 –** セッション中、**select into** と **delete** のみを SQL 文として複写するには、次のように指定します。

```
set repmode on 'DS'
```

- **例 2 –** セッション中、データベースレベルの設定やテーブルレベルの設定に関係なく SQL 文の複写を無効にするには、次のように指定します。

```
set repmode never
```

- **例 3 –** 次の例は、セッションレベルの設定がどのようにオブジェクトレベルの設定より優先されるかを示します。この例では、**update** 文のみを、SQL 文の複写を使用して複写します。

```
set repmode on 'U'
go
sp_setrepdefmode tabname, on, 'UDI'
go
```

- **例 4 –** 次の例は、セッションレベルで 1000 ローとしてスレッショルドを定義する方法を示します。

```
set repmode ‘threshold’, ‘1000’
go
```

使用法

- セッションレベルのオプションは、ログイン時に “login trigger” を使用して設定するか、バッチの先頭で設定します。セッション設定により、テーブル設定またはデータベース設定が上書きされます。
- セッションレベルの設定は、セッション中にのみ有効です。ストアドプロシージャまたはトリガ内でオプションを設定した場合、ストアドプロシージャまたはトリガの実行が終了すると、設定はテーブルレベルまたはデータベースレベルの設定に復元されます。

参照：

- sp_setrepdbmode (668 ページ)
- sp_setrepdefmode (671 ページ)

set repthreshold

SQL 文の複写がセッションでアクティブになるまでに、複写される SQL 文が影響を与える必要がある最小ロー数を指定します。

構文

```
set repthreshold value
```

パラメータ

- **value –** SQL 文の複写がセッションでアクティブになるまでに、複写される SQL 文が影響を与える必要がある最小ロー数を指定します。

例

- **例 1 –** 次の例は、データベースレベルおよびテーブルレベルのスレッショルド設定がない場合にセッションレベルでスレッショルドを 23 に定義したり、テーブルレベルおよびデータベースレベルのスレッショルド設定を上書きしたりする方法を示します。

  ```
  set repthreshold 23
  go
  ```

- **例 2 –** 次の例は、セッションレベルでデフォルトの 50 にスレッショルドをリセットする方法を示します。

  ```
  set repthreshold 0
  go
  ```

- **例 3 –** Adaptive Server ストアドプロシージャでは **set repthreshold** を呼び出すことができます。次の例は、**set_rep_threshold_23** ストアドプロシージャを作成し、**my_proc** ストアドプロシージャでそれを呼び出す方法を示します。

  1. **set_rep_threshold_23** ストアドプロシージャを作成します。

     ```
     create procedure set_rep_threshold_23
     as
     set repthreshold 23
     update my_table set my_col = 2  (statement 2)
     go
     ```

  2. **my_proc** ストアドプロシージャを作成します。

     ```
     create procedure my_proc
     as
     update my_table set my_col = 1   (statement 1)
     exec set_rep_threshold_23
     update my_table set my_col = 3   (statement 3)
     go
     ```

3. **my_proc** を実行し、**set_repthreshold_23** を呼び出します。

```
exec my_proc
go
```

my_proc ストアドプロシージャで、最初に文 1 がスレッショルド 50 で実行されます。次に文 2 がスレッショルド 23 で実行されます。次に文 3 がスレッショルド 50 で実行されます。**set repthreshold 23** コマンドは、**set_rep_threshold_23** プロシージャを実行するときにのみ有効であるためです。

- **例 4 –** 次の例は、セッションレベルのスレッショルドをエクスポート可能にする方法を示します。このため、プロシージャについて **export_options** 設定を 'on' に設定し、外部スコープのプロシージャがストアドプロシージャにより設定された SQL 文の複写スレッショルドを設定するように、SQL 文の複写スレッショルドを設定します。

1. **set_repthreshold_23** ストアドプロシージャを作成し、**export_options** を 'on' に設定します。

```
create procedure set_repthreshold_23
as
set repthreshold 23             (statement 4)
set export_options on
update my_table set my_col = 2  (statement 2)
go
```

2. **my_proc** ストアドプロシージャを作成します。

```
create procedure my_proc
as
update my_table set my_col = 1   (statement 1)
exec set_rep_threshold_23
update my_table set my_col = 3   (statement 3)
go
```

3. **my_proc** を実行し、**set_repthreshold_23** を呼び出します。

```
exec my_proc
go
```

最初に文 1 がスレッショルド 50 で実行されます。次に文 2 がスレッショルド 23 で実行されます。次に文 3 がスレッショルド 50 で実行されます。**set repthreshold 23** コマンドのスコープがセッションのスコープであるためです。

- **例 5 –** ログイントリガを作成し、特定のログイン ID の複写スレッショルドを自動的に設定します。

  - **threshold** ストアドプロシージャをスレッショルド設定 23 で作成し、エクスポートを有効にします。

```
create proc threshold
as
set repthreshold 23
```

```
set export_options on
go
```

- ユーザ “Bob” がログインしたときに **threshold** ストアドプロシージャを自動的に実行するように Adaptive Server で指定します。

```
sp_modifylogin Bob, 'login script', threshold
go
```

Bob が Adaptive Server にログインすると、セッションの SQL 文の複写スレッショルドが 23 に設定されます。

使用法

- デフォルトのスレッショルドは 50 ローです。つまり、DML 文が少なくとも 51 ローに影響を与えると、Adaptive Server は SQL 文の複写を使用します。デフォルトのスレッショルドを使用するには、**threshold** パラメータを 0 に設定します。**threshold** パラメータの範囲は 0 ～ 10,000 です。
- Adaptive Server ストアドプロシージャでは **set repthreshold** を呼び出すことができます。
- セッションレベルのスレッショルドはエクスポート可能です。このため、プロシージャについて **export_options** 設定を ‘on’ に設定し、外部スコープのプロシージャがストアドプロシージャにより設定された SQL 文の複写スレッショルドを設定するように、SQL 文の複写スレッショルドを設定します。
- セッションレベルのスレッショルドは、ログイン時に “login trigger” を使用して設定するか、バッチの先頭で設定します。セッション設定により、テーブル設定またはデータベース設定が上書きされます。
- セッションレベルのスレッショルドは、セッション中にのみ有効です。ストアドプロシージャまたはトリガ内でスレッショルドを設定した場合、ストアドプロシージャまたはトリガの実行が終了すると、設定はテーブルレベルまたはデータベースレベルの設定に復元されます。
- セッションレベルで設定したスレッショルドは、テーブルレベルとデータベースレベルのスレッショルドよりも優先されます。テーブルレベルで設定したスレッショルドは、データベースレベルで設定したスレッショルドよりも優先されます。

参照：

- sp_setrepdbmode (668 ページ)
- sp_setrepdefmode (671 ページ)
- set repmode (606 ページ)

sp_configure 'enable rep agent threads'

Adaptive Server での RepAgent スレッド統合を有効または無効にします。

構文

```
sp_configure 'enable rep agent threads'[, 1 | 0]
```

パラメータ

- **1** – データサーバの RepAgent 統合を有効にします。
- **0** – データサーバの RepAgent 統合を無効にします。

使用法

- **sp_configure 'enable rep agent threads'** は、Adaptive Server version 12.0 以降のデータベースの RepAgent を有効にするために使用します。
- **sp_configure 'enable rep agent threads'** は、現在の値、デフォルト値、最後に変更された値を表示するオプションを指定しないで使用してください。
- RepAgent は次の順序で有効にします。
  - **sp_addserver** - RepAgent の Adaptive Server を識別します。これは、一度だけ必要になります。
  - **sp_configure 'enable rep agent threads'** - RepAgent のデータサーバを有効にします。これは、一度だけ必要になります。
  - **sp_config_rep_agent** - RepAgent のデータベースを有効にします。

  sp_addserver の詳細については『Adaptive Server Enterprise リファレンスマニュアル』を参照してください。

パーミッション

sp_configure で設定パラメータを変更するには、“sa” または “sso” パーミッションが必要です。

sp_configure を実行してパラメータとその値を表示するには、パーミッションは必要ありません。

参照：

- sp_config_rep_agent (614 ページ)

sp_configure 'Rep Agent Thread administration'

現在の RepAgent スレッドプールのサイズと、他の RepAgent スレッドパラメータの設定を表示します。

構文

```
sp_configure 'Rep Agent Thread administration'
```

例

- **例 1 –** 次のように入力します。

  ```
  sp_configure 'Rep Agent Thread administration'
  ```

  次のようなメッセージが表示されます。

  ```
  Group: Rep Agent Thread Administration

  Parameter       Default  Memory  Config  Run     Unit      Type
  Name                     Used    Value   Value

  enable rep      0        0       1       1       switch    dynamic
  agent threads

  replication     4096     8194    4096    4096    memory    dynamic
  agent memory                                     pages(2k)
  size
  ```

  この例は、**enable rep agent threads** がスイッチのオン/オフを切り換える動的パラメータであることを示します。動的パラメータの変更に、RepAgent の再起動は必要ありません。

使用法

sp_configure 'replication agent memory size' を使用してマルチスレッド RepAgent のメモリを増やす前に、**sp_configure** '**Rep Agent Thread administration**' を使用して、現在、RepAgent プールに割り付けられているメモリを調べます。

sp_configure の詳細については、『Adaptive Server Enterprise リファレンスマニュアル』を参照してください。

パーミッション

sp_configure で設定パラメータを変更するには、“sa” または “sso” パーミッションが必要です。

sp_configure を実行してパラメータとその値を表示するには、パーミッションは必要ありません。

sp_configure 'replication agent memory size'

Adaptive Server がマルチスレッド RepAgent の RepAgent スレッドプールに割り付けるメモリを変更します。

構文

```
sp_configure 'replication agent memory size', repagent_mem_size
```

パラメータ

- **repagent_mem_size –** RepAgent スレッドプールに割り付けるメモリのページ数です。

例

- **例 1 –** RepAgent スレッドプールのサイズを 8194 ページに設定するには、次のように入力します。

  ```
  sp_configure 'replication agent memory size', 8194
  ```

  次のようなメッセージが表示されます。

  ```
  Group: Rep Agent Thread Administration

  Parameter       Default  Memory  Config  Run    Unit      Type
  Name                     Used    Value   Value

  replication     4096     16430   8194    8194   memory    dynamic
  agent memory                                    pages(2k)
  size
  (1 row affected)
  Configuration option changed. ASE need not be rebooted since the
  option is dynamic.
  Changing the value of 'replication agent memory size' to '8194'
  increases the amount of memory ASE uses by 8236 K.
  ```

使用法

sp_configure 'replication agent memory size' を使用してマルチスレッド RepAgent のメモリを増やす前に、**sp_configure** '**Rep Agent Thread administration**' を使用して、現在、RepAgent プールに割り付けられているメモリを調べます。

『Replication Server 管理ガイド 第 2 巻』の「パフォーマンスチューニング」の「Multi-Path Replication」で「複数のプライマリレプリケーションパス」の「マルチスレッド RepAgent および RepAgent の複数のパスを有効にする」を参照してください。

sp_configure の詳細については、『Adaptive Server Enterprise リファレンスマニュアル』を参照してください。

パーミッション

sp_configure で設定パラメータを変更するには、“sa” または “sso” パーミッションが必要です。

sp_configure を実行してパラメータとその値を表示するには、パーミッションは必要ありません。

sp_config_rep_agent

Adaptive Server データベースの RepAgent スレッドの設定パラメータを変更または表示します。

構文

```
sp_config_rep_agent [dbname[, RepAgent_parameter]
```

パラメータ

- ***dbname*** – RepAgent を設定するデータベースの名前です。
- ***RepAgent パラメータ*** – RepAgent に影響を及ぼす設定パラメータ。 RepAgent 設定パラメータ (617 ページ) を参照してください。

例

- **例 1** – `pubs2` データベースの RepAgent を有効にします。RepAgent は "repusr1" とパスワード "reppwd1" を使用して "repsvr1" に接続します。

  ```
  sp_config_rep_agent pubs2, 'enable', 'repsvr1',
    'repusr1', 'reppwd1'
  ```

- **例 2** – `pubs2` データベースの設定情報を表示します。

  ```
  sp_config_rep_agent pubs2
  ```

  ```
  Parameter NameDefaultConfig ValueRun Value
  ------------------------------------------------------------
  priority5 5 5
  trace flags0 0 0
  scan timeout 151515
  retry timeout606060
  rs usernamen/ars1_userrs1_user
  batch ltl true true true
  rs servernamen/ars1rs1
  send buffer size2k4k4k
  trace log filen/an/an/a
  ```

```
connect databasen/an/apdb1
connect dataserver n/an/apds1
scan batch size 1000 1000 1000
security mechanism n/an/an/a
msg integrityfalsefalsefalse
unified loginfalsefalsefalse
skip ltl errors falsefalsefalse
msg origin checkfalsefalsefalse
short ltl keywords falsefalsefalse
msg confidentialityfalsefalsefalse
data limits filter modestop stop stop
msg replay detectionfalsefalsefalse
mutual authentication falsefalsefalse
send structured oqids falsefalsefalse
send warm standby xactsfalsefalsefalse
msg out-of-sequence checkfalsefalsefalse
skip unsupported featuresfalsefalsefalse
send maint xacts to replicatefalsefalsefalse
net password encryptiontruetruetrue
startup delay055
cluster instance name     coordinatorcoordinator   coordinator
bind to engine-122
ltl batch size163841638416384
```

- **例 3 –** 特定のパラメータの値を表示します。

```
sp_config_rep_agent pubs2, 'scan batch size'
```

```
Parameter NameDefaultConfig ValueRun Value
-----------------------------------------------------------------
scan batch size              1000       1000             1000
```

- **例 4 – scan_timeout** (pubs2 データベース) を 60 秒に設定します。

```
sp_config_rep_agent pubs2, 'scan timeout', '60'
```

- **例 5 –** RepAgent が 50 秒待機した後に起動するように設定します。

```
sp_config_rep_agent pubs2, 'startup delay', '50'
```

- **例 6 –** ASE1 で無効になっている RepAgent を起動します。

```
1> sp_config_rep_agent pdb,
'cluster instance name','ASE1'
2> go
```

```
Parameter NameDefaultConfig ValueRun Value
-----------------------------------------------------------------
cluster instance namecoordinatorASE1ASE1
```

使用法

- **sp_config_rep_agent** は、Adaptive Server のデータベースに対する RepAgent の設定に使用します。
- RepAgent は次のように有効にします。

  - **sp_addserver** - RepAgent の Adaptive Server を識別します。これは、各画面で一度だけ必要になります。
  - **sp_configure 'enable rep agent thread'** - RepAgent のデータサーバを設定します。これは、各画面で一度だけ必要になります。
  - **sp_config_rep_agent** - RepAgent のデータベースを設定します。

  sp_addserver の詳細については『Adaptive Server Enterprise リファレンスマニュアル』を参照してください。
- **sp_config_rep_agent** でパラメータを設定した後、**sp_start_rep_agent** を使用して RepAgent を再起動して、パラメータを有効にする必要があります。
- パラメータを指定しないで **sp_config_rep_agent** を実行すると、Adaptive Server は RepAgent に対して有効なすべてのデータベースのデフォルト値、設定値、ランタイム値を表示します。
  dbname だけを指定すると、Adaptive Server は 指定したデータベースのデフォルト値、設定値、ランタイム値を表示します。
- **sp_config_rep_agent** で指定したプロパティは、データベースの *sysattributes* テーブルに格納され、*RA* の属性クラスを持ちます。
- **sp_config_rep_agent** を使用して RepAgent の設定パラメータを設定するのは、**sp_configure** を使用してデータサーバの RepAgent を有効にした後にしてください。
- repserver_user は、**connectsource** パーミッションを持っていなければなりません。

ネットワークベースセキュリティの設定

注意： RepAgent に対するネットワークベースのセキュリティは、Adaptive Server で **sp_configure** を使用すると有効になります。詳細については、『Adaptive Server Enterprise システム管理ガイド』を参照してください。

- あるセキュリティメカニズムがすべてのセキュリティプロパティをサポートするとは限りません。Replication Server で **admin security_property** を実行し、セキュリティメカニズムのプロパティを確認してください。詳細については、**admin security_property** を参照してください。
- RepAgent で有効に指定されたセキュリティメカニズムは、Replication Server で有効に指定されたものと同じでなければなりません。RepAgent と Replication Server のセキュリティ設定には互換性が必要です。

RepAgent の設定	互換性のある Replication Server の設定
true	• required • not required
false	not required

- **unified_login** が true の場合、データベースで RepAgent を有効にするときに **rs_password** パラメータを NULL に指定してください。
- セキュリティ設定を 1 つ以上設定してセキュリティメカニズムは指定しない場合には、Adaptive Server は `$SYBASE/$SYBASE_ASE/config/libtcl.cfg` の SECURITY セクションに最初に指定されているデフォルトのメカニズムを初期化します。

パーミッション

sp_configure_rep_agent には、“sa” または “dbo” パーミッション、もしくは replication_role が必要です。

参照：

- sp_configure 'enable rep agent threads' (611 ページ)
- sp_help_rep_agent (632 ページ)
- sp_start_rep_agent (678 ページ)
- sp_stop_rep_agent (681 ページ)

RepAgent 設定パラメータ

sp_config_rep_agent 設定パラメータを使用して、RepAgent の動作を設定および制御します。

RepAgent 設定パラメータ	説明
'activate monitoring', {'true' | 'false'}	タイミング情報の監視と収集を有効にします。 情報は、Adaptive Server モニタリングテーブルで収集されます。テーブルをクエリして、収集した情報を取得できます。『Adaptive Server Enterprise パフォーマンス & チューニングシリーズ: モニタリングテーブル』を参照してください。 有効な値: true または false デフォルト: false

RepAgent 設定パラメータ	説明
'auto start', {'true' | 'false'}	Adaptive Server が再起動しデータベースをリカバリしたときに RepAgent を自動的に起動するかどうかを指定します。 Adaptive Server の再起動時に RepAgent を自動的に起動するには、true に設定します。 有効な値: true または false デフォルト: false
'batch ltl'[, 'true' | 'false']	RepAgent が Replication Server に LTL コマンドをバッチで送信するか、一度に 1 つずつ送信するかを指定します。 true に設定すると、LTL コマンドを Replication Server にバッチで送信します。 それ以外の場合は、LTL コマンドを 1 回 に 1 つずつ Replication Server に送信します。 有効な値: true または false デフォルト: true
'bind to engine'[, *engine_number*]	RepAgent の実行を指定したエンジン番号に制限します。 RepAgent を専用エンジンまたは使用量の少ないエンジンで実行すると、RepAgent のパフォーマンスを向上させることができます。 **engine_number** の値の範囲は 1 ～ (**max online engines** - 1) です。 デフォルトは -1 です。これは、RepAgent をすべてのエンジンで実行できることを意味します。 **注意： bind to engine** 句によって、指定したエンジン番号での他のユーザタスクまたはシステムタスクの実行が制限されることはありません。 max online engines の説明については、『Adaptive Server Enterprise システム管理ガイド: 第 2 巻』の「**スレッドモードでのエンジンの設定**」を参照してください。
'connect database'[, '*connect_database_name*']	リカバリモードで Replication Server に接続するときに RepAgent が使用する、テンポラリデータベースの名前を指定します。 これは RepAgent が **connect source** コマンドで使用するデータベース名で、通常は、プライマリデータベースです。

RepAgent 設定パラメータ	説明
'connect dataserver'[, '*connect_dataserver_name*']	リカバリモードで Replication Server に接続するときに RepAgent が使用するデータサーバの名前を指定します。 これは RepAgent が **connect source** コマンドで使用するデータサーバ名で、通常は、プライマリデータベースのデータサーバです。
'cluster instance name'[, 'coordinator' | '*instance_name*']	RepAgent が起動されるインスタンスを制御します。 デフォルトでは、RepAgent はコーディネータの役割を持つインスタンスで起動します。 ただし、クラスタ内で宣言された任意のインスタンスで起動するように設定することもできます。 有効な値は **coordinator** または **instance name** です。 デフォルト値: **coordinator**

RepAgent 設定パラメータ	説明
'data limits filter mode'[, 'off' | 'stop' | 'skip' |'truncate']	ログレコード内に 250 個を超えるカラムや、長さ 255 バイトを超えるカラムまたはパラメータが含まれる場合に、RepAgent がそのログレコードをどのように処理してから Replication Server に送信するかを指定します。 有効な値: • **off** - RepAgent はすべてのレコードを渡します。 Replication Server 12.1 以前のバージョンでは、この設定が望ましくない結果をもたらす場合があります。 • **stop** - ログレコードに Replication Server 12.1 以前のバージョンの制限を超えるデータが含まれる場合、RepAgent は停止します。 • **skip** - RepAgent は Replication Server 12.1 以前のバージョンの制限を超えるデータを含むログレコードをスキップし、エラーログにメッセージを通知します。 • **truncate** - RepAgent は、カラムごとに 255 バイトを超えるデータ、テーブルごとに 250 カラムを超えるデータをトランケートします。 デフォルト値: • **off** - Replication Server 12.5 以降 • **stop** - Replication Server 12.1 以前 **注意：** Replication Server バージョン 12.1 以前では、**data_limits_filter_mode, off** を使用しないことをお奨めします。この設定により、RepAgent がワイドデータをスキップまたはトランケートしたり、停止したりすることがあるためです。
'ddl path for unbound objects', {'all' | 'default'}	すべてのパスまたは Multi-Path Replication 環境のデフォルトパスで、バインドされていないオブジェクトの SQL および DDL 文を送信するかどうかを指定します。 有効な値: すべてまたはデフォルト デフォルト: すべて

RepAgent 設定パラメータ	説明
'disable'[, 'preserve secondary truncpt']	データベースに対する RepAgent 使用のマーク付けを解除します。セカンダリトランケーションポイントを保持するには、**preserve secondary truncpt** を指定してください。 デフォルトではセカンダリトランケーションポイントは IGNORE に設定され、無効となります。 **disable** は、Replication Server を以前のバージョンにダウングレードしたり、プライマリデータベースを別のステータスに変更したりするときにだけ使用します。 このコマンドは `sysattributes` テーブル内のすべての RepAgent エントリをトランケートします。
'enable', '*repserver_name*', '*repserver_username*', '*repserver_password*'	データベースに RepAgent の使用を示すマークを付け、セカンダリトランケーションポイントを有効に設定します。 このパラメータは Replication Server パスワードをコード化して、Replication Server 名、Replication Server ユーザ、およびコード化したパスワードを指定されたデータベースの `sysattributes` テーブルに挿入します。 • *repserver_name* - RepAgent が接続してログトランザクションを転送する先の Replication Server の名前です • *repserver_username* - RepAgent スレッドが Replication Server に接続するときに使用するユーザ名です • *repserver_password* - RepAgent が Replication Server に接続するときに使用するパスワードです ネットワークベースのセキュリティが有効なときに **unified login** を確立する場合は、データベースで RepAgent を有効にするときに *repserver_password* に NULL を指定してください。
'ha failover'[, 'true' |'false']	SAP のフェールオーバ機能がインストールされている場合、サーバがフェールオーバした後に RepAgent を自動的に起動するかどうかを指定します。 有効な値は true または false です。 デフォルト: true

RepAgent 設定パラメータ	説明
'ltl batch size'[, *ltl_batch_size*]	RepAgent が Replication Server にバッチで送信できる LTL データの最大サイズをバイト単位で設定します。 有効な値の範囲: 16,384 ～ 2,147,483,647 バイト デフォルト値: 16,384 バイト LTL バッチのサイズを大きくすると、RepAgent のパフォーマンスを向上させることができます。 各 LTL バッチの最後に、RepAgent は前のバッチにエラーがないかチェックします。 LTL バッチのサイズを大きくすると、RepAgent が LTL エラーをチェックする回数が減ります。
'ltl metadata reduction', {'true' |'false'}	true に設定すると RepAgent テーブルのメタデータ低減が有効になります。 RepAgent のテーブルメタデータの低減を有効にした場合、Replication Server ではエグゼキュータコマンドキャッシュを使用したキャッシュが自動的に有効になります。 有効な値: true または false デフォルト: false 『Replication Server 管理ガイド 第 2 巻』の「パフォーマンスチューニング」の「チューニングパラメータの使用についての注意事項」の「エグゼキュータコマンドキャッシュ」を参照してください。

RepAgent 設定パラメータ	説明
'max number replication paths', {'*max number replication paths value*'}	RepAgent が複数の複写パスを介してプライマリデータベースのデータを複写するために使用できるパスの最大数を設定します。RepAgent は、各 RepAgent パスに対して 1 つの RepAgent 送信者スレッドを生成します。 有効な値の範囲は、 1 から MAXINT (2,147,483,647 パス) の間です。 デフォルト値: 1 **max number replication paths** がパスにバインドされる複写オブジェクトを持つパスの数より少ない場合は、エラーが報告されて終了します。 複数のプライマリ複写パスを作成するには、**multithread rep agent** RepAgent パラメータを使用し、マルチスレッド RepAgent を有効にします。 **max number replication paths** が 1 より大きい場合で **multipath distribution model** が **connection** に設定されていない場合、RepAgent はパス専用にバインドしないすべての複写オブジェクトのデフォルトのパスを引き続き使用します。 『Replication Server 管理ガイド 第 2 巻』の「パフォーマンスチューニング」の「Multi-Path Replication」で「複数のプライマリ複写パス」の「マルチスレッド RepAgent および RepAgent の複数のパスを有効にする」の「RepAgent 用の複写パスの最大数の設定」を参照してください。

RepAgent 設定パラメータ	説明
'max schema cache per scanner'[, '*max_schema_cache_per_scanner_value*']	各スキャナスレッドが複写に必要なオブジェクトスキーマの格納に使用できるメモリの最大量をバイト単位で設定します。 値の範囲: 524,288 から MAXINT (2,147,483,647 パス) の間です。 デフォルト値: 524,288 バイト Multi-Path Replication 環境向けに**複数スキャナ**で RepAgent を設定すると、RepAgent は **max schema cache per scanner** を各スキャナの最大キャッシュサイズにして、各スキャナに独自のスキーマキャッシュを指定します。 Adaptive Server は各スキャナスキーマキャッシュに、**sp_configure 'replication agent memory size'** で設定可能な既存の RepAgent グローバルメモリプールからメモリを割り当てます。 すべてのスキャナスキーマキャッシュに必要な合計メモリ量は、**max schema cache per scanner** に使用中のパス数をかけた値です。 **注意：** **max schema cache per scanner** を変更する前に、RepAgent メモリプールに十分なメモリがあることを確認してください。 **max schema cache per scanner** の変更を反映させるには、RepAgent を再起動する必要があります。
'msg confidentiality'[, 'true' | 'false']	Replication Server に送信されるメッセージをすべて暗号化するかどうかを指定します。 有効な値: true または false デフォルト: false
'msg integrity'[, 'true' | 'false']	Replication Server と交換するすべてのメッセージについて、不正変更の有無をチェックするかどうかを指定します。 有効な値: true または false デフォルト: false
'msg origin check'[, 'true' | 'false']	Replication Server から受信した各メッセージの送信元をチェックするかどうかを指定します。 有効な値: true または false デフォルト: false

RepAgent 設定パラメータ	説明
'msg out-of-sequence check'[, 'true' | 'false']	Replication Server から受信したメッセージの順番をチェックするかどうかを指定します。 有効な値: true または false デフォルト: false
'msg replay detection'[, 'true' | 'false']	Replication Server から受信したメッセージが傍受されリプレイされていないことをチェックするかどうかを指定します。 有効な値: true または false デフォルト: false
'multipath distribution model', {'object'|'connection'|'filter'}	RepAgent のマルチパス分散モデルは次のように設定します。 • **object** - オブジェクトバインド別分散。 RepAgent は、複数のオブジェクト (テーブルやストアドプロシージャなど) を特定の複写パスにバインドすることで、これらオブジェクトの並列複写を有効にします。 • **connection** - コネクション別分散。 RepAgent はユニークなシステムプロセス ID (spid) と使用可能な複写パスの数に従って複写パスを介してトランザクションを分散します。 • **filter** - フィルタ別分散。 RepAgent は 1 つのプライマリテーブルの 1 つの行の 1 つ以上の列の値に従って、複写パスを介してトランザクションを分散します。 デフォルト値: **オブジェクト** 分散モデルを変更し、新しいバインドを追加したり、RepAgent が既存のバインドを新しい分散モデルと関連付けられなくなった場合、RepAgent に警告が表示され、新しい分散モデルで一部のバインドが無視されることが示されます。 ただし、RepAgent では非アクティブなバインドも保持されます。 非アクティブなバインドのタイプと対応する分散モデルに戻すと、RepAgent は以前の非アクティブバインドの使用を再開します。 たとえば、オブジェクトバインド別分散からカラムフィルタ別分散に変更すると、RepAgent は変更前に設定していたテーブルとストアドプロシージャのバインドをすべて無視します。

RepAgent 設定パラメータ	説明
'multiple_scanners', {'true' | 'false'}	複数の RepAgent スキャナスレッドを有効または無効にします。 有効な値: true または false RepAgent で、Multi-Path Replication 環境の各パス専用のスキャナスレッドを持つ複数のスキャナスレッドを生成するには、true を設定します。 デフォルトは false で、単一のスキャナスレッドのみがすべての複写パスで共有されます。 変更内容を有効にするには、RepAgent を再起動する必要があります。
'multithread rep agent', {'true' | 'false'}	true に設定すると、RepAgent スキャナと送信者アクティビティに対して別々のスレッドを使用するマルチスレッド RepAgent が有効になります。 マルチスレッド RepAgent の有効化は、複数のプライマリ複写パスを作成するための前提条件です。 有効な値: true または false デフォルト: false 『Replication Server 管理ガイド 第 2 巻』の「パフォーマンスチューニング」の「Multi-Path Replication」で「複数のプライマリ複写パス」の「マルチスレッド RepAgent および RepAgent の複数のパスを有効にする」の「マルチスレッド RepAgent の有効化」を参照してください。 デフォルト: false
'mutual authentication'[, 'true' | 'false']	RepAgent が Replication Server に接続するときに、相互認証チェックを必要とするかどうかを指定します。 有効な値: true または false デフォルト: false
'net password encryption'[, 'true' | 'false']	RepAgent から Replication Server への接続を開始するために、クライアント側のパスワード暗号化ハンドシェイク方式を使用するか、または通常の非暗号化パスワードハンドシェイク方式を使用するかを指定します。 有効な値: true または false デフォルト: true

RepAgent 設定パラメータ	説明
'number of send buffers', {'*num_of_send_buffers*'}	マルチスレッド RepAgent のスキャナと送信者タスクが使用できる送信バッファの最大数を設定します。 **multithread rep agent** RepAgent パラメータを使用してマルチスレッド RepAgent を有効にして、複数のプライマリ複写パスを作成します。 デフォルト値: 500 バッファ 範囲: 有効な値の範囲は、 50 から MAXINT (2,147,483,647 バッファ) の間です。 変更内容を有効にするには、RepAgent を再起動する必要があります。 『Replication Server 管理ガイド 第 2 巻』の「パフォーマンスチューニング」の「Multi-Path Replication」で「複数のプライマリ複写パス」の「マルチスレッド RepAgent および RepAgent の複数のパスを有効にする」の「送信バッファの数の設定」を参照してください。
'priority'[, 'priority_value']	個々の RepAgent に相対的な優先値を設定します。 優先度の有効値の範囲は 0 から 7 です。 • 0 - 最も高い優先度 • 4 - 推奨される優先度が高い設定 • 5 - 推奨される優先度が中の設定 • 6 - 推奨される優先度が低い設定 デフォルト値: 5 **priority** の値を 0 に設定しないことをおすすめします。0 に設定すると、パフォーマンスに悪影響を与える可能性があります。
'retry timeout'[, '*retry_timeout_in_seconds*']	リトライ可能なエラーの後、または Replication Server が停止したとき、Replication Server への再接続までに RepAgent がスリープする秒数を指定します。 デフォルト値: 60 秒
'rs servername'[, '*repserver_name*']	RepAgent が接続してトランザクションログからトランザクションを転送する先の Replication Server の名前。 Replication Server の名前を変更した場合は、**rs servername** を使用します。

RepAgent 設定パラメータ	説明
'rs username'[, '*repserver_username*']	RepAgent が Replication Server へのログインに使用するユーザ名。RepAgent ユーザ名を変更する場合は、**rs username** を使用します。
'rs password'[, '*repserver_password*']	RepAgent が Replication Server へのログインに使用するパスワード。RepAgent パスワードを変更する場合は、**rs password** を使用します。
'scan batch size'[,'*no_of_qualifying_log_records*']	Replication Server に送信する各バッチのログレコードの最大数を指定します。 この件数のレコードが送信されると、RepAgent は Replication Server に新しいセカンダリトランケーションポイントを要求します。 デフォルト値: 1000 レコード **scan batch size** パラメータは、Multi-Path Replication を有効にしない場合にのみ有効です。 • Multi-Path Replication を有効にしない場合、RepAgent がセカンダリトランケーションポイントを要求する頻度は、**scan batch size** と **ltl batch size** との組み合わせに依存します。 バッチのログレコードの数が **scan batch size** の値に達すると、RepAgent は処理のためのセカンダリトランケーションポイント要求をキューに配置します。 ただし、**ltl batch size** で指定された LTL データのバイト数を RepAgent が Replication Server に送信する場合、RepAgent 送信者スレッドが処理するのは、キューに配置されたセカンダリトランケーションポイント要求のみです。 **ltl batch size** を増加すると複写パフォーマンスは向上しますが、セカンダリトランケーションポイントが高速で移動しないためにプライマリデータベースログが満杯になることを回避するために、セカンダリトランケーションポイント要求の数への影響を考慮してください。 • Multi-Path Replication を有効にする場合、**trunc point request interval** によって、RepAgent が Replication Server にセカンダリトランケーションポイントを要求する頻度が決定されます。

RepAgent 設定パラメータ	説明
'scan timeout'[, '*scan_timeout_in_seconds*']	RepAgent がトランザクションログ内のすべてのレコードのスキャンと処理を終了し、Replication Server が新しいセカンダリトランケーションポイントを送信して以前に送信されたレコードの受信確認をしていない場合に、RepAgent がスリープする秒数を指定します。**scan timeout** に指定された秒数が経過すると、Replication Server に再びセカンダリトランケーションポイントを問い合わせます。 デフォルト値: 15 秒 Replication Server が新しいセカンダリトランケーションポイントを送信するかトランザクションログを拡張して前に送信されたレコードを確認するまで、RepAgent による Replication Server への問い合わせが続けられます。 Replication Server がすべてのレコードの確認を終了し、その後新しいトランザクションレコードがログに到着しなかった場合には、RepAgent はトランザクションログが拡張されるまでスリープします。
'security mechanism'[, '*mechanism_name*']	RepAgent が Replication Server に接続するときに使用するネットワークベースメカニズムを指定します。
'send buffer size'[, '2K' | '4K' |'8K' | '16K']	RepAgent が Replication Server との通信に使用する送信バッファのサイズ (キロバイト) を制御します。 送信バッファのサイズを大きくすると RepAgent が Replication Server と通信する回数は少なくなりますが、メモリの消費量は増加します。 有効な値は次のとおりです。 2K、4K、8K、16K デフォルト値: 2K **注意：** 送信バッファのサイズは、データベースページのサイズとは関係がありません。

RepAgent 設定パラメータ	説明
'send maint xacts to replicate'[, 'true' | 'false']	サブスクリプションを作成するサイトへの分配のために、メンテナンスユーザからのレコードを RepAgent から Replication Server に送信するかどうかを指定します。 true に設定すると、RepAgent は、メンテナンスユーザによって生成されたレコードを、サブスクリプションを作成するサイトに分配するため Replication Server に送信します。 それ以外の場合、RepAgent はメンテナンスユーザからのレコードを Replication Server に送信しません。 有効な値: true または false デフォルト: false
'send structured oqids'[, 'true' | 'false']	RepAgent がオリジンキュー ID (qid) を構造化トークンとして送信するか、バイナリ文字列として送信するかを指定します。構造化トークンを指定すると、LTL で必要な領域が減少し、スループットが向上します。 有効な値: true または false デフォルト: false
'send warm standby xacts'[, 'true' | 'false']	RepAgent がメンテナンスユーザのトランザクション、スキーマ変更、システムトランザクションをウォームスタンバイデータベースに送信するかどうかを指定します。 このオプションは、ウォームスタンバイ設定で現在アクティブなデータベースの RepAgent だけに使用してください。 通常、スキーマトランザクションとシステムトランザクションは、ウォームスタンバイデータベースには送信されません。 true に設定すると、RepAgent は、スキーマトランザクション、システムトランザクション、メンテナンスユーザトランザクションを送信します。 それ以外の場合、RepAgent は、トランザクションをウォームスタンバイデータベースに送信しません。 有効な値: true または false デフォルト: false

RepAgent 設定パラメータ	説明
'short ltl keywords'[, 'true' | 'false']	RepAgent が Replication Server に送信する LTL を、省略形にするかどうかを指定します。省略形を使用すると、必要な領域と送信されるデータ量が減少します。 true に設定すると、RepAgent は LTL の省略形を使用するため、必要な領域と、Replication Server に送信されるデータ量が減少します。 有効な値: true または false デフォルト: false
'skip ltl errors'[, 'true' | 'false']	RepAgent で LTL コマンドのエラーを無視するかどうかを指定します。 このオプションは通常リカバリモードで使用します。 true に設定すると、RepAgent は **distribute** コマンドについて Replication Server から返されるエラーを記録して、そのエラーをスキップします。 false に設定した場合、これらのエラーが発生すると RepAgent は停止します。 有効な値: true または false デフォルト: false
'skip unsupported features'[, 'true' | 'false']	RepAgent に、Replication Server でサポートしていない Adaptive Server の機能のログレコードを無視させます。 このオプションは、通常、Replication Server のバージョンが Adaptive Server のバージョンより古い場合に使用します。 有効な値: true または false デフォルト: false
'startup delay'[, '*delay_value*']	Adaptive Server の起動時のどの段階で、特定の RepAgent を起動するかを制御します。 指定した時間だけ RepAgent の自動起動を遅らせて、RepAgent が Replication Server に接続しようとする前に Replication Server が稼働できるようにします。 デフォルトでは、RepAgent は自動起動時に遅延なしで起動します。 値を秒単位で設定すると、RepAgent の起動が、指定された秒数だけ遅れます。 デフォルト値: 0 秒です。

RepAgent 設定パラメータ	説明
'trunc point request interval'[, '*trunc_point_request_interval_value*']	RepAgent から Replication Server へのセカンダリトランケーションポイント要求の頻度を設定します。 値の範囲: 1 秒から MAXINT (2,147,483,647 秒) の間です。 デフォルト値: 10 秒 パラメータは動的であるため、パラメータ値に変更を加えた後に RepAgent を再起動する必要はありません。 ただし、RepAgent は前の設定の間隔が満了してから新しい値を適用します。 たとえば、現在の間隔が 60 秒で、この間隔を 100 秒に変更すると、現在の 60 秒の間隔が満了してから、新しい 100 秒の間隔が開始されます。
'unified login'[, 'true' | 'false']	ネットワークベースのセキュリティシステムが有効なときに、RepAgent から他のサーバへの接続にセキュリティクレデンシャル (true) とパスワード (false) のどちらを使用するかを指定します。 有効な値: true または false デフォルト: false

sp_help_rep_agent

RepAgent スレッドの静的および動的情報を表示します。

構文

```
sp_help_rep_agent [dbname[,
'recovery'|'process'|'config'|'scan'|'security'|'send'|'scan_verbos
e'|'all']]
```

パラメータ

- **dbname –** 情報を表示する RepAgent に対応するデータベースの名前です。 sp_help_rep_agent がデータベースに関して表示する情報は、Rep Agent に対して設定するモード (シングルスレッド、Multi-Path Replication 、または複数スキャナ) によって異なります。
- **recovery –** RepAgent に関するリカバリステータス情報を表示します (それぞれのレプリケーションパス、各パス内でスキャナによってスキャンされたログレコードの数、各スキャナのステータスなど)。
- **process –** RepAgent スレッドに関する情報を表示します。

- **config –** RepAgent に関する設定情報を表示します。
- **scan –** 複数のスキャナを有効にした場合に各スキャナに関連付けられるパスなど、RepAgent に関するログスキャン情報を表示します。
- **scan_verbose –** 複数のスキャナを有効にした場合に各スキャナに関連付けられるパスなど、RepAgent に関するログスキャン情報を表示するとともに、現在のマーカからログの最後まで処理されるログページ数を表示します。
- **セキュリティ –** ネットワークベースセキュリティメカニズムの現在の設定を表示します。
- **send –** RepAgent に割り振った送信バッファの数、送信側とスキャナの spid 番号、および複数のパスとスキャナを有効にした場合は、それぞれの送信側とスキャナに関連したパスに関する情報を表示します。
- **all –** 指定したデータベースに対して有効になっている RepAgent について、上記のすべての情報を表示します。

注意： **all** パラメータを使用する場合、**scan_verbose** 出力のセクションは表示されません。

例

- **例 1 –** 単一スレッドの RepAgent に関するリカバリ情報を表示します。

```
sp_help_rep_agent pdb, 'recovery'
```

次のような内容が表示されます。

```
Replication  Agent Recovery status

dbname connect connect status  rs servernamers username
 dataserverdatabase
------ ------------------  --------------------------------
pdb     sqlserver1 pdb   scanning  repsvr1repusr1
```

- **例 2 –** デフォルトパスと 1 つの代替レプリケーションパスが設定され、**multiple scanners** が **false** に設定されて単一のデフォルトスキャナのみが有効になっている、マルチスレッド RepAgent のリカバリとパスに関する情報を表示します。

```
sp_help_rep_agent pdb, 'recovery'
```

次のような内容が表示されます。

```
Replication  Agent Recovery status

dbname pathname   connect connect status  log records
            dataserverdatabase              scanned
------ --------   ------------------  -------------------
pdb    default    sqlserver1 pdb   sleeping  6000
pdb    path1      sqlserver1 pdb   sleeping  5999
```

- **例 3 –** 2 つの代替レプリケーションパスとデフォルトパスが設定され、**multiple scanners** が **true** に設定されて複数のスキャナが有効になっている、マルチスレッド RepAgent のリカバリとパスに関する情報を表示します。

```
sp_help_rep_agent pdb, 'recovery'
```

それぞれのスキャナがリカバリを実行している場合は、パスごとに行が表示されます。

```
Replication  Agent Recovery status

dbname pathname   connect connect status  log records
            dataserverdatabase              scanned
------ --------   ------------------  -------------------
pdb    path1      sqlserver1 pdb   sleeping  5999

pdb    path2      sqlserver1 pdb    sleeping  6000

pdb    default    sqlserver1 pdb   sleeping  6000
```

- **例 4 –** 単一スレッドの RepAgent に関するプロセス情報を表示します。

```
sp_help_rep_agent pdb2, 'process'
```

次のような内容が表示されます。

```
Replication Agent Process Status

dbnamespid start_marker  end_marker  current_marker
----- ---- ------------  ----------  --------------
pdb2 12(1240,0)  (1241,11)   (1241,11)


sleep statusstate     retry count last error
--------------------  --------------------
not sleepingsleeping  00
```

- **例 5 –** 2 つのレプリケーションパスが定義され、単一のデフォルトスキャナによってサポートされる、マルチスレッド RepAgent に関するプロセス情報を表示します。

```
sp_help_rep_agent pdb2, 'process'
```

次のような内容が表示されます。

```
Replication Agent Scanner Process Status

dbnamepathname  scanner_spid start_marker  end_marker
--------------  ------------ ------------ ----------
pdb2   n/a       12       (1240,0)      (1243,4)


current_marker  sleep_status  state
--------------  ------------  --------
(1243,4)        end of log    sleeping


Replication Agent Sender Process Status

dbnamepathname  sender_spid sleep_status  state
------ --------  ------------------------------
pdb2  default   22      empty queue   sleeping
pdb2  path1     14           empty queue   sleeping
```

```
pdb2  path2     15           empty queue   sleeping

retry_count  last_error  scanner_spid
-----------  ----------  ------------
0            0           12
0            0           12
0            0           12
```

- **例 6 –** 2 つのレプリケーションパスが設定され、複数のスキャナによってサポートされ、パスごとに 1 つのスキャナスレッドが存在する、マルチスレッド RepAgent に関するプロセス情報を表示します。

```
sp_help_rep_agent pdb2, 'process'
```

次のような内容が表示されます。

```
Replication Agent Coordinator Process Status

dbname  spid  sleep_status  state
------  ----  ------------  --------
pdb2    13    sleeping      sleeping


Replication Agent Scanner Process Status

dbnamepathname  scanner_spid start_marker  end_marker
--------------  ------------ ------------ ----------
pdb2   default   25       (2055,0)      (2060,3)
pdb2   path1     12       (1240,0)      (1243,4)
pdb2   path2     11       (1109,0)      (1131,3)


current_marker  sleep_status  state
--------------  ------------  --------
(2060,3)        end of log    sleeping
(1243,4)        end of log    sleeping
(1131,3)        end of log    sleeping


Replication Agent Sender Process Status

dbnamepathname  sender_spid sleep_status  state
------ --------  -------------------------------
pdb2  default   22      empty queue   sleeping
pdb2  path1     14      empty queue   sleeping
pdb2  path2     15      empty queue   sleeping

retry_count  last_error  scanner_spid
-----------  ----------  ------------
0            0           25
0            0           12
0            0           11
```

- **例 7 –** 複数のレプリケーションパスの有無は関係なく、単一のデフォルトスキャナに関するスキャン情報を表示します。

```
sp_help_rep_agent pdb2, 'scan'
```

次のような内容が表示されます。

```
Replication Agent Scan status

dbname  pathname  scanner_spid  start_marker  end_marker
------  --------  ------------  ------------  ----------
pdb2    n/a       33            (74281,0)    (74281,0 )

current_marker  log_recs_scanned   oldest_transaction
--------------  -----------------   -------------
( 74281,0)      8                   (0,0)
```

注意： レプリケーションパスはスキャナではなく送信側に関連付けられているので、パス名は表示されません。

- **例 8 –** 2 つのレプリケーションパスとデフォルトパス、および複数のスキャナが指定された、マルチスレッド RepAgent に関するスキャン情報を表示します。

```
sp_help_rep_agent pdb2, 'scan'
```

次のような内容が表示されます。

```
Replication Agent Scan status

dbname  pathname  scanner_spid  start_marker  end_marker
------  --------  ------------  ------------  ----------
pdb2    default   32            (21055,0)    (21060,3)
pdb2    path1     33            (74281,0)    (74281,7)
pdb2    path2     34            (74281,0)    (74281,7)

current_marker  log_recs_scanned   oldest_transaction
--------------  -----------------   -------------
(21060,3)       32                  (-1,0)
(74281,7)       32                  (0,0)
(74281,7)       32                  (0,0)
```

- **例 9 –** 2 つの複写パスとデフォルトパス、および複数のスキャナを持つマルチスレッド RepAgent についてスキャン情報を表示し、現在のマーカからログの末尾までに処理されるログページ数を表示します。`log pages left` カラムを参照してください。

```
sp_help_rep_agent pdb2, 'scan_verbose'
```

次のような内容が表示されます。

```
Replication Agent Scan status

dbname  pathname  scanner_spid  start_marker  end_marker
------  --------  ------------  ------------  ----------
pdb2    default   34            (1099,0)     (1113,1)
pdb2    path1     35            (1099,0)     (1113,1)
pdb2    path2     36            (1099,0)     (1113,1)

current_marker  log_pages_left  log_recs_scanned
--------------  ---------------  ----------------
(1113,1)        0                125
(1113,1)        0                125
```

```
(1113,1)        0               125

oldest_transaction
------------------
(0,0)
(0,0)
(0,0)
```

- **例 10 –** 単一スレッドの RepAgent に関する送信側スレッド情報を表示します。

```
sp_help_rep_agent pdb1, 'send'
```

次のような内容が表示されます。

```
Replication Agent Send Status

dbnamepathname  sender_spid total_send_buffers
send_buffers_used
----- --------  -----------   ------------------
------------------
pdb1   n/a       12       0       0


scanner_spid
------------
12
```

注意： デフォルトスキャナのみを使用する単一スレッドの RepAgent の場合、送信側とスキャナの spid は同じです。

- **例 11 –** 複写パスが定義されておらず、デフォルトの単一スキャナまたは複数スキャナを使用するように設定された、マルチスレッド RepAgent に関する送信側スレッド情報を表示します。

```
sp_help_rep_agent pdb2, 'send'
```

次のデフォルトパスのみが表示されます。

```
Replication Agent Send Status

dbnamepathname  sender_spid total_send_buffers  send_buffers_used
----- --------  -----------   ------------------
------------------
pdb2  default   14         50      0


scanner_spid
------------
12
```

- **例 12 –** 2 つの定義済み代替レプリケーションパスとデフォルトパス、および複数のスキャナが指定された、マルチスレッド RepAgent に関する送信側スレッド情報を表示します。

```
sp_help_rep_agent pdb2, 'send'
```

次のような内容が表示されます。

```
Replication Agent Send Status

dbnamepathname  sender_spid total_send_buffers  send_buffers_used
----- --------  ----------   ------------------
-----------------
pdb2  default   22           50                   0
pdb2  path1     14           50                   0
pdb2  path2     21           50                   0

scanner_spid
------------
15
12
23
```

使用法

- **sp_help_rep_agent** は、RepAgent が有効なデータベースで使用してください。
- パラメータを指定しないで **sp_help_rep_agent** を実行すると、Adaptive Server は RepAgent が有効になっているすべてのデータベースに関する情報を表示します。
- 表 44 : 'recovery' を指定した sp_help_rep_agent からの出力のカラム説明 (638 ページ)は、**recovery** パラメータを指定した **sp_help_rep_agent** の出力を示します。

表 44 : 'recovery' を指定した sp_help_rep_agent からの出力のカラム説明

カラム	説明
dbname	リカバリ中に Replication Server に転送されるデータのアーカイブログが格納されているデータベースの名前。
pathname	複数のレプリケーションパスおよびスキャナを設定した場合に、それぞれの送信側またはスキャナプロセスに関連付けられたレプリケーションパスの名前。
connect data-server	ノーマルモードでトランザクションログが Replication Server に転送されたデータベースのある、オリジナルデータサーバの名前。この情報は、Replication Server に送信される LTL **connect source** コマンドに指定されている。
connect database	ノーマルモードでトランザクションログが Replication Server に転送されたオリジナルデータベースの名前。この情報は、Replication Server に送信される LTL **connect source** コマンドに指定されている。

カラム	説明
status	RepAgent またはスキャナのアクティビティを示します。 ステータス値は次のとおりです。 • not running - RepAgent は稼働していません。 • not active - RepAgent はリカバリモードではありません。 • initial - RepAgent はリカバリモードで初期化中。 • end of log - RepAgent はリカバリモードであり、トランザクションログの終わりに到達した。 • sleeping - RepAgent はスリープモードであり、スキャンして送信する新規レコードを待機中。 • unknown - 上記以外。
log records scanned	RepAgent スキャナタスクによってスキャンされたデータベースログレコードの数。
rs servername	単一スレッドの RepAgent の場合は、RepAgent が情報を転送する先の Replication Server の名前。 `sysattributes` 設定を無効にするときに、このオプションを使用します。 複数のスキャナを設定している場合は、プライマリデータベースからの宛先 Replication Server が複数存在する可能性があるので、*rs servername* は表示されません。
rs username	単一スレッドの RepAgent の場合は、RepAgent が Replication Server にログインするときに使用するログイン名。 `sysattributes` 設定を無効にするときに、このオプションを使用します。 複数のスキャナを設定している場合は、プライマリデータベースからの宛先 Replication Server が複数存在する可能性があるので、*rs username* は表示されません。

- 表 45 : 'config' を指定した sp_help_rep_agent からの出力のカラム説明 (639 ページ)は、**config** パラメータを指定した **sp_help_rep_agent** の出力を示します。

表 45 : 'config' を指定した sp_help_rep_agent からの出力のカラム説明

カラム	説明
dbname	設定情報のクエリを行っている対象のデータベースの名前。
auto start	サーバ起動時に RepAgent が自動的に起動する場合は “true”。起動しない場合は “false”。
rs servername	RepAgent がログトランザクションを転送する Replication Server の名前。
rs username	RepAgent スレッドが Replication Server にログインするときに使用するログイン名。ログイン名は、Replication Server で **connect source** パーミッションを付与されている必要がある。

カラム	説明
scan batch size	Replication Server に送信される各バッチの最大ログレコード数。 デフォルトは 1,000。
scan timeout	RepAgent がトランザクションログ内のすべてのレコードのスキャンと処理を終了し、Replication Server がセカンダリトランケーションポイントを送信して以前に送信されたレコードの受信確認をしていない場合に、RepAgent がスリープする秒数。 デフォルトは 15 秒。
retry timeout	リトライ可能なエラーの後、または Replication Server が停止したとき、Replication Server への再接続までに RepAgent がスリープする秒数。 デフォルトは 60 秒。
skip ltl errors	RepAgent が LTL コマンドのエラーを無視する場合は “true”。エラーの発生時に RepAgent が停止する場合は “false”。リカバリモードでは **skip ltl errors** は通常 “true” に設定される。 デフォルトは “false”。
batch ltl	RepAgent が LTL コマンドをバッチにして Replication Server に送信する場合は “true”。LTL コマンドがフォーマットが済んだものから Replication Server に送信される場合は “false”。 デフォルトは “false”。
send warm standby xacts	RepAgent がスキーマ、システムトランザクション、およびメンテナンスユーザによる更新も含めたすべての更新を Replication Server に送信し、ウォームスタンバイアプリケーションのスタンバイデータベースに適用する場合は “true”。RepAgent がスタンバイデータベースに更新を送信しない場合は “false”。 デフォルトは “false”。
connect dataserver	リカバリモードで実行中に、RepAgent が Replication Server への接続に使用するデータサーバの名前。RepAgent がリカバリモードで実行中でない場合、*dbname* データベースのデータサーバの名前が入る。
connect database	リカバリモードで実行中に、RepAgent が Replication Server への接続に使用するデータベースの名前。RepAgent がリカバリモードで実行中でない場合、*dbname* データベースの名前が入る。
send maint commands to replicate	RepAgent がメンテナンスユーザからのレコードをレプリケートデータベースに送信する場合は “true”。送信しない場合は “false”。 デフォルトは “false”。

カラム	説明
ha failover	SAP のフェールオーバ機能がインストールされている場合、サーバがフェールオーバした後に RepAgent を自動的に起動するかどうかを指定する。 デフォルトは “true”。
skip unsupported features	RepAgent に、Replication Server でサポートしていない Adaptive Server の機能のログレコードを無視させる。このオプションは、通常、Replication Server のバージョンが Adaptive Server のバージョンより古い場合に使用する。 デフォルトは “false”。
short ltl keywords	RepAgent が Replication Server に送信する LTL を、省略形にするかどうかを指定する。省略形を使用すると、必要な領域と送信されるデータ量が減少する。 デフォルト値は “false”。
send buffer size	RepAgent が Replication Server との通信に使用する送信バッファのサイズを制御する。送信バッファのサイズが大きくなると、RepAgent が Replication Server と通信する回数は減少するが、使用メモリ量は増加する。 デフォルト値は “2K”。
priority	個々の RepAgent に相対的な優先値を設定する。**priority** の値の範囲は 0 ～ 7。0 は、最も高い優先度を示す。デフォルト値は 5。 **注意：** **priority** の値を 0 に設定しないことをおすすめします。
send structured oqids	RepAgent がオリジンキュー ID (OQID) を構造化トークンとバイナリ文字列のどちらとして送信するかを指定する。構造化トークンを指定した方が、LTL で必要な領域が少なくなり、スループットが向上する。 デフォルト値は “false”。

カラム	説明
data limits filter mode	新しい制限値が適用された長いカラムやパラメータ、またはより多くのカラムやパラメータを含むログレコードがある場合、Replication Server に送信する前に、それらを RepAgent で処理する方法を指定する。 • **off** - すべてのログレコードの送信を許可する。 • **stop** - ワイドデータを含むログレコードを検出すると、RepAgent は停止する。 • **skip** - ログレコードにワイドデータが含まれる場合、RepAgent はそのログレコードをスキップし、エラーログにメッセージを記録する。 **data_limits_filter_mode** のデフォルト値は Replication Server のバージョンによって異なる。Replication Server versions 12.1 以前のデフォルト値は “stop”、Replication Server versions 12.5 以降のデフォルト値は “off”。
startup delay	RepAgent の起動遅延時間 (秒単位)。デフォルトは 0。
cluster instance name	RepAgent を起動するクラスタインスタンスの名前。デフォルト値は **'coordinator'**。
bind to engine	RepAgent の実行用に指定したエンジン番号。範囲は -1 ～ (**max online engines** - 1)。ここで、**max online engines** は Adaptive Server の設定パラメータ。デフォルト値は -1 で、RepAgent をすべてのエンジンでできることを意味する。
ltl batch size	RepAgent が Replication Server に特定のバッチで送信できる LTL データの最大サイズ (バイト単位)。最小値 (デフォルト値) は 16,384 バイト。最大値は 2,147,483,647 バイト。
multithread_rep_agent	マルチスレッド RepAgent が有効であるかどうかを指定する。マルチスレッド RepAgent は、RepAgent スキャナと送信者アクティビティに対して別々のスレッドを使用する。マルチスレッド RepAgent は、プライマリレプリケーションパスを作成するための前提条件である。 デフォルト値は false。
number_of_send_buffers	マルチスレッド RepAgent のスキャナと送信者タスクが使用できる送信バッファの最大数。 有効な値の範囲は、1 ～ MAXINT (2,147,483,647 バッファ) の間。デフォルトは 50 バッファ。

カラム	説明
multipath_distribution_model	RepAgent の複写分散モデルを次のように指定する。 • **object** - モデルをオブジェクトバインド別分散に設定 (デフォルト)。 • **connection** - モデルをコネクション別分散に設定する。 • **connection** - モデルをカラムフィルタ別分散に設定する。 デフォルトは **object**。
multiple_scanners	複数の RepAgent スキャナスレッドを有効または無効にする。 RepAgent で、Multi-Path Replication 環境の各パス専用のスキャナスレッドを持つ複数のスキャナスレッドを生成するには、true を設定する。 デフォルトは false で、単一のスキャナスレッドのみがすべての複製パスで共有される。

- 表 47 : 'send' を指定した sp_help_rep_agent からの出力のカラム説明 (644 ページ)は、**process** パラメータを指定した **sp_help_rep_agent** の出力を示します。

表 46 : 'process' を指定した sp_help_rep_agent からの出力のカラム説明

カラム	説明
dbname	プロセス情報のクエリを行っている対象のデータベースの名前。
pathname	複数のレプリケーションパスおよびスキャナを設定した場合に、それぞれの送信側またはスキャナプロセスに関連付けられたレプリケーションパスの名前。
spid	データサーバ内のプロセスのシステムプロセス ID。 次のようになる。 • 単一スレッドの RepAgent - spid は、送信側とスキャナの両方のタスクを実行する RepAgent プロセスを示します。 • 複数スレッドの RepAgent - 複数のスキャナを有効にした場合、spid はコーディネータタスクを示します。
scanner_spid	データサーバ内の各スキャナプロセスのシステムプロセス ID。
sender_spid	データサーバ内の各送信側プロセスのシステムプロセス ID。
start marker	現在のバッチで最初にスキャンされたレコードを識別します。
end marker	現在のバッチで最後にスキャンされるレコードを識別します。
current marker	現在スキャン中のレコードを識別します。

カラム	説明
sleep status	スリープステータス値は次のとおりです。 • waiting for rewrite - RepAgent は 2 フェーズコミットトランザクションがコミットされるのを待っています。 • end of log - RepAgent はログの終わりに到達し、ログが拡張されるのを待っています。 • connect retry - RepAgent は、Replication Server への接続を試みる前の待機状態です。 • sleeping - RepAgent タスクはサスペンドされており、アクティビティの待機状態です。 • empty queue - RepAgent 送信側タスクが処理するトランザクションがキューに存在せず、アクティビティの待機状態です。 • not sleeping - 上記以外。 RepAgent はアクティブです。
state	コーディネータ、スキャナ、または送信側タスクの状態値。 • "Sleeping" - RepAgent タスクはサスペンドされており、アクティビティの待機状態です。 • "Awake" - RepAgent タスクはアクティブです。
retry count	最後に成功した接続以降に、RepAgent が Replication Server への接続に失敗した回数。
last error	最後に発生した Replication Server エラーまたはコネクションエラーのエラー番号。

- 表 47 : 'send' を指定した sp_help_rep_agent からの出力のカラム説明 (644 ページ)は、**send** パラメータを指定した **sp_help_rep_agent** の出力を示します。

表 47 : 'send' を指定した sp_help_rep_agent からの出力のカラム説明

カラム	説明
dbname	送信側スレッド情報のクエリを行っている対象のデータベースの名前。
pathname	複数のレプリケーションパスおよびスキャナを設定した場合に、それぞれの送信側またはスキャナプロセスに関連付けられたレプリケーションパスの名前。
sender_spid	データサーバ内の各送信側プロセスのシステムプロセス ID。
scanner_spid	データサーバ内の各スキャナプロセスのシステムプロセス ID。
total_send_buffers	各送信者タスクに割り当てられた送信バッファの数。
send_buffers_used	各送信者タスクにより使用されている送信バッファの数。

- 表 48 : 'scan' と 'scan_verbose' を指定した sp_help_rep_agent からの出力のカラム説明 (645 ページ)は、**scan** パラメータと **scan_verbose** パラメータを指定した **sp_help_rep_agent** の出力を示します。

表 48 : 'scan' と 'scan_verbose' を指定した sp_help_rep_agent からの出力のカラム説明

カラム	説明
dbname	スキャナスレッド情報のクエリを行っている対象のデータベースの名前。
pathname	複数のレプリケーションパスおよびスキャナを設定した場合に、それぞれの送信側またはスキャナプロセスに関連付けられたレプリケーションパスの名前。
scanner_spid	データサーバ内の各スキャナプロセスのシステムプロセス ID。
start marker	現在のバッチで最初にスキャンされたレコードを識別する。
end marker	現在のバッチで最後にスキャンされるレコードを識別する。
current marker	現在スキャン中のレコードを識別する。
log pages left	現在のマーカからログの最後までの処理されるログページの数。 **注意： scan_verbose** を使用する場合のみ表示されます。
log recs scanned	現在のバッチでスキャンされたログレコードの数。
oldest transaction	現在スキャンしているバッチ内の最も古いトランザクションを識別する。

- 表 49 : 'security' を指定した sp_help_rep_agent からの出力のカラム説明 (645 ページ)は、**security** パラメータを指定した **sp_help_rep_agent** の出力を示します。

表 49 : 'security' を指定した sp_help_rep_agent からの出力のカラム説明

カラム	説明
dbname	セキュリティ情報のクエリを行っている対象のデータベースの名前。
security mechanism	有効になっているセキュリティメカニズムの名前。
unified login	RepAgent が Replication Server への接続にクレデンシャル (“true”) とパスワード (“false”) のどちらを使用するかを指定する。デフォルトは “false”。

カラム	説明
mutual authentication	RepAgent が Replication Server に接続するときに、相互認証チェックを使用するかどうかを指定する。デフォルトは “false”。
msg confidentiality	RepAgent から Replication Server へ送信されるすべてのデータに対し、メッセージを暗号化するかどうか。デフォルトは “false”。
msg integrity	RepAgent が Replication Server と交換するすべてのデータに対し、メッセージの整合性チェックを行うどうかを指定する。デフォルトは “false”。
msg replay detection	第三者によるデータの傍受とリプレイを RepAgent でチェックして検出するかどうかを指定する。デフォルトは “false”。
msg origin check	RepAgent が Replication Server から送信されたデータの送信元を確認するかどうかを指定する。デフォルトは “false”。
msg out-of-sequence	RepAgent が Replication Server からのメッセージを送信順に受信したことを確認するかどうかを指定する。デフォルトは “false”。
net password encryption	Replication Server との接続をクライアント側パスワード暗号化ハンドシェイクによって開始するかどうかを示す。デフォルトは “true”。

パーミッション

sp_help_rep_agent には、“sa” または “dbo” パーミッション、もしくは **replication_role** が必要です。

参照：

- sp_config_rep_agent (614 ページ)
- sp_start_rep_agent (678 ページ)
- sp_stop_rep_agent (681 ページ)

sp_replication_path

プライマリデータベースから Replication Server への代替レプリケーションのパスを作成して管理します。

構文

```
sp_replication_path 'dbname', {
'add' 'physical_path', 'repserver_name', 'rs_username',
'rs_password' |
'add', 'logical', 'logical_path', 'physical_path' |
'drop', 'physical_path' |
'drop', 'logical', 'logical_path', [,'physical_path'] |
```

```
'bind', '{table | sproc | filter}', '[table_owner].object_name',
'path_name' |
'unbind', '{table | sproc | filter | path}', 'object_name',
{'path_name' | all} |
'config', 'path_name', 'config_parameter', 'config_value' |
'list'[, 'all | table | sproc | filter' [, 'object_name']]
```

パラメータ

- **dbname –** RepAgent を設定するデータベースの名前です。
- **add –** *dbname* から Replication Server への代替物理 RepAgent パスを追加します。
  - *physical_path* - 代替 RepAgent パスの名前です。
  - *repserver* - *dbname* から接続する複写の名前です。
  - *rs_username* - *repserver* に接続する適切な権限のあるユーザ名です。通常、これはメンテナンスユーザです。
  - *rs_password* - *rs_username* のパスワードです。
- **add, logical –** 複数の Replication Server の物理パスに現在バインドされているデータやオブジェクトバウンドの分散に使用できる論理 RepAgent パスを追加します。
  - *logical_path* - 論理パスの名前です。
- **drop –** 送信先としての Replication Server を、プライマリ複写のデフォルトパスでない物理複写のパスから削除します。
- **drop, logical –** 物理パスなど、論理複写のパスから要素を削除します。
- **bind –** オブジェクトを物理または論理プライマリ複写のパスに関連付けます。バインドされたオブジェクトは複写中、常に同じパスに従います。
  - **table** | **sproc** | **filter** - オブジェクトのタイプを指定します。テーブル、ストアドプロシージャ (**sproc**)、またはフィルタを指定できます。
  - *[table_owner].object_name* - テーブル、ストアドプロシージャ、またはフィルタ名を指定し、オプションでテーブル所有者名を指定します。

    注意： オブジェクトがテーブルの場合にテーブル所有者を指定しなければ、dbo、すなわちデータベース所有者が所有しているテーブルにのみバインドが適用されます。
  - *path_name* - 物理または論理複写パス名です。
- **unbind –** バインドされたオブジェクトと物理または論理複写のパスとの関連付けを削除します。
  - **table** | **sproc** | **filter** | **path** - オブジェクトのタイプを指定します。テーブル、ストアドプロシージャ (**sproc**)、フィルタ、またはパスを指定できます。
  - *[ttable_owner.]objject_name* - テーブル、ストアドプロシージャ、またはフィルタ名を指定し、オプションでテーブル所有者名を指定します。

注意： オブジェクトがテーブルの場合にテーブル所有者を指定しなければ、dbo が所有しているテーブルにのみバインド解除が適用されます。

  - *path_name* | **all** - 物理パス名か論理パス名、またはすべてのパスを指定します。 **path** を *object_type* と指定し、*object_name* でパス名を指定し、**all** オプションを指定した場合、指定したパス名からすべてのオブジェクトがバインド解除されます。
- **config –** 代替複写のパスのパラメータ値を設定します。
  - *config_parameter* - **rs username** または **rs password** です。
  - *config_value* - **rs username** には *rs_username*、**rs password** には *rs_password*。
- **list –** アクティブバインド状態または非アクティブバインド状態の複写オブジェクトについての情報を表示します。 アクティブバインドとは、RepAgent が現在の分散モデルのもとでデータの複写に使用するバインドです。
  - no option - アクティブバインドのみを表示する場合は、オプションを指定しないでください
  - **all** | **table** | **sproc** | **filter** | **path** - **all** を使用して複写パスへのすべてのアクティブバインドおよび非アクティブバインドをリストするか、オブジェクトのタイプを指定します。テーブル、ストアドプロシージャ (**sproc**)、フィルタ、またはパスを指定できます。
  - *object_name* - 特定のオブジェクトのバインド関係を表示します。 オブジェクトの名前を指定する場合は、テーブル、ストアドプロシージャ、フィルタ、パスなどのオブジェクトタイプを指定する必要があります。

例

- **例 1 –** 代替物理レプリケーションパスの作成
  - RS2 ユーザ ID と RS2_password を使用して、PDS データサーバ内の pdb データベースと RS2 Replication Server の間に pdb_1 代替物理複写パスを作成します。 PDS で次のように入力します。

    ```
    sp_replication_path 'pdb', 'add', 'pdb_1', 'RS2', 'RS2_user',
    'RS2_password'
    ```
  - RS1 ユーザ ID と RS1_password を使用して、PDS データサーバ内の pdb データベースと RS1 Replication Server の間に pdb_2 代替物理複写パスを作成します。 PDS で次のように入力します。

    ```
    sp_replication_path 'pdb', 'add', 'pdb_2', 'RS1', 'RS1_user',
    'RS1_password'
    ```

  これで、pdb からの物理複写パスが 3 つ作成されました。pdb_1、pdb_2、既存のデフォルトパスの複写パスで、代替物理複写パスを作成する前に、RS1 または RS2 に作成する必要があります。

- **例 2 –** pdb_1 物理パスによってサポートされている logical_1 論理パスを作成します。PDS で次のように入力します。

  ```
  sp_replication_path 'pdb', 'add', 'logical', 'logical_1', 'pdb_1'
  ```

- **例 3 –** 既存の logical_1 論理パスをサポートする pdb_2 物理パスを追加します。

  ```
  sp_replication_path 'pdb, 'add', 'logical', 'logical_1', 'pdb_2'
  ```

- **例 4 –** 物理パスの送信先として RS1 Replication Server を削除します。

  ```
  sp_replication_path 'pdb', 'drop', 'RS1'
  ```

- **例 5 –** 物理パスを論理パスから削除します。

  - logical_1 から pdb_1 を削除します。

    ```
    sp_replication_path 'pdb', 'drop', 'logical', 'logical_1',
    'pdb_1'
    ```

  - logical_1 から pdb_2 を削除します。

    ```
    sp_replication_path 'pdb', 'drop', 'logical', 'logical_1',
    'pdb_2'
    ```

- **例 6 –** logical_1 論理パスを削除します。

  ```
  sp_replication_path 'pdb', 'drop', 'logical', 'logical_1'
  ```

- **例 7 –** オブジェクトを物理パスまたは論理レプリケーションパスにバインドします。

  バインドするもの

  - `t1` テーブルを pdb_2 複写パスにバインドします。

    ```
    sp_replication_path 'pdb', 'bind', 'table', 't1', 'pdb_2'
    ```

  - `owner1` が所有する `t2` テーブルを pdb_2 複写パスに:

    ```
    sp_replication_path 'pdb', 'bind', 'table', 'owner1.t2',
    'pdb_2'
    ```

  - **sproc1** ストアドプロシージャを pdb_2 複写パスに:

    ```
    sp_replication_path 'pdb', 'bind', 'sproc', 'sproc1', 'pdb_2'
    ```

  - `dt1` 次元テーブルオブジェクトを論理パスのすべての場所に:

    ```
    sp_replication_path 'pdb', 'bind', 'table', 'dt1', 'everywhere'
    ```

  - `F1` フィルタを pdb_1 複写パスに:

    ```
    sp_replication_path 'pdb', 'bind', 'filter', 'F1', 'pdb_1'
    ```

    フィルタをパスにバインドする前に、**create replication filter** コマンドで F1 フィルタを作成する必要があります。

  必要に応じて、*object_name* でワイルドカード文字のアスタリスク "*"、パーセント "%"、または両方の組み合わせを使用して、パスにバインドする名前の範囲または一致する文字を指定します。 たとえば、さまざまなワイルドカード

文字の組み合わせと一致する名前のテーブルを pdb_2 複写パスにバインドするには、以下のコマンドを実行します。

- `sp_replication_path 'pdb', 'bind', 'table', 'a*', 'pdb_2'`
- `sp_replication_path 'pdb', 'bind', 'table', 'au%rs', 'pdb_2'`
- `sp_replication_path 'pdb', 'bind', 'table', 'a*th%s', 'pdb_2'`
- `sp_replication_path 'pdb', 'bind', 'table', 'authors%', 'pdb_2'`

- **例 8 –** オブジェクトを複写パスからバインド解除します。

  削除するもの

  - `t1` テーブルから pdb_2 複写パスへのバインド:

    ```
    sp_replication_path 'pdb', 'unbind', 'table', 't1', 'pdb_2'
    ```
  - `t1` テーブルのすべてのバインド:

    ```
    sp_replication_path 'pdb', 'unbind', 'table', 't1', 'all'
    ```
  - pdb_2 複写パスへのすべてのオブジェクトのバインド:

    ```
    sp_replication_path 'pdb', 'unbind', 'path', 'pdb_2', 'all'
    ```

- **例 9 –** 代替レプリケーションパスのパスワードとユーザ ID を変更します。

  変更するもの

  - pdb_1 代替複写パスが RS1 に接続するために使用するユーザ名を 'RS1_user' に:

    ```
    sp_replication_path 'pdb', 'config', 'pdb_1', 'rs_username',
    'RS1_user'
    ```
  - pdb_1 が RS1 への接続に使用するパスワードを 'january' に:

    ```
    sp_replication_path 'pdb', 'config', 'pdb_1', 'rs password',
    'january'
    ```

- **例 10 –** 分散モデルがオブジェクトバインドの場合に、アクティブバインド状態のすべてのオブジェクトのパスの関係を表示します。

  ```
  sp_replication_path 'pdb','list'
  go
  ```

  オブジェクトタイプ (テーブルの場合は T、ストアドプロシージャの場合は P) とオブジェクトがバインドされている物理パスまたは論理パスを示す出力が表示されます。

  ```
  Binding              Type     Path
  -------------------- -------- --------------------
  dbo.dt1              T        everywhere
  dbo.sproc1           P        pdb_1
  dbo.sproc1           P        pdb_2
  dbo.t1               T        pdb_2
  dbo.t2               T        pdb_1

  (5 rows affected)
  ```

```
Logical Path                  Physical Path
----------------------------  --------------------
everywhere                    pdb_1
everywhere                    pdb_2

(2 rows affected)
Physical Path                  Destination
----------------------------   --------------------
pdb_1                          RS2
pdb_2                          RS1

(2 rows affected)
(return status = 0)
```

- **例 11 –** 分散モデルがオブジェクトフィルタの場合に、アクティブバインド状態のすべてのオブジェクトのパスの関係を表示します。

```
sp_replication_path 'pdb','list'
go
```

オブジェクトタイプ (複写フィルタの場合は RF) とフィルタがバインドされているパスを示す出力が表示されます。

```
Binding              Type     Path
-------------------- -------- --------------------
dbo.F1               RF       pdb_1
dbo.F2               RF       pdb_2

(2 rows affected)


Physical Path                  Destination
------------------------------ --------------------
pdb_1                          RS2
pdb_2                          RS1
default                        RS_default
(3 rows affected)
(return status = 0)
```

- **例 12 –** 分散モデルがオブジェクトフィルタの場合に、アクティブバインド状態および非アクティブバインド状態のすべてのオブジェクトを含むパスの関係を表示します。

```
sp_replication_path 'pdb','list', 'all'
go
```

オブジェクトタイプ (フィルタ (RF)、テーブル (T)、ストアドプロシージャ (P)) とオブジェクトがバインドされているパス、およびオブジェクトがアクティブバインド状態か非アクティブバインド状態を示す出力が表示されます。

```
Binding              Type     Path
-------------------- -------- --------------------
dbo.F1               RF       pdb_1
dbo.F2               RF       pdb_2
(2 rows affected)
```

```
Inactive Binding     Type     Path
-------------------- -------- --------------------
dbo.dt1              T        everywhere
dbo.sproc1           P        pdb_1
dbo.sproc1           P        pdb_2
dbo.t1               T        pdb_2
dbo.t2               T        pdb_1

(5 rows affected)


Physical Path                 Destination
----------------------------- --------------------
pdb_3                         RS2
pdb_4                         RS1

(2 rows affected)
(return status = 0)
```

- **例 13 –** 分散モデルがオブジェクトバインドの場合に、バインドされたすべてのテーブルに関する情報を示します。

```
sp_replication_path 'pdb','list','table'
go
```

次のようなメッセージが表示されます。

```
Binding              Type     Path
-------------------- -------- --------------------
dbo.dt1              T        everywhere
dbo.t1               T        pdb_2
dbo.t2               T        pdb_1

(3 rows affected)
(return status = 0)
```

- **例 14 –** 分散モデルがオブジェクトバインドの場合に、バインドされたすべてのストアドプロシージャに関する情報を示します。

```
sp_replication_path 'pdb','list','sproc'
go
```

次のようなメッセージが表示されます。

```
Binding              Type     Path
-------------------- -------- --------------------
dbo.sproc1           P        pdb_1
dbo.sproc1           P        pdb_2
dbo.sproc2           P        pdb_1

(3 rows affected)
(return status = 0)
```

- **例 15 –** 分散モデルがオブジェクトバインドの場合に、バインドされたすべてのフィルタに関する情報を示します。

```
sp_replication_path 'pdb','list','filter'
go
```

フィルタが非アクティブバインド状態であることが示されます。

```
Inactive Binding     Type     Path
-------------------- -------- --------------------
dbo.F1               RF       pdb_1
dbo.F2               RF       pdb_2

(2 rows affected)

(return status = 0)
```

- **例 16 –** 分散モデルがオブジェクトバインドの場合に、**sproc1** ストアドプロシージャに関する情報のみを示します。

```
sp_replication_path 'pdb','list','sproc','sproc1'
go
```

次のようなメッセージが表示されます。

```
Binding              Type     Path
-------------------- -------- --------------------
dbo.sproc1           P        pdb_2
dbo.sproc1           P        pdb_1

(2 rows affected)
(return status = 0)
```

- **例 17 –** 分散モデルがオブジェクトバインドの場合に、**F1** フィルタに関する情報のみを示します。

```
sp_replication_path 'pdb','list','filter','F1'
go
```

フィルタが非アクティブバインド状態であることが示されます。

```
Inactive Binding     Type     Path
-------------------- -------- --------------------
dbo.F1               RF       pdb_1


(1 row affected)

(return status = 0)
```

- **例 18 –** 分散モデルがフィルタの場合に、すべての複写パスに関する情報を表示するには:

```
sp_replication_path 'pdb','list','path'
go
```

次のようなメッセージが表示されます。

```
Path             Type  Binding      Active
---------------  ----  ----------   ------------
everywhere       T     dbo.dt1      No
pdb_1            P     dbo.sproc1   No
pdb_1            RF    dbo.F1       Yes
pdb_1            T     dbo.t2       No
pdb_2            P     dbo.sproc1   No
pdb_2            RF    dbo.F2       Yes
```

```
pdb_2             T     dbo.t1       No

(7 rows affected)


Logical Path                  Physical Path
----------------------------- ------------------------
everywhere                    pdb_1
everywhere                    pdb_2

(2 rows affected)


Physical Path                 Destination
----------------------------- ----------------------
pdb_1                         RS2
pdb_2                         RS1

(2 rows affected)
(return status = 0)
```

- **例 19 –** 分散モデルが接続の場合に、すべての複写パスに関する情報を表示するには:

```
sp_replication_path 'pdb','list','path'
go
```

次のようなメッセージが表示されます。

```
Path              Type  Binding      Active
---------------   ----  ----------   ------------
everywhere        T     dbo.dt1      No
pdb_1             P     dbo.sproc1   No
pdb_1             RF    dbo.F1       No
pdb_1             T     dbo.t2       No
pdb_2             P     dbo.sproc1   No
pdb_2             RF    dbo.F2       No
pdb_2             T     dbo.t1       No

(7 rows affected)


Logical Path                  Physical Path
----------------------------- ------------------------
everywhere                    pdb_1
everywhere                    pdb_2

(2 rows affected)


Physical Path                 Destination
----------------------------- ----------------------
pdb_1                         RS2
pdb_2                         RS1
default                       RS_Default
```

```
(2 rows affected)
(return status = 0)
```

- **例 20 –** 分散モデルがフィルタの場合に、pdb_1 物理パスに関する情報のみを表示するには:

```
sp_replication_path 'pdb','list','path','pdb_1'
go
```

次のようなメッセージが表示されます。

```
Path              Type  Binding      Active
---------------   ----  ----------   ------------
pdb_1             P     dbo.sproc1   No
pdb_1             RF    dbo.F1       Yes
pdb_1             T     dbo.t2       No

(2 rows affected)
'
Physical Path                   Destination
------------------------------- -------------------
pdb_1                           RS2

(1 row affected)
(return status = 0)
```

- **例 21 –** 分散モデルがオブジェクトバインドの場合に、"logical_1" 論理複写パスに関する情報のみを表示するには:

```
sp_replication_path 'pdb','list','path','logical_1'
go
```

次のようなメッセージが表示されます。

```
Path                  Type     Binding
--------------------- -------- -------------------------
logical_1            T        dbo.dt1

(1 rows affected)
Logical Path                    Physical Path
------------------------------- --------------------------
logical_1                      pdb_1
logical_1                      pdb_2

(2 rows affected)
Physical Path                   Destination
------------------------------- -------------------------
pdb_1                           RS2
pdb_2                           RS1

(2 rows affected)
(return status = 0)
```

注意： 論理パスの基になる物理パスも表示されます。

使用法

- オブジェクトをパスにバインドする前に、プライマリデータベースと Replication Server との間に代替プライマリコネクションを作成し、そのコネクションをプライマリデータベースから Replication Serve への代替 RepAgent 複写パスに関連付ける必要があります。「Replication Server」の「パフォーマンスチューニング」の「Multi-Path Replication」を参照してください。
- テーブルとストアドプロシージャを、Multi-Path Replication に作成する物理パスまたは論理パスにバインドできます。
- 複写パスにバインドするオブジェクトは複写中、常に同じパスに従います。
- テーブル、ストアドプロシージャ、フィルタは複数のパスにバインドできます。 複写中、テーブル、ストアドプロシージャ、フィルタは指定したすべてのパスを介してレプリケートされます。
- フィルタ別分散モデルを設定して、テーブルまたはストアドプロシージャを複写パスにバインドできます。 同様に、オブジェクトバインド別分散モデルを設定して、フィルタをパスにバインドできます。 ただし、バインドされたオブジェクトまたはフィルタの複写は、対応する分散モデルを有効にするまで有効になりません。 たとえば、分散モデルがフィルタの場合に、テーブルを複写パスにバインドしようとすると:

  ```
  > sp_replication_path primdb, bind, 'table', 'T2', 'PP1'
  go
  ```

  次のようなメッセージが表示されます。

  ```
  Warning: Under the current 'filter' distribution model this
  binding will be ignored.
  The table 'T2' is bound to path 'PP1'.
  ```

- "n/a" という名前の代替パスを追加することはできません。

パーミッション

sp_replication_path には、“sa” または “dbo” パーミッションか **replication_role** が必要です。

sp_reptostandby

スタンバイデータベースに複写するようデータベースをマーク付けしたり、そのマーク付けを解除したりします。サポートされるスキーマ変更およびデータ変更のユーザテーブルへの複写を有効にします。

構文

```
sp_reptostandby dbname [, 'L1' | 'all' | 'none'] [, use_index]
```

パラメータ

- **dbname –** アクティブデータベースの名前です。
- **L1 –** スキーマ複写機能セットのサポートレベルを Adaptive Server バージョン 12.0 で最初に導入したサポートレベルに設定します。Adaptive Server を上位のサポートレベル (**L2**、**L3** など) が実装されている新しいバージョンにアップグレードする場合は、サポートレベルは Adaptive Server バージョン 12.0 のサポートレベルのままになります。現在のところ、Adaptive Server バージョン 12.0 以降にはサポートレベル **L1** のみが実装されています。
- **all –** スキーマ複写機能セットのサポートレベルを現在の Adaptive Server で実装されている最高のサポートレベルに設定します。Adaptive Server を新しいバージョンにアップグレードすると、新しいバージョンで実装されている最高のサポートレベルが自動的に有効になります。
- **none –** すべてのデータベーステーブルの複写のマーク付けを解除して、データおよびスキーマのスタンバイデータベースへの複写をオフに切り替えます。

  注意： **none** キーワードを指定した **sp_reptostandby** を使用して複写をオフに切り替えると、Adaptive Server はすべてのユーザテーブルを排他モードでロックして、複写のマーク付けを解除されたすべてのテーブルについてログレコードを書き込みます。データベースに大量のユーザテーブルがある場合、この処理は時間がかかることがあります。
- **use_index –** `text`、`unitext`、`image`、または `rawobjects` カラムに複写のインデックスを使用するようにデータベースにマーク付けします。明示的に複写対象としてマーク付けされていないそれらのテーブルには、内部インデックスが作成されます。

  use_index オプションは、15.7 SP100 より前のバージョンの Adaptive Server で作成した LOB カラムを含むテーブルでのみ有効になります。 Adaptive Server 15.7 SP100 では、**use_index** は廃止されました。これは、RepAgent が LOB 列の複写に必要とする情報がバックリンクポインタの形ですでに入手可能であるためで、したがってバージョン 15.7 SP100 以降にデータベースをアップグレードすると、RepAgent は **use_index** を無視します。

例

- **例 1 –** *pubs2* の複写ステータスを **all** に設定し、テキストポインタおよびイメージポインタにグローバルインデックスを作成します。

```
sp_reptostandby pubs2,'all'
```

- **例 2 –** SQL 文の複写ステータスをデータベースレベルで表示します。

```
1> sp_reptostandby pubs2
2> go

The replication status for database 'pubs2' is 'ALL'.
```

```
The replication mode for database 'pubs2' is ' udis'.
(return status = 0)
```

使用法

- **sp_reptostandby** は、Adaptive Server バージョン 11.5 以降のデータベースで使用します。アクティブデータベースとスタンバイデータベースの RepAgent も有効にする必要があります。
- データ操作言語 (DML) コマンド、サポートされているデータ定義言語 (DDL) コマンド、サポートされているシステムプロシージャをスタンバイデータベースにコピーします。
- データベースが master データベースの場合、ユーザデータベースの複写でサポートされている DDL コマンドとシステムプロシージャは master データベースの複写ではサポートされません。
  DDL コマンドまたはシステムプロシージャにパスワード情報が含まれている場合、パスワード情報は送信元 ASE システムテーブルに格納されている暗号化テキストのパスワード値を使用して複写環境を介して送信されます。
- **sp_reptostandby** は、ウォームスタンバイデータベースに複写するようデータベースをマーク付けします。このプロシージャでは、レプリケートデータベースへの複写は有効にはなりません。
- **sp_reptostandby** が実行され、ウォームスタンバイが有効になった後は、複写ステータスを **never** に設定することにより、個々のデータベーステーブルの複写を個別にオフにできます。**isql** セッションに対する DDL と DML コマンドおよびプロシージャの複写を制御するには、**set replication** コマンドを使用してください。
- デフォルトでは、**sp_reptostandby** は *text*、*unitext*、または *image* データを **replicate_if_changed** とマーク付けします。このステータスを **always_replicate** または **do_not_replicate** に変更することはできません。
- ウォームスタンバイアプリケーションに通常の複写が含まれる場合、*text*、*unitext*、または *image* データのカラムは、**always_replicate** または **replicate_if_changed** として扱われることがあります。
  - **sp_setreptable** によってマーク付けされた *text*、*unitext*、または *image* カラムが **always_replicate** (デフォルト) として指定されている場合、すべての *text*、*unitext*、または *image* カラムは **always_replicate** として扱われます。
  - *text*、*unitext*、または *image* カラムを **sp_setrepcol** で **do_not_replicate** または **replicate_if_changed** に指定すると、すべての *text*、*unitext*、または *image* カラムは **replicate_if_changed** として扱われます。
- Transact-SQL の **writetext** コマンドを複写するには、データベースが LOB データを格納するテキストページを指すデータローにアクセスする必要があります。このデータローへのアクセスを許可するために、Adaptive Server は最初のテキストページのバックリンクポインタまたは複写用に作成されたインデックスを

使用します。 カラム、テーブル、またはデータベースレベルでインデックスを作成するプロセスでは、複写をサポートするための情報を提供する負荷の高い操作が必要になります。

以前のバージョンからアップグレードしたのではない、Adaptive Server version 15.7 SP100 以降のデータベースでは、Adaptive Server はデフォルトでデータベースへの LOB バックリンクポインタを作成して管理するので、**sp_reptostandby** がすぐに有効になります。 したがって、テーブルの複写を設定するときにインデックスを作成する必要はありません。 LOB カラムの複写に必要な情報がバックリンクポインタの形ですでに利用できる場合、Adaptive Server は **use_index** パラメータを無視します。

ただし、Adaptive Server 15.7 SP100 より前のバージョンで作成したデータベースを使用している場合や、そのデータベースからアップグレードした場合は、インデックスの作成によって複写の設定に時間がかかることがあります。 処理時間を短縮するには、該当するレベル (カラム、テーブル、またはデータベース)で **dbcc shrinkdb_setup** を実行してバックリンクポインタを作成し、バックリンクを最新の状態にします。

dbcc shrinkdb_setup は、以前に **use_index** でマーク付けしたカラム、テーブル、またはデータベースの複写インデックスを suspect (疑わしい) としてマーク付けします。 これらのオブジェクトのインデックスは、**dbcc shrinkdb_setup**.の実行後には必要ないため、**dbcc reindex** を使用して削除できます。

- Adaptive Server 15.7 SP100 では、**use_index** は廃止されました。 Adaptive Server 15.7 SP100 より前のバージョンでは、**use_index** を使用すると、ノンクラスタードインデックスの作成中に共有テーブルロックが保持されます。
- **sp_reptostandby** を **none** オプションを指定して 実行する場合、複写用のインデックスを使用するようにデータベースが最初にマーク付けされていると、複写用に作成されたこれらすべてのインデックスが削除されます。

必要条件と制限

- スタンバイデータベースはアクティブデータベースと同じか、それ以降のリリースレベルでなければなりません。両方のデータベースのディスクの割り付け、セグメント名、ロールは同じでなければなりません。詳細については、『Adaptive Server Enterprise システム管理ガイド』を参照してください。
- ログイン情報は、スタンバイデータベースに複写されません。
- 別のデータベースの名前が指定されたコマンドまたはプロシージャの複写は、そのデータベースがスタンバイサーバにない場合は失敗します。
- **create table** などのサポートされている DDL コマンドに、ローカル変数を含めることはできません。
- 次のコマンドは、スタンバイデータベースにコピーされません。
  - **select into** と **update statistics**

    - **sp_dboption**、**sp_configure** などのデータベースオプションまたは設定オプション
- データベースが master データベースの場合
    - ユーザテーブルとユーザストアドプロシージャは複写されません。
    - ターゲットデータベースは **dump** または **load** を使用してマテリアライズできません。矛盾を解決するようにデータを処理できる **bcp** などの他の方法を使用してください。
    - 送信元 ASE サーバおよび送信先 ASE サーバの両方で master データベースの複写機能をサポートしている必要があります。
    - 送信元 ASE サーバと送信先 ASE サーバの両方で、同じハードウェアアーキテクチャタイプ (32 ビットバージョンと 64 ビットバージョン間でも互換性があります)、および同じオペレーティングシステム (異なるバージョンにおいても互換性があります) を使用している必要があります。
- master データベースを複写する場合、次のシステムプロシージャは、master データベース内で実行する必要があります。
    - **sp_addlogin**
    - **sp_defaultdb**
    - **sp_defaultlanguage**
    - **sp_displaylevel**
    - **sp_droplogin**
    - **sp_locklogin**
    - **sp_modifylogin**
- **drop index** を使用して、*text*、*unitext*、*image*、または *rawobject* の複写用に作成したインデックスを手動で削除することはできません。複写インデックスのステータスを変更するには、サポートされている複写ストアドプロシージャ **sp_reptostandby**、**sp_setreptable**、**sp_setrepcol** のみを使用できます。

パーミッション

sp_reptostandby には、“sa” または “dbo” パーミッション、もしくは **replication_role** が必要です。

参照：

- set replication (605 ページ)
- sp_setrepcol (664 ページ)
- sp_setreptable (675 ページ)
- sp_setrepproc (673 ページ)

サポートされている DDL コマンドとシステムプロシージャ

sp_reptostandby を使用して複写を有効にしたときに Replication Server がスタンバイデータベースで再生成する DDL コマンド、Transact-SQL コマンド、Adaptive Server システムプロシージャです。

Adaptive Server 12.5 以降で複写がサポートされるコマンドとストアドプロシージャについては、アスタリスク (*) が付いています。

サポートされている DDL コマンドは、次のとおりです。

- **alter encryption key**
- **alter key**
- **alter login**
- **alter login profile**
- **alter...modify owner** - Replication Server は、所有者の異なるテーブルを別のテーブルとして扱います。**alter...modify owner** を使用して Adaptive Server でレプリケートされたテーブルの所有者を変更するには、該当するテーブル複写定義も変更する必要があります。『Replication Server 管理ガイド 第 1 巻』の「複写テーブルの管理」の「複写定義の修正」の「複写定義の変更」の「複写定義の可能な変更」の「Changing Table Owner」を参照してください。
- **alter {precomputed result set | materialized view}**
- **alter table**
- **create default**
- **create encryption key**
- **create function**
- **create index**
- **create key**
- **create login**
- **create login profile**
- **create plan***
- **create {precomputed result set | materialized view}**
- **create procedure**
- **create rule**
- **create schema***
- **create table**
- **create trigger**
- **create view**
- **drop default**
- **drop function**
- **drop login**

- **drop login profile**
- **drop index**
- **drop {precomputed result set | materialized view}**
- **drop procedure**
- **drop rule**
- **drop table**
- **drop trigger**
- **drop view**
- **grant**
- **installjava*** - **installjava** の複写は、MSA 環境ではサポートされていません。
- **refresh {precomputed result set | materialized view}**
- **remove java***
- **revoke**
- **truncate {precomputed result set | materialized view}**

サポートされているシステムプロシージャは、次のとおりです。

- **sp_add_qpgroup***
- **sp_addalias**
- **sp_addgroup**
- **sp_addmessage**
- **sp_addtype**
- **sp_adduser**
- **sp_bindefault**
- **sp_bindmsg**
- **sp_bindrule**
- **sp_cachestrategy**
- **sp_changegroup**
- **sp_chgattribute**
- **sp_commonkey**
- **sp_config_rep_agent**
- **sp_drop_all_qplans***
- **sp_drop_qpgroup***
- **sp_dropalias**
- **sp_dropgroup**
- **sp_dropkey**
- **sp_dropmessage**
- **sp_droptype**
- **sp_dropuser**
- **sp_encryption**

- **sp_export_qpgroup***
- **sp_foreignkey**
- **sp_hidetext**
- **sp_import_qpgroup***
- **sp_primarykey**
- **sp_procxmode**
- **sp_recompile**
- **sp_rename**
- **sp_rename_qpgroup***
- **sp_replication_path**
- **sp_restore_system_role**
- **sp_setrepcol**
- **sp_setrepdefmode**
- **sp_setrepproc**
- **sp_setreptable**
- **sp_unbindefault**
- **sp_unbindmsg**
- **sp_unbindrule**

master データベースの複写でサポートされている DDL コマンドセットとシステムプロシージャは、ユーザデータベースの複写でサポートされているセットとは異なります。

データベースが master データベースの場合、次の DDL コマンドがサポートされています。

- **alter role**
- **create role**
- **drop role**
- **grant role**
- **revoke role**

データベースが master データベースの場合、次のシステムプロシージャがサポートされています。

- **sp_addexternlogin**
- **sp_addlogin**
- **sp_addremotelogin**
- **sp_addserver**
- **sp_defaultdb**
- **sp_defaultlanguage**
- **sp_displaylevel**

- **sp_dropexternlogin**
- **sp_droplogin**
- **sp_dropremotelogin**
- **sp_dropserver**
- **sp_locklogin**
- **sp_maplogin**
- **sp_modifylogin**
- **sp_password**
- **sp_passwordpolicy - allow password downgrade** を除くすべてのオプションで複写されます。
- **sp_role**

sp_setrepcol

text、*unitext*、または *image* カラムの複写ステータスを設定または表示します。

構文

```
sp_setrepcol table_name [, {column_name | null}
     [, {do_not_replicate | always_replicate |
replicate_if_changed}]]
 [, use_index]
```

パラメータ

- **table_name –** 複写テーブルの名前です。**sp_setrepcol** を実行する前に、**sp_setreptable** を使用してテーブルの複写を有効にする必要があります。
- **column_name –** テーブル内の *text*、*unitext*、または *image* カラムの名前です。テーブルのすべての *text*、*unitext*、または *image* カラムに複写ステータスを設定するには、カラム名に **null** を指定します。
- **do_not_replicate –** *text*、*unitext*、または *image* カラムの複写情報をログに記録しないように指定します。複写用のインデックスを使用するように以前にカラムをマーク付けしている場合は、**do_not_replicate** を設定すると、インデックスが削除されます。
- **always_replicate –** このパラメータが指定されると、ローのいずれかのカラムが変更されたときに、Adaptive Server が *text*、*unitext*、または *image* カラムの複写情報をログに記録します。このステータスは、変更されていない *text*、*unitext*、または *image* カラムも複写されるためオーバヘッドが増加しますが、ローのマイグレーションやノンアトミックマテリアライゼーション中の変更によって発生する、データの不整合を防止します。
- **replicate_if_changed –** このパラメータが指定されると、Adaptive Server は、*text*、*unitext*、または *image* カラムのデータが変更されたときだけ、これらのカ

ラムの複写情報をログに記録します。このステータスではオーバヘッドは減少しますが、ローのマイグレーションやノンアトミックマテリアライゼーション中の変更のために、データの不整合が発生する可能性があります。

- **use_index** – `text`、`unitext`、`image`、または `rawobjects` カラムに複写のインデックスを使用するようにデータベースにマーク付けします。明示的に複写対象としてマーク付けされていないそれらのテーブルには、内部インデックスが作成されます。

  use_index オプションは、15.7 SP100 より前のバージョンの Adaptive Server で作成した LOB カラムを含むテーブルでのみ有効になります。 Adaptive Server 15.7 SP100 では、**use_index** は廃止されました。これは、RepAgent が LOB 列の複写に必要とする情報がバックリンクポインタの形ですでに入手可能であるためで、したがってバージョン 15.7 SP100 以降にデータベースをアップグレードすると、RepAgent は **use_index** を無視します。

例

- **例 1 –** すべての *text*、*unitext*、または *image* カラム (*au_pix* テーブル) に関する複写ステータスを表示します。*au_pix* は、**sp_setreptable** で複写するようマーク付けされている必要があります。

  ```
  sp_setrepcol au_pix
  ```

- **例 2 –** *カラム* (*au_pix* テーブル) の複写ステータスを表示します。pic は *text*、*unitext*、または *image* データ型カラムである必要があります。

  ```
  sp_setrepcol au_pix, pic
  ```

- **例 3 –** *au_pix* テーブルの *pic* カラム (*image* データ型) のステータスを **replicate_if_changed** に設定します。このテーブルは *pubs2* データベースにあり、他に *text*、*unitext*、または *image* カラムはありません。

  ```
  sp_setrepcol au_pix, pic, replicate_if_changed
  ```

- **例 4 –** *au_pix* テーブルのすべての *text*、*unitext*、または *image* カラムのステータスを、**replicate_if_changed** に設定します。

  ```
  sp_setrepcol au_pix, null, replicate_if_changed
  ```

- **例 5 –** 圧縮 LOB カラムの複写を無効にします。

  ```
  sp_setrepcol table_name, lob_column_name, 'do_not_replicate'
  ```

使用法

- **sp_setrepcol** で *text*、*unitext*、または *image* カラムの複写方法を指定するのは、**sp_setreptable** でテーブルの複写を有効にした後で行います。
- テーブル名を指定して **sp_setrepcol** を実行し、そのテーブルのすべての *text*、*unitext*、または *image* カラムの複写ステータスを表示したり、テーブル名と

text、*unitext*、または *image* カラム名を指定してそのカラムの複写ステータスを表示したりすることもできます。

- **replicate_if_changed** オプションを指定すると、*text*、*unitext*、または *image* カラムを複写するオーバヘッドが減少しますが、次のような制限や注意事項があります。
  - あるカラムに **replicate_if_changed** ステータスを指定した場合、そのカラムを含む複写定義のステータスも **replicate_if_changed** でなければなりません。
  - カラムの複写ステータスを **replicate_if_changed** に設定した場合、そのカラムを含む複写定義のオートコレクションを "on" に設定できません。
  - 複写ステータスを **replicate_if_changed** に設定した *text*、*unitext*、または *image* カラムがある場合に、ノンアトミックサブスクリプションマテリアライゼーションを使用すると、Replication Server はエラーログファイルにメッセージを表示します。これは、サブスクリプションマテリアライゼーション中にアプリケーションがプライマリテーブルを変更した場合、データの一貫性が失われる可能性があるという警告メッセージです。
  - サブスクリプションへのローのマイグレートを許可するアプリケーションで、いずれかの *text*、*unitext*、または *image* カラムの複写ステータスを **replicate_if_changed** に設定している場合、ローがサブスクリプションにマイグレートして *text* または *image* データが見つからないと、Replication Server はエラーログに警告メッセージを表示します。

    replicate_if_changed ステータスの *text*、*unitext*、または *image* カラムがプライマリテーブルでの更新オペレーションで変更されておらず、更新の結果ローがサブスクリプションにマイグレートすると、レプリケートテーブルに挿入されたローは *text*、*unitext*、または *image* データを失うことになります。この場合は、**rs_subcmp** プログラムを実行してレプリケートテーブルとプライマリテーブルのデータを一致させます。

    サブスクリプションに **where** 句が含まれている場合、ローのマイグレーションが発生する可能性があります。サブスクリプションの **where** 句に指定されたカラムを更新すると、ローがサブスクリプションに対して有効になる、つまりサブスクリプションにマイグレートします。

    この状態が発生すると、Replication Server はレプリケートデータベースで **insert** を実行する必要があります。**insert** には、プライマリデータベースで変更されていない *text*、*unitext*、または *image* カラムも含めて、すべてのカラムの値が必要です。
- テーブルが **sp_reptostandby** でマーク付けされている場合、**sp_setrepcol** を使用して *text*、*unitext*、または *image* カラムの複写ステータスを変更することはできません。この場合、*text*、*unitext*、*image* カラムは、常に **replicate_if_changed** として扱われます。
- ウォームスタンバイアプリケーションに通常の複写が含まれ、テーブルを **sp_reptostandby** と **sp_setreptable** でマーク付けしている場合、*text*、*unitext*、ま

たは *image* データのカラムは **always_replicate** または **replicate_if_changed** として扱われることがあります。

  - **sp_setreptable** によってマーク付けされた *text*、*unitext*、または *image* カラムが **always_replicate** (デフォルト) として指定されている場合、すべての *text*、*unitext*、および *image* カラムは **always_replicate** として扱われます。
  - *text*、*unitext*、または *image* カラムを **sp_setrepcol** で **do_not_replicate** または **replicate_if_changed** に指定すると、すべての *text*、*unitext*、または *image* カラムは **replicate_if_changed** として扱われます。

- インデックスステータスの優先度は、カラム、テーブル、データベースの順番になります。テーブルが *text*、*unitext*、*image*、または *rawobject* カラムのインデックスを使用するようにマーク付けされている場合でも、そのいずれかのカラムのインデックスを使用しないときには、カラムのステータスはテーブルのステータスに優先されます。
- **drop index** を使用して、*text*、*unitext*、*image*、または *rawobject* の複写用に作成したインデックスを手動で削除することはできません。複写インデックスのステータスを変更するには、サポートされている複写ストアドプロシージャ **sp_reptostandby**、**sp_setreptable**、**sp_setrepcol** のみを使用できます。
- Transact-SQL の **writetext** コマンドを複写するには、データベースが LOB データを格納するテキストページを指すデータローにアクセスする必要があります。このデータローへのアクセスを許可するために、Adaptive Server は最初のテキストページのバックリンクポインタまたは複写用に作成されたインデックスを使用します。 カラム、テーブル、またはデータベースレベルでインデックスを作成するプロセスでは、複写をサポートするための情報を提供する負荷の高い操作が必要になります。

  以前のバージョンからアップグレードしたのではない、Adaptive Server version 15.7 SP100 以降のデータベースでは、Adaptive Server はデフォルトでデータベースへの LOB バックリンクポインタを作成して管理するので、**sp_reptostandby** がすぐに有効になります。 したがって、テーブルの複写を設定するときにインデックスを作成する必要はありません。 LOB カラムの複写に必要な情報がバックリンクポインタの形ですでに利用できる場合、Adaptive Server は **use_index** パラメータを無視します。

  ただし、Adaptive Server 15.7 SP100 より前のバージョンで作成したデータベースを使用している場合や、そのデータベースからアップグレードした場合は、インデックスの作成によって複写の設定に時間がかかることがあります。 処理時間を短縮するには、該当するレベル (カラム、テーブル、またはデータベース)で **dbcc shrinkdb_setup** を実行してバックリンクポインタを作成し、バックリンクを最新の状態にします。

  dbcc shrinkdb_setup は、以前に **use_index** でマーク付けしたカラム、テーブル、またはデータベースの複写インデックスを suspect (疑わしい) としてマーク付け

します。 これらのオブジェクトのインデックスは、**dbcc shrinkdb_setup**.の実行後には必要ないため、**dbcc reindex** を使用して削除できます。

- Adaptive Server 15.7 SP100 では、**use_index** は廃止されました。 Adaptive Server 15.7 SP100 より前のバージョンでは、**use_index** を使用すると、ノンクラスタードインデックスの作成中に共有テーブルロックが保持されます。

パーミッション

sp_setrepcol には、“sa” または “dbo” パーミッション、もしくは **replication_role** が必要です。

参照：

- sp_reptostandby (656 ページ)
- sp_setreptable (675 ページ)

sp_setrepdbmode

データベースレベルの SQL 文の複写を 1 つ以上の特定の DML オペレーションタイプに対して有効または無効にします。

構文

```
sp_setrepdbmode dbname [, "option [option [...]]" [, "on" | "off"]
['threshold', 'value']
```

```
option ::= { U | D | I | S }
```

パラメータ

- **dbname –** SQL 文の複写を有効にするデータベースの名前です。
- **option –** 次の DML オペレーションの任意の組み合わせです。
  - U - **update**
  - D - **delete**
  - I - **insert select**
  - S - **select into**

  データベースの複写モードを **UDIS** の任意の組み合わせに設定すると、RepAgent は、個々のログレコードと Replication Server が SQL 文を作成するために必要な情報の両方を送信します。
- **on –** 指定した DML オペレーションの SQL 複写を有効にします。

- **off** – データベースレベルでの SQL 文の複写を、すべてのタイプの DML オペレーションに対して無効にします。*option* で指定したオペレーションは関係ありません。
- **'threshold', 'value'** – SQL 文の複写がアクティブになるまでに、複写される SQL 文が影響を与える必要がある最小ロー数を指定します。 *value* に 0 を指定すると、スレッショルドはデフォルト値の 50 ローにリセットされます。

例

- **例 1 – delete** 文と **select into** 文を複写します。

```
sp_setrepdbmode pdb, 'DS', 'on'
```

- **例 2** – 現在の SQL の複写設定を表示します。

```
1> sp_setrepdbmode pdb1
2> go
```

```
The replication mode for database 'pdb1' is 'us'.
(return status = 0)
```

- **例 3** – データベースレベルですべての SQL 文の複写を無効にするには、次のように指定します。

```
sp_setrepdbmode pdb, 'D', 'off'
```

- **例 4** – スレッショルド値を 100 ローで設定します。

```
sp_setrepdbmode pubs2, 'threshold', '100'
go
```

- **例 5** – 次の例では、*pubs2* データベースと *table1* テーブルについて異なるスレッショルドをデータベースレベルとテーブルレベルで設定する方法を示します。

  1. データベースレベルでのスレッショルドをデフォルト値の 50 ローにリセットします。

     ```
     sp_setrepdbmode pubs2, 'threshold', '0'
     go
     ```

  2. **update**、**delete**、**insert**、および **select into** の各オペレーションの SQL 文の複写を *pubs2* に対して有効にします。

     ```
     sp_setrepdbmode pubs2, 'udis', 'on'
     go
     ```

  3. *table1* (*pubs2*) の SQL 文の複写をトリガします (**update**、**delete**、**insert**、および **select into** の各オペレーションが *table1* で実行され、1,000 を超えるローに影響を及ぼすときのみ)。

     ```
     sp_setrepdefmode table1, 'threshold', '1000'
     go
     ```

- **例 6** – 次の例では、*pubs2* のスレッショルドをデータベース・レベルで定義すると同時に、*table1* や *table2* などのテーブルに対してさまざまなオペレーションを定義する方法を示します。

1. データ操作言語 (DML) 文が 100 を超えるローに影響を及ぼすときには SQL 文の複写をトリガするようスレッショルドをデータベースレベルで設定します。

```
sp_setrepdbmode pubs2, 'threshold', '100'
go
```

2. SQL 文の複写を使用してオペレーションを複写させる 2 つの特定のテーブルについて、別のオペレーションセットを定義します。**update**、**delete**、および **insert** の各オペレーションは *table1* に対するもの、delete オペレーションは *table2* に対するものです。

```
sp_setrepdefmode table1, 'udi', 'on'
go
sp_setrepdefmode table2, 'd' 'on'
go
```

この例では、**delete** オペレーションが *table2* に対して実行されるかまたは任意の DML が *table1* 上で実行される場合、データベースレベルで定義されたスレッショルド 100 ローに達すると、SQL 文の複写がトリガされます。

使用法

- SQL 文の複写をデータベースレベルで設定できるのは、 **sp_reptostandby** が **ALL** または **L1** に設定され、データベースが複写されるようにマーク付けされている場合のみです。
- デフォルトのスレッショルドは 50 ローです。つまり、DML 文が少なくとも 51 ローに影響を与えると、Adaptive Server は SQL 文の複写を使用します。 デフォルトのスレッショルドを使用するには、**threshold** パラメータを 0 に設定します。 **threshold** パラメータの範囲は 0 から 10,000 です。
- データベースレベルで複写を設定すると同時に、SQL 文の複写のデータベースレベルでのスレッショルドを設定することができます。例：

```
sp_reptostandby pubs2, 'none'
go
sp_setrepdbmode pubs2, 'threshold'
go
```

一方、データベースレベルで複写を設定すると同時に、データベースレベルでオペレーションを定義することはできません。データベースレベルの SQL 文の複写には、データベース全体が複写されることが必要であり、オペレーションのみを複写することはできないためです。たとえば、次は実行できません。

```
sp_reptostandby pubs2, 'none'
go
sp_setrepdbmode pubs2, 'udis', 'on'
go
```

- セッションレベルで設定したスレッショルドは、テーブルレベルとデータベースレベルのスレッショルドよりも優先されます。テーブルレベルで設定したス

レッショルドは、データベースレベルで設定したスレッショルドよりも優先されます。

参照：

- set repmode (606 ページ)
- sp_setrepdefmode (671 ページ)
- set repthreshold (608 ページ)

sp_setrepdefmode

複写するようマーク付けされたテーブルの所有者ステータスを変更または表示し、テーブルレベルの SQL 文の複写を特定の DML オペレーションに対して有効または無効にします。

構文

```
sp_setrepdefmode table_name [, 'owner_on' | 'owner_off' |
'SQLDML_option [SQLDML_option [ ...]]' [, 'on' | 'off' | 'never' ] |
'threshold', 'value']
```

```
SQLDML_option ::= { U | D | I }
```

パラメータ

- **table_name –** 現在のデータベース内の、**sp_setreptable** で複写するようマーク付けされているテーブルの名前です。
- **owner_on –** テーブルが複写されるようマーク付けされるときに、テーブル名と所有者名の両方が考慮されるように、所有者ステータスを変更します。同じ名前で所有者が異なる複数のテーブルの複写を可能にします。
- **owner_off –** テーブルが複写されるようマーク付けされるときに、テーブル名だけが考慮されるように所有者ステータスを変更します。
- **SQLDML_option –** 次の DML オペレーションのいずれかです。
  - U - **update**
  - D - **delete**
  - I - **insert select**

  テーブルの複写モードを **UDI** の任意の組み合わせに設定すると、RepAgent は、指定された DML オペレーションでの SQL 文の複写を有効にするための追加の情報を送信します。
- **on –** 指定した DML オペレーションの SQL 複写を有効にします。
- **off –** *option* で指定した文に関係なく、テーブルレベルでの SQL 文の複写設定を解除し、データベースレベルの設定に従います。

- **never** – データベースの設定や、UDI パラメータが指定されているかどうかに関係なく SQL 文の複写を無効にします。
- **'threshold', 'value'** – SQL 文の複写がアクティブになるまでに、複写される SQL 文が影響を与える必要がある最小ロー数を指定します。

例

- **例 1** – SQL 文の複写を、**update**、**delete**、および **insert select** の各オペレーション (テーブル *t* での実行) について有効にします。

```
1> sp_setrepdefmode t, 'UDI', 'on'
2> go
```

- **例 2** – スレッショルドを 10 に設定します。Adaptive Server は SQL の複写をテーブル *t* に対して使用します (DML 文が少なくとも 11 ローに影響を与える場合)。

```
sp_setrepdefmode t, 'threshold', '10'
```

- **例 3** – SQL の複写設定およびテーブル *rs_ticket_history* の所有者ステータスを表示します。

```
1> sp_setrepdefmode rs_ticket_history,'udi'
2> go
```

```
The replication status for 'rs_ticket_history' is
currently  owner_off, 'udi'.
The replication threshold for table 'rs_ticket_history'
is '0'.
(return status = 0)
```

- **例 4** – スレッショルドをデフォルト値に設定します。

```
sp_setrepdefmode t, 'threshold', '0'
```

使用法

- **sp_setrepdefmode** は、RepAgent が有効な Adaptive Server データベースに使用します。
- テーブル名だけを指定して **sp_setrepdefmode** を実行すると、SQL の複写設定とテーブルの所有者ステータスが表示されます。
- テーブルのモードを変更するには **sp_setrepdefmode** を使用してください。**sp_setreptable** ではテーブルの所有者モードは変更できません。
- **sp_setrepdefmode** オプションが指定され、現在のテーブルのモードが “owner on” である場合、**sp_setrepdefmode** は、**owner_off** モードのすべての複写テーブルでテーブル名がユニークかどうかをチェックします。名前がユニークである場合には、**sp_setrepdefmode** により、テーブルのモードが **owner off** に変更されます。名前がユニークでない場合には、プロシージャは失敗します。
- デフォルトのスレッショルドは 50 ローです。つまり、DML 文が少なくとも 51 ローに影響を与えると、Adaptive Server は SQL 文の複写を使用します。デフォ

ルトのスレッショルドを使用するには、**threshold** パラメータを 0 に設定します。**threshold** パラメータの範囲は 0 ～ 10,000 です。

パーミッション

sp_setrepdefmode には、“sa” または “dbo” パーミッション、もしくは **eplication_role** が必要です。

参照：

- set repmode (606 ページ)
- sp_setreptable (675 ページ)
- sp_setrepdbmode (668 ページ)
- set repthreshold (608 ページ)

sp_setrepproc

ストアドプロシージャの複写を有効または無効にします。また、ストアドプロシージャの現在の複写ステータスを表示します。

構文

```
sp_setrepproc [proc_name [,'false' | 'table' |
'function' [,'log_current' | 'log_sproc']]]
```

パラメータ

- **proc_name –** 現在のデータベースにあるストアドプロシージャ名です。
- **false –** ストアドプロシージャの複写を無効にします。
- **table –** テーブル複写定義に関連するストアドプロシージャの複写を有効にします。
- **function –** ファンクション複写定義に関連するストアドプロシージャの複写を有効にします。
- **log_current –** 複写しているストアドプロシージャの実行結果を、複写されたストアドプロシージャがあるデータベースではなく、現在のデータベースに記録します。
- **log_sproc –** 複写しているストアドプロシージャの実行結果を、現在のデータベースではなく、複写されたストアドプロシージャがあるデータベースに記録します。**log_sproc** がデフォルトです。

例

- **例 1 –** 現在のデータベース内のすべてのストアドプロシージャの複写ステータスを表示します。各プロシージャについて、複写が有効かどうか、ファンクション複写定義またはテーブル複写定義のどちらを使用した複写が有効かを示します。

```
sp_setrepproc
```

- **例 2 – upd_pubs** ストアドプロシージャの複写ステータスを表示します。ストアドプロシージャについて、複写が有効かどうか、ファンクション複写定義またはテーブル複写定義のどちらを使用した複写が有効かを示します。

```
sp_setrepproc upd_pubs
```

- **例 3 –** ファンクション複写定義で使用する **upd_pubs** ストアドプロシージャの複写を有効にします。**upd_pubs** の実行結果は、**upd_pubs** があるデータベースに記録されます。

```
sp_setrepproc upd_pubs, 'function'
```

- **例 4 –** テーブル複写定義で使用する **upd_pubs** ストアドプロシージャの複写を有効にします。**upd_pubs** の実行結果は、**upd_pubs** があるデータベースに記録されます。

```
sp_setrepproc upd_pubs, 'table'
```

- **例 5 –** ファンクション複写定義で使用する upd_pubs ストアドプロシージャの複写を有効にします。**upd_pubs** の実行結果は、現在のデータベースに記録されます。

```
sp_setrepproc upd_pubs, 'function', 'log_current'
```

- **例 6 –** ファンクション複写定義で使用する **upd_publ** ストアドプロシージャの複写を有効にします。**upd_pubs** の実行結果は、**upd_pubs** があるデータベースに記録されます。

```
sp_setrepproc upd_pubs, 'function', 'log_sproc'
```

使用法

- データベース内のすべての複写ストアドプロシージャを表示するには、**sp_setrepproc** をパラメータなしで使用します。
- 特定のストアドプロシージャの現在の複写ステータスを表示するには、**sp_setrepproc***proc_name* を、他のパラメータを指定しないで使用します。
- Adaptive Server version 11.5 以降を使用している場合、ユーザストアドプロシージャを **sp_setrepproc** で有効にしていると、その中で実行される、サポートされている DDL コマンドとストアドプロシージャは、スタンバイデータベースにコピーされます。

ユーザストアドプロシージャを **sp_setrepproc** で有効にしていないと、その中で実行される、サポートされている DDL コマンドとストアドプロシージャはスタンバイデータベースにコピーされません。

- Adaptive Server は複写ストアドプロシージャを実行するためにトランザクションを開始するため、プロシージャの設計時には、次の点を考慮してください。
  - 複写ストアドプロシージャが DDL コマンド (**create table** など) を含んでいる場合、データベースオプション “DDL-in-Tran” がデータベース上で有効でないかぎり、Adaptive Server Enterprise はエラーを発生します。
  - 複写ストアドプロシージャがトランザクションとトランザクションをロールバックするロールバックコマンドを含んでいる場合、ロールバックコマンドはプロシージャ全体の実行をロールバックします。
  - 外部トランザクションのため、Adaptive Server はプロシージャの実行が完了するまですべてのロックを保持します。

参照：

- sp_reptostandby (656 ページ)
- sp_setreptable (675 ページ)

sp_setreptable

Adaptive Server テーブルの複写を有効または無効にします。また、テーブルの現在の複写ステータスを表示します。

構文

```
sp_setreptable [table_name [, {'true' | 'false' | 'never'}
[, {owner_on | owner_off | null}] [, use_index]]]
```

パラメータ

- **table_name –** 複写対象としてマーク付けされているテーブルの名前です。
- **true –** データベースが複写するようマーク付けされているかどうかに関係なく、テーブルを複写するよう明示的にマーク付けします。
- **false –** 以前は複写が有効になっていたテーブルに対し、複写ステータスを無効にします。
- **never –** データベースの複写設定に関係なく、テーブルでの複写を無効にします。
- **owner_on –** テーブルが複写するようマーク付けされるときに、テーブル名と所有者名の両方が考慮されるようにテーブルのモードを設定します。同じ名前で所有者が異なる複数のテーブルの複写を可能にします。このオプションは、Adaptive Server バージョン 11.5 以降のデータベース用です。

- **owner_off** – テーブルが複写するようマーク付けされるときに、テーブル名だけが考慮されるようにテーブルのモードを設定します。デフォルト値。このオプションを指定すると、複写するようマーク付けする各テーブルの名前はユニークでなければなりません。このオプションは、Adaptive Server バージョン 11.5 以降のデータベース用です。
- **null** – owner パラメータに渡すときに、**owner_off** のデフォルト値を設定します。
- **use_index** – `text`、`unitext`、`image`、または `rawobjects` カラムに複写のインデックスを使用するようにデータベースにマーク付けします。明示的に複写対象としてマーク付けされていないそれらのテーブルには、内部インデックスが作成されます。

  use_index オプションは、15.7 SP100 より前のバージョンの Adaptive Server で作成した LOB カラムを含むテーブルでのみ有効になります。 Adaptive Server 15.7 SP100 では、**use_index** は廃止されました。これは、RepAgent が LOB 列の複写に必要とする情報がバックリンクポインタの形ですでに入手可能であるためで、したがってバージョン 15.7 SP100 以降にデータベースをアップグレードすると、RepAgent は **use_index** を無視します。

例

- **例 1** – 現在のデータベース内の、**sp_setreptable** で複写するようマーク付けされている、すべてのテーブルの複写ステータスを表示します。

```
sp_setreptable
```

- **例 2** – *publishers* テーブルの複写ステータスを表示します。

```
sp_setreptable publishers
```

- **例 3** – *publishers* テーブルの複写を有効にします。

```
sp_setreptable publishers, 'true'
```

- **例 4** – 所有者がそれぞれ異なる *publishers* という名前の複数のテーブルを複写可能にします。

```
sp_setreptable publishers, 'true', owner_on
```

- **例 5** – *publishers* という名前のテーブル (所有者 *dbo* に属し、データベース *pubs2* に格納されている) を複写します。

```
sp_setreptable 'pubs2.dbo.publishers', 'true', owner_on
```

- **例 6** – テーブル *t1* の複写ステータスを削除し、*t1* がインデックスを使用するように複写対象として最初にマーク付けされていた場合には、複写インデックスを削除します。

```
sp_setreptable t1, 'false'
```

- **例 7** – テーブル *tnever* (データベース *pdb*) での複写を無効にするには、次のように指定します。

```
sp_reptostandby pdb, 'ALL'
go
sp_setreptable tnever, 'never'
go
```

使用法

- データベース内のすべての複写テーブルを表示するには、**sp_setreptable** をパラメータなしで使用します。
- 特定のテーブルの現在の複写ステータスを表示するには、**true** または **false** を指定しないで **sp_setreptable***table_name* を使用します。
- **owner_on** オプションを指定すると、同じ名前で所有者の異なる複数のテーブルが、レプリケートデータベースおよびウォームスタンバイデータベースに複写可能となります。この場合、該当するテーブルの複写定義に所有者情報も含まれていないと、複写に失敗することがあります。
- **sp_setreptable** で複写するようマーク付けしたテーブルは、**sp_setrepdefmode** システムプロシージャで所有者モードを変更できます。
- 複写インデックスステータスの優先度は、カラム、テーブル、データベースの順番になります。たとえば、インデックスを使用して複写するようにマーク付けされているデータベースでは、テーブルのステータスがインデックスのステータスよりも優先されます。
- 1 つ以上の *text*、*unitext*、*image*、または *rawobject* カラムを含む大きなテーブルが複写対象としてマーク付けされている場合、内部処理が単一のトランザクションで実行されるため時間がかかることがあります。処理を高速化するには、**use_index** オプションを使用してすべての *text*、*unitext*、*image*、または *rawobject* カラムにグローバルノンクラスタードインデックスを作成します。
- **use_index** を使用すると、グローバルノンクラスタードインデックスの作成時に共有テーブルロックが保持されます。
- **drop index** を使用して、*text*、*unitext*、*image*、または *rawobject* の複写用に作成したインデックスを手動で削除することはできません。 複写インデックスのステータスを変更するには、サポートされている複写ストアドプロシージャ **sp_reptostandby**、**sp_setreptable**、**sp_setrepcol** のみを使用できます。
- Transact-SQL の **writetext** コマンドを複写するには、データベースが LOB データを格納するテキストページを指すデータローにアクセスする必要があります。このデータローへのアクセスを許可するために、Adaptive Server は最初のテキストページのバックリンクポインタまたは複写用に作成されたインデックスを使用します。 カラム、テーブル、またはデータベースレベルでインデックスを作成するプロセスでは、複写をサポートするための情報を提供する負荷の高い操作が必要になります。
  以前のバージョンからアップグレードしたのではない、Adaptive Server version 15.7 SP100 以降のデータベースでは、Adaptive Server はデフォルトでデータベースへの LOB バックリンクポインタを作成して管理するので、**sp_reptostandby** がすぐに有効になります。 したがって、テーブルの複写を設

定するときにインデックスを作成する必要はありません。 LOB カラムの複写に必要な情報がバックリンクポインタの形ですでに利用できる場合、Adaptive Server は **use_index** パラメータを無視します。

ただし、Adaptive Server 15.7 SP100 より前のバージョンで作成したデータベースを使用している場合や、そのデータベースからアップグレードした場合は、インデックスの作成によって複写の設定に時間がかかることがあります。 処理時間を短縮するには、該当するレベル (カラム、テーブル、またはデータベース)で **dbcc shrinkdb_setup** を実行してバックリンクポインタを作成し、バックリンクを最新の状態にします。

dbcc shrinkdb_setup は、以前に **use_index** でマーク付けしたカラム、テーブル、またはデータベースの複写インデックスを suspect (疑わしい) としてマーク付けします。 これらのオブジェクトのインデックスは、**dbcc shrinkdb_setup**.の実行後には必要ないため、**dbcc reindex** を使用して削除できます。

- Adaptive Server 15.7 SP100 では、**use_index** は廃止されました。 Adaptive Server 15.7 SP100 より前のバージョンでは、**use_index** を使用すると、ノンクラスタードインデックスの作成中に共有テーブルロックが保持されます。

パーミッション

sp_setreptable には、“sa” または “dbo” パーミッション、もしくは **replication_role** が必要です。

参照：

- sp_reptostandby (656 ページ)
- sp_setrepcol (664 ページ)
- sp_setrepdefmode (671 ページ)
- sp_setrepproc (673 ページ)

sp_start_rep_agent

指定されたデータベースの RepAgent スレッドを起動します。

構文

```
sp_start_rep_agent dbname[, {'recovery' | 'recovery_foreground’ |
'resync' | 'resync purge'
                                                                  |
'resync init'} [, 'connect_dataserver',
'connect_database'[, 'repserver_name', repserver_username',
'repserver_password']]]
```

パラメータ

- **dbname –** RepAgent を起動するデータベースの名前です。
- **recovery –** リカバリアクションを開始するためのリカバリモードで RepAgent を起動します。リカバリモードは、キューが失われた場合のキューの再構築に使用します。

  リカバリモードでは、Replication Server 名、ユーザ名、パスワードも指定できます。*sysattributes* の設定を上書きするには、これらのパラメータを指定してください。
- **recovery_foreground – recovery_foreground** には **recovery** と同じ機能があります。ただし、画面には Adaptive Server のエラーログではなくリカバリの進行状況情報が表示されます。リカバリの進行状況情報の表示が終了してコマンドプロンプトが表示されると、リカバリが完了します。
- **resync –** トランケーションポイントに変更がなく、RepAgent が最後に処理したところからトランザクションログの処理を続けることになっているときは、オプションを指定しないで再同期データベースマーカを送信します。
- **resync purge –** 再同期データベースマーカを送信するために purge オプションを指定すると、新しいインバウンドトランザクションを受け取る前にインバウンドキュー内のすべてのオープントランザクションをパージして重複の検出をリセットするよう、Replication Server に指示できます。
- **resync init –** 再同期データベースマーカを送信するために init オプションを指定すると、インバウンドキュー内のすべてのオープントランザクションをパージして重複の検出をリセットし、アウトバウンド DSI をサスペンドするよう、Replication Server に指示できます。
- **connect_dataserver –** オフラインログのリカバリに使用するデータサーバの名前です。
- **connect_database –** オフラインログのリカバリに使用するデータベースの名前です。
- **repserver_name –** RepAgent が接続する Replication Server の名前です。
- **repserver_user_name –** RepAgent が Replication Server に接続するときに使用するユーザ名です。
- **repserver_password –** RepAgent が Replication Server に接続するときに使用するパスワードです。

例

- **例 1 –** *pubs2* データベースの統合された RepAgent を起動します。RepAgent は、**sp_config_rep_agent** で指定された Replication Server に接続します。トランザクションログのスキャンを開始して、フォーマットされた LTL コマンドを Replication Server に送信します。

```
sp_start_rep_agent pubs2
```

- **例 2 –** *svr2* データサーバに接続している *pdb2* データベースの RepAgent を、リカバリモードで起動します。

```
sp_start_rep_agent pubs2 for_recovery, svr2, pdb2
```

- **例 3 –** クライアントにデータベース *db2* のリカバリの状態を出力するように RepAgent を設定します。

```
sp_start_rep_agent db2, recovery_foreground, ds, db1
```

```
RepAgent(5). Starting recovery, processing log records
between (1018, 0) and (2355, 2).
RepAgent(5). Processed 1000 log records.
RepAgent(5). Processed 2000 log records.
RepAgent(5). Processed 3000 log records.
RepAgent(5). Processed 4000 log records.
RepAgent(5). Processed 5000 log records.
RepAgent(5). Processed 6000 log records.
RepAgent(5). Processed 7000 log records.
RepAgent(5). Processed 8000 log records.
RepAgent(5). Processed 9000 log records.
RepAgent(5). Processed 10000 log records.
RepAgent(5). Processed 11000 log records.
RepAgent(5). Processed 12000 log records.
RepAgent(5). Processed 13000 log records.
RepAgent(5). Processed 14000 log records.
RepAgent(5). Processed 15000 log records.
RepAgent(5). Processed 16000 log records.
RepAgent(5). Processed 17000 log records.
RepAgent(5). Processed 18000 log records.
RepAgent(5). Processed 19000 log records.
RepAgent(5). Processed 20000 log records.
RepAgent(5). Processed 20084 log records, recovery
complete.
Replication Agent thread is started for database 'db2'.
(return status = 0)
```

使用法

- **sp_start_rep_agent** は、RepAgent が有効なデータベースに使用します。
- **sp_start_rep_agent** コマンドは、**sp_config_rep_agent** で有効にした RepAgent を起動するのに使用します。**sp_start_rep_agent** で一度起動した RepAgent は、それ以降、サーバの起動時にデータサーバがリカバリした後に自動的に起動します。
- **sp_stop_rep_agent** を使用して RepAgent を停止した後は、自動起動は無効になります。**sp_start_rep_agent** を使用して、自動起動を再度有効にしてください。
- オフラインリカバリでは、アーカイブされたトランザクションログがテンポラリリカバリデータベースにダンプされることがあります。この場合、テンポラリリカバリデータベースのトランザクションログにあるレコードをレプリケートデータベースに転送できます。テンポラリトランザクションログをスキャン

するには、テンポラリデータサーバ名とデータベース名を指定し、**recovery** または **recovery_foreground** のいずれかで **sp_start_rep_agent** を実行してください。リカバリでは、トランザクションログのスキャンを完了すると、RepAgent は停止します。次のトランザクションダンプがロードされた後、前回指定したオプションで **sp_start_rep_agent** を実行し、RepAgent を再起動してください。

パーミッション

sp_start_rep_agent には、“sa” または “dbo” パーミッション、もしくは **replication_role** が必要です。

参照：

- sp_help_rep_agent (632 ページ)
- sp_stop_rep_agent (681 ページ)

sp_stop_rep_agent

指定されたデータベースの RepAgent スレッドを停止します。

構文

```
sp_stop_rep_agent dbname[, 'nowait']
```

パラメータ

- **dbname –** RepAgent を停止するデータベースの名前です。
- **nowait –** 実行中の操作の完了を待たずに、ただちに RepAgent を停止します。

  デフォルトでは、現在のバッチが終了してから RepAgent を停止します。

例

- **例 1 –** *pubs2* データベースの統合 RepAgent を停止します。デフォルトの設定のため、現在のバッチの処理が終了してから RepAgent を停止します。

  ```
  sp_stop_rep_agent pubs2
  ```

使用法

- **sp_stop_rep_agent** は、RepAgent が有効なデータベースで使用してください。
- **sp_stop_rep_agent** を使用して RepAgent を停止した後は、サーバ起動時にデータベースがオンラインになっても、RepAgent は自動的には起動しません。自

動起動を再度有効にするには、**sp_start_rep_agent** プロシージャを実行してください。

- **sp_stop_rep_agent** は非同期のプロシージャであるため、実行に多少時間がかかることがあります。RepAgent のステータスをチェックするには、**sp_who** を使用してください。

パーミッション

sp_start_rep_agent には、“sa” または “dbo” パーミッション、もしくは **replication_role** が必要です。

参照：

- sp_config_rep_agent (614 ページ)
- sp_help_rep_agent (632 ページ)
- sp_start_rep_agent (678 ページ)

RSSD ストアドプロシージャ

Replication Server とともに使用される RSSD ストアドプロシージャを以下に示します。

rs_capacity

ステーブルキューの必要なサイズを見積もる場合に使用します。**rs_fillcaptable** ストアドプロシージャと一緒に使用してください。

構文

```
rs_capacity TranDuration, FailDuration, SaveInterval, MatRows
```

パラメータ

- **TranDuration –** 一番長いトランザクションの継続時間を秒単位で指定します。デフォルトは 5 秒です。
- **FailDuration –** 障害時にキューに情報を保持する時間を分単位で指定します。デフォルトは 60 分です。
- **SaveInterval –** メッセージの受信を確認した後に、メッセージを保持する時間を分単位で指定します。デフォルトは 1 分です。
- **MatRows –** サブスクリプションでマテリアライズされるローの数です。デフォルトは 1,000 です。

例

- **例 1 – rs_fillcaptable** ストアドプロシージャのページで説明する例の場合、次のパラメータを指定して **rs_capacity** ストアドプロシージャを使用します。

```
rs_capacity
 60,/* TranDuration maximum 60 seconds */
 360, /* FailDuration 6 hours */
 10,/* SaveInterval 10 minutes */
 3500 /* Materialize 3500 rows */
```

rs_capacity は個々のキューに必要なサイズの見積もりを返します。また、複写定義とマテリアライズするローの数によって必要なサブスクリプションマテリアライゼーションキューのサイズを見積もります。

使用法

- **rs_capacity** は、*rs_captable* テーブル (**rs_fillcaptable** ストアドプロシージャを使って作成) にあるデータを使用して、ステーブルキューの必要なサイズの見積もりを計算します。**rs_capacity** は、**rs_fillcaptable** を使用して複写定義の内容をこのテーブルに記録した後で実行してください。

参照：
- rs_fillcaptable (692 ページ)

rs_delexception

例外ログにあるトランザクションを削除します。

構文

```
rs_delexception [transaction_id]
```

パラメータ

- **transaction_id –** 削除するトランザクションの番号です。

例

- **例 1 –** 例外ログからトランザクション番号 1234 を削除します。

  ```
  rs_delexception 1234
  ```

使用法

- パラメータを指定しないと、**rs_delexception** は例外ログにあるトランザクションの情報を表示します。
- 有効な *transaction_id* を指定して **rs_delexception** を実行すると、トランザクションが削除されます。トランザクションの *transaction_id* を調べるには、パラメータを指定しないで **rs_helpexception** または **rs_delexception** を実行します。

参照：
- rs_helpexception (706 ページ)
- rs_delexception_date (685 ページ)
- rs_delexception_id (686 ページ)
- rs_delexception_range (687 ページ)

rs_delexception_date

rs_exceptscmd、rs_exceptshdr、および rs_systext システムテーブル内の例外ログで、トランザクションの日付によって指定された範囲のトランザクションを削除します。

構文

```
rs_delexception_date transaction_date_start [,transaction_date_end]
```

パラメータ

- **transaction_date_start –** 削除するトランザクションの範囲の最初の日付。日付は二重引用符で囲む。
- **transaction_date_end –** 削除するトランザクションの範囲の最後の日付。範囲の最後の日付 (トランザクション開始日付) の指定は省略可能。日付は二重引用符で囲む。

例

- **例 1 –** 開始日が 2010 年 10 月 1 日のトランザクションを例外ログから削除します。

  ```
  rs_delexception_date "10/01/2010"
  ```

- **例 2 –** 開始日が 2010 年 10 月 1 日から 2010 年 10 月 31 日までの範囲にあるトランザクションを例外ログから削除します。

  ```
  rs_delexception_date "10/01/2010", "10/31/2010"
  ```

使用法

- *transaction_date_end* と *transaction_date_end* の日付には、RSSD のホスト Adaptive Server または ERSSD として機能する SQL Anywhere データベースがサポートしている形式と異なる形式を入力できます。使用できる日付と時刻の形式については、以下を参照してください。
  - 『Adaptive Server Enterprise リファレンスマニュアル：ビルディングブロック』の「システムデータ型とユーザ定義データ型」の「日付と時刻のデータ型」の「日付および時刻データの入力」
  - 『SQL Anywhere サーバ - SQL リファレンス』の「SQL データ型」の「日付と時刻データ型」の「日付と時刻をデータベースに送信する」

- **rs_delexception_date** は、*transaction_date_start* から *transaction_date_end* までの範囲のトランザクション (*transaction_date_start* と *transaction_date_end* を含む) を例外テーブルから削除します。
- パラメータを指定しないと、**rs_delexception_date** はエラーメッセージを表示します。 **rs_helpexception** または **rs_delexception** をパラメータなしで実行した場合は、"org date" カラムを参照して、例外ログにおける現在の有効なトランザクションと開始日を取得します。
- *transaction_date_start* でのみ有効な日付を指定し、2 つ目の有効な日付を *transaction_date_end* で指定しないと、**rs_delexception_date** は *transaction_date_start* で指定したトランザクションのみを削除します。
- 入力したコマンドによってトランザクションが削除されない場合、**rs_delexception_date** はエラーメッセージを表示します。

参照：

- rs_delexception (684 ページ)

rs_delexception_id

rs_exceptscmd、rs_exceptshdr、および rs_systext システムテーブル内の例外ログで、トランザクション ID によって指定された範囲のトランザクションを削除します。

構文

```
rs_delexception_id transaction_id_start [,transaction_id_end]
```

パラメータ

- **transaction_id_start –** 削除するトランザクションの範囲の最初の ID 番号。
- **transaction_id_end –** 削除するトランザクションの範囲の最後の ID 番号。範囲の最後のトランザクションの指定は省略可能。

例

- **例 1 –** ID 番号が 1234 のトランザクションを例外ログから削除します。トランザクションを 1 つ削除する場合は、**rs_delexception** も使用できます。

  ```
  rs_delexception_id 1234
  ```

- **例 2 –** ID 番号 1234 ～ 9800 のトランザクションをすべて例外ログから削除します。

  ```
  rs_delexception_id 1234, 9800
  ```

使用法

- **rs_delexception_id** は、*transaction_id_start* から *transaction_id_end* までの範囲のトランザクション (*transaction_id_start* と *transaction_id_end* を含む) を例外テーブルから削除します。
- パラメータを指定しないと、**rs_delexception_id** はエラーメッセージを表示します。 現在、例外ログ内にある有効なトランザクションのリストを取得するには、**rs_helpexception** または **rs_delexception** をパラメータなしで実行します。
- トランザクション ID の有効な値を 1 つだけ *transaction_id_start* で指定して、2 つ目のトランザクション ID 番号を *transaction_id_end* で指定しないと、**rs_delexception_id** は *transaction_id_start* で指定したトランザクションのみを削除します。
- トランザクション ID 番号として 0 (ゼロ) を入力して、2 つ目のトランザクション ID 番号を入力しないと、**rs_delexception_id** は例外ログ内のすべてのトランザクションを削除します。
- 123.456 のような浮動小数点の数値を入力し、以下を使用する場合：
  - **ERSSD – rs_delexception_id** は整数 123 のみを処理し、小数点以下の数値を無視します。
  - **RSSD – rs_delexception_id** がエラーメッセージと一緒に返され、コマンドを再入力できます。
- 入力したコマンドによってトランザクションが削除されない場合、**rs_delexception_id** はエラーメッセージを表示します。

参照：

- rs_delexception (684 ページ)

rs_delexception_range

システムテーブル `rs_exceptscmd`、`rs_exceptshdr`、および `rs_systext` にある例外ログ内の送信元サイトかユーザ、または送信先サイトによって指定された範囲のトランザクションを削除します。

構文

```
rs_delexception_range
{{"origin"|"org"}, "origin_data_server.origin_database" |
, {"destination"|"dest"},
"destination_data_server.destination_database" |
, "user", "origin_user"}
```

パラメータ

- **"origin"/"org", "*origin_data_server.origin_database*" – "origin"** または短縮形で **"org"** と入力して、例外ログから削除するトランザクションを開始したデータサーバとデータベースを指定します。パラメータは二重引用符で囲み、カンマでパラメータを区切ります。
- **"destination"/"dest", "*destination_data_server.destination_database*" – destination** または短縮形で **"dest"** と入力して、例外ログから削除するトランザクションを受け取ったデータサーバとデータベースを指定します。パラメータは二重引用符で囲み、カンマでパラメータを区切ります。
- **"user", "*origin_user*" – "user"** と入力して、例外ログから削除するトランザクションを開始したユーザを指定します。パラメータは二重引用符で囲み、カンマでパラメータを区切ります。

例

- **例 1 –** SYDNEY_DS データサーバの `south_db` データベースから開始したトランザクションを例外ログから削除します。

```
rs_delexception_range "org", "SYDNEY_DS.south_db"
```

- **例 2 –** TOKYO_DS データサーバの `east_db` データベースが受け取ったトランザクションを例外ログから削除します。

```
rs_delexception_range "destination", "TOKYO_DS.east_db"
```

- **例 3 –** rsuser1 というユーザが開始したトランザクションを例外ログから削除します。

```
rs_delexception_range "user", "rsuser1"
```

使用法

- 一度に入力できるパラメータとその値は 1 つだけです。たとえば、"**org**"、"*origin_dataserver.origin_database*" の次に "**user**"、"*origin_user*" と入力することはできません。
- パラメータを入力して値を指定する必要があります。 パラメータを指定しないと、**rs_delexception_range** はエラーメッセージを表示します。 `Origin Site`、`Dest. Site`、および `Dest` を参照してください。 **rs_helpexception** または **rs_delexception** をパラメータなしで実行した場合は、`User` の各カラムを参照して、例外ログ内の有効なトランザクションに対して各カラムの現在の値のリストを取得します。
- **rs_delexception_range** と一緒に **"origin"**、**"destination"**、または**"user"** のみを入力し、対応する値を指定しなければ、**rs_delexception_range** はエラーメッセージを表示します。

- 入力したコマンドによってトランザクションが削除されない場合、**rs_delexception_range** はエラーメッセージを表示します。

参照：

- rs_delexception (684 ページ)

rs_dump_stats

admin stats によって RSSD に収集された Replication Server の統計をカンマ区切りフォーマットで抽出します。

構文

```
rs_dump_stats ['comment']
```

パラメータ

- **comment –** 表示される統計に関するオプションの説明です。出力ファイルの最初の行に表示されます。

例

- **例 1 –** コメント “Stats from 01/31/2006” を使用して Replication Server の統計を抽出します。

  ```
  rs_dump_stats 'Stats from 01/31/2006'
  ```

  カウンタデータのカラムは次の順序です。

  - 監視期間のタイムスタンプ
  - 監視期間中のカウンタからなる監視の数
  - 監視された値の合計
  - 最後に監視された値
  - 監視された最大値

  カウンタカテゴリ (カウンタカテゴリの詳細については、『Replication Server 管理ガイド 第 2 巻』の「パフォーマンスチューニング」の「カウンタを使ったパフォーマンスのモニタリング」を参照) に応じて、監視の数と監視の合計数、および最後に監視された値と監視された最大値の間に密接な相関関係がある場合があります。たとえば、observer カウンタは、メッセージがキューから読み取られる回数など、イベントの監視数をカウントするだけです。observer カウンタでは、監視数と監視された値の合計が同じになります。同様に、最後に監視された値と監視された最大値の両方が 1 になります (監視期間に読み取られたメッセージがない場合、両方の値は 0 になります)。

注意： 出力の右側にあるコメントは、例を説明するために含まれています。これらは、**rs_dump_stats** の出力の一部ではありません。

```
Comment: Stats from 01/31/2006    == Provided label
Oct 17 2005  3:13:47:716PM    == End of the first observation
period
Oct 17 2005  3:14:24:730PM    == End of the last observation period
2    == Number of observation periods
0    == Number of minutes in each obs period.
0 if less than one.(Calculated as the number of minutes between
the first
and last obs period, divided by the number of observations.)
16384== Number of bytes in an SQM Block to
aid calculations
64== Number of blocks in an SQM Segment
to aid calculations
CM== Module Name. See rs_help_counter
for a complete list.
13== Instance ID. See admin stats for an
explanation.
-1== Inst Val/Mod Type. Further instance
qualification when needed.
dCM== Instance description.
CM: Outbound database connection
requests== Counter description.
CMOBDBReq== Counter display name.
13003           , , 13,  -1== Counter ID and instance qualifying
information.
Oct 17 2005  3:13:47:716PM,  52,
52,  1,  1== Counter data. One row output for
each observation period. See below for
explanation.
Oct 17 2005  3:14:24:730PM,  42,
42,  1,  1
ENDOFDATA== End of output for the previous
counter
CM: Outbound non-database
connection requests== Start of output for the next counter
CMOBNonDBReq
13004           , , 13,  -1
Oct 17 2005  3:13:47:716PM,  2,  2,  1,  1
Oct 17 2005  3:14:24:730PM,  2,  2,  1,  1
ENDOFDATA
.
.
.
CM: Time spent closing an ob fadeout conn
CMOBConnFadeOutClose
13019           , , 13,  -1
Oct 17 2005  3:13:47:716PM,  0,  0,  0,  0
Oct 17 2005  3:14:24:730PM,  2,  6,  2,  4
ENDOFDATA
DIST== Start of output for the next
module/instance
102
```

```
-1
DIST, 102 pds03.tpcc
DIST: Commands read from inbound queue
CmdsRead
30000           , , 102,  -1
Oct 17 2005  3:13:47:716PM,  1,  1,  1,  1
Oct 17 2005  3:14:24:730PM,  1,  1,  1,  1
ENDOFDATA
.
.
.
DSIEXEC: Number of 'message' results
DSIEResMsg
57127           , , 103,  7
Oct 17 2005  3:13:47:716PM,  1,  1,  1,  1
Oct 17 2005  3:14:24:730PM,  1,  1,  1,  1
ENDOFDATA
(return status = 0)== End of output
```

使用法

- **rs_dump_stats** の出力をテキストファイルに取り込み、スプレッドシートや他の分析ツールで分析できます。
- **rs_dump_stats** の出力が含まれているテキストファイルが大きすぎて分析ツールに読み込めない場合は、ファイルを複数のファイルに分割できます。
  - 新しい各ファイルには、元のファイルの最初の 7 つのローと最後のローが含まれている必要があります。
  - 新しい各ファイルの最初の 7 つのローと最後のローの間に、特定のモジュールインスタンスに関連付けられたすべてのローを挿入します。

  通常、分析ツールに応じて、同じファイルに 1 つのモジュールのすべてのインスタンスを含める必要はありません。
- **rs_dump_stats** は RSSD に保存されている統計を削除または変更しません。
- **rs_dump_stats** は監視結果がないカウンタをリストしますが、それらのカウンタデータローは表示しません。**rs_dump_stats** は、サンプリング期間内で少なくとも 1 つの監視結果があるすべてのカウンタのカウンタデータローを表示します。

参照：

- rs_helpcounter (698 ページ)
- admin stats (89 ページ)

rs_fillcaptable

既存の複写定義に対するトランザクションの見積もり率を、*rs_captable* テーブルに記録します。

構文

```
rs_fillcaptable RepDefName, InChRateI, InChRateD, InChRateU,
OutChRateI, OutChRateD, OutChRateU, InTranRate, OutTranRate, DelFlag
```

パラメータ

- **RepDefName –** 複写定義の名前です。
- **InChRateI –** 複写されないものも含めた、秒ごとの挿入数です。デフォルトは秒ごとに 15 回挿入します。
- **InChRateD –** 複写されないものも含めた、秒ごとの削除数です。デフォルトは秒ごとに 15 回削除します。
- **InChRateU –** 複写されないものも含めた、秒ごとの更新数です。デフォルトは秒ごとに 15 回更新します。
- **OutChRateI –** 複写されない挿入は含まない、秒ごとの挿入数です。デフォルトは秒ごとに 15 回挿入します。
- **OutChRateD –** 複写されない削除は含まない、秒ごとの削除数です。デフォルトは秒ごとに 15 回削除します。
- **OutChRateU –** 複写されない更新は含まない、秒ごとの更新数です。デフォルトは秒ごとに 15 回更新します。
- **InTranRate –** データベースの秒ごとのトランザクション数です。デフォルトは秒ごとに 5 トランザクションです。
- **OutTranRate –** データベースの秒ごとの複写トランザクション数です。デフォルトは秒ごとに 5 トランザクションです。
- **DelFlag –** 指定した複写定義のローを更新する場合は、“n” または “N” を指定します。指定した複写定義のローを *rs_captabley* から削除する場合は、“y” または “Y” を指定します。*DelFlag* に “Y”、*RepDefName* に “ALL” を指定すると、*rs_captable* テーブルの内容をすべて削除できます。

例

- **例 1 –** この例では、プライマリデータベースでの全体のトランザクション率は、秒ごとに 10 トランザクションです。10 トランザクションのうち 8 トランザクションは複写されます。したがって、このデータベースの *InTranRate* は 10、*OutTranRate* は 8 です。

T1 と T2 という 2 つの複写トランザクションがあります。T1 は秒ごとに 5 回実行され、*table1* を 2 回更新し、*table2* へ 1 回挿入します。T2 は秒ごとに 3 回実行され、*table1* へ 2 回挿入し、*table2* へ 1 回挿入します。

レプリケートデータベースにはサブスクリプションが 2 つあり、それぞれが複写データの半分を受け取ります。トランザクションは、2 つのサブスクリプションに均等に分配されます。したがって、アウトバウンドの見積もりはインバウンドの見積もりの 50 パーセントになります。

次の表は、この例の内容をまとめたものです。

		table1			table2		
		挿入	更新	削除	挿入	更新	削除
インバウンド	T1 (5 / 秒)		10		5		
	T2 (3 / 秒)	6			3		
	合計	6	10		8		
アウトバウンド	50% 複写	3	5		4		

この例に対してステーブルキューの必要サイズを見積もるには、まず *rs_captable* テーブルの内容をクリアします。次に前述したパラメータを指定して **rs_fillcaptable** を実行します。終了したら、*rs_captable* テーブルの新しい内容を使用して **rs_capacity** ストアドプロシージャを実行します。

- **例 2 –** この例は、*rs_captable* テーブルをクリアします。

```
rs_fillcaptable @RepDefName = 'ALL', @DelFlag = 'Y'
```

- **例 3 –** この例は、1 つ目の複写定義に対する値を *rs_captable* テーブルに記録します。

```
rs_fillcaptable
 repdef1, /* replication definition for table1 */
 6,/* InChRateI */
 0,/* InChRateD */
 10, /* InChRateU */
 3,/* OutChRateI */
 0,/* OutChRateD */
 5,/* OutChRateU */
 10, /* InTranRate */
 8,/* OutTranRate */
 n /* DelFlag */
```

- **例 4 –** この例は、2 つ目の複写定義に対する値を *rs_captable* テーブルに記録します。

```
rs_fillcaptable
 repdef2, /* replication definition for table2 */
 8,/* InChRateI */
```

```
0,/* InChRateD */
0,/* InChRateU */
4,/* OutChRateI */
0,/* OutChRateD */
0,/* OutChRateU */
10, /* InTranRate */
8,/* OutTranRate */
n /* DelFlag */
```

ここに挙げた例からの出力情報を使用してステーブルキューの必要サイズを見積もる方法については、「**rs_capacity**」を参照してください。

使用法

- **rs_fillcaptable** は、ステーブルキューの見積もりに含める各複写定義のトランザクション内容を記録するために使用します。
- **rs_fillcaptable** は、*rs_captable* という名前のワークテーブルを管理します。このテーブルには、データベース内の各複写定義に対する変更率の見積もりが格納されます。
- **rs_fillcaptable** の出力は **rs_capacity** ストアドプロシージャの入力値として使用します。

参照：

- rs_capacity (683 ページ)

rs_helpcheckrepdef

プライマリーキーのカラム、引用符付きのテーブル名またはカラム名、カスタマイズされたファンクション文字列を定義するためにのみ存在する複写定義を表示します。

構文

```
rs_helpcheckrepdef [replication_definition]
```

パラメータ

- **replication_definition –** 入力したテキストから始まる名前で複写定義を指定します。

例

- **例 1 –** プライマリ Replication Server に 2 つの複写定義が定義されているとします。

- **authors** - プライマリキー情報のみを指定します。

```
create replication definition authors
        with primary at NY_DS.pdb1
        (au_id varchar(11),
        au_lname varchar(40) ,
        au_fname varchar(20) ,
        phone char(12),
        address varchar(40),
        city varchar(20),
        state char(2),
        zip char(5),
        contract bit)
        primary key (au_id)
```

- **titleauthor** - プライマリキーに加えて、異なるターゲットカラム名を指定します。

```
create replication definition titleauthor
        with primary at NY_DS.pdb1
        (au_id varchar(11) as author,
        title_id varchar(6) as title,
        au_ord tinyint,
        royaltyper int)
        primary key (au_id, title_id)
```

プライマリ Replication Server の RSSD または ERSSD で **rs_helpcheckrepdef** と入力した場合は、次のように出力されます。

```
Replication Definition Name
---------------------------
authors


(1 row affected)
(return status = 0)
```

使用法

- プライマリ Replication Server の RSSD または ERSSD で **rs_helpcheckrepdef** を実行します。
- *replication_definition* にテキストを入力しない場合は、**rs_helpcheckrepdef** を使用すると、プライマリキーを定義するためにのみ存在する複写定義すべてと、引用符付きのテーブル名またはカラム名がリスト表示されます。
- *replication_definition* にテキストを入力した場合は、**rs_helpcheckrepdef** を使用すると、*replication_definition* に入力したテキストで始まる名前で、プライマリキーと引用符付きのテーブル名またはカラム名を定義するためにのみ存在する複写定義がすべて表示されます。
- RepAgent がプライマリキーと引用符付き識別子の情報を送信し始めたら、**rs_helpcheckrepdef** で指定した複写定義を削除できます。

rs_helpclass

エラークラス、ファンクション文字列クラス、プライマリ Replication Server を表示し、継承クラスの場合は親クラスも表示します。

構文

```
rs_helpclass [class_name]
```

パラメータ

- **class_name –** エラークラスまたはファンクション文字列のクラス名に対応した文字列です。文字列は、名前の全体または最初の部分と一致させてください。

例

- **例 1 –** Replication Server にあるすべてのエラークラスとファンクション文字列クラスについての情報を表示します。

```
rs_helpclass
```

```
Function String Class(es)PRS for CLASSParent Class
------------------------------ ------------------
-------------------------
rs_default_function_classNot Yet Defined.Base class
rs_sqlserver_function_classNot Yet Defined.Base class
sqlserver2_function_classTOKYO_RSrs_default_function_class

Error Class(es)PRS for CLASS
-----------------------------
-----------------------------------------
rs_db2_error_classNot Yet Defined.
rs_msss_error_classNot Yet Defined.
rs_oracle_error_classNot Yet Defined.
rs_sqlserver_error_classNot Yet Defined.
rs_udb_error_classNot Yet Defined

RepServer Error Class(es)PRS for CLASS
------------------------------
----------------------------------------
rs_repserver_error_classNot Yet Defined.
```

- **例 2 –** *sqlserver2_function_class* ファンクション文字列クラスの情報を表示します。

```
rs_helpclass sqlserver2_function_class
```

使用法

注意： エラークラスとファンクション文字列クラスのより詳細な情報を取得するには、**admin show_function_classes** コマンドを使用してください。

- パラメータを指定しないと、**rs_helpclass** は定義されているすべてのエラークラスとファンクション文字列クラスをリストします。
- *class_name* を指定すると、**rs_helpclass** は *class_name* の文字列と一致するエラークラスとファンクション文字列クラスをリストします。
- Replication Server で定義されていないクラスの場合 (Adaptive Server のデフォルトクラスがこれに該当)、**rs_helpclass** はそのクラスを未定義として表示し、その定義方法を示します。

rs_helpclassfstring

ファンクション文字列クラススコープがあるファンクション文字列について、ファンクション文字列情報を表示します。

構文

```
rs_helpclassfstring class_name
[, function_name]
```

パラメータ

- **class_name –** ファンクション文字列を表示するファンクション文字列クラスです。
- **function_name –** ファンクション名に対応する文字列です。文字列は、ファンクション名の全体または最初の部分と一致させてください。

例

- **例 1 –** ファンクション文字列クラス *rs_sqlserver_function_class* のすべてのファンクションのパラメータとファンクション文字列テキストを表示します。

  ```
  rs_helpclassfstring rs_sqlserver_function_class
  ```

- **例 2 –** *rs_sqlserver_function_class*. の **rs_usedb** ファンクションについて、ファンクション文字列テキストを表示します。

  ```
  rs_helpclassfstring rs_sqlserver_function_class, rs_usedb
  ```

  ```
  Function NameFString Name FSClass Name
  -------------- ------------ --------------------------
  rs_usedb rs_usedb rs_sqlserver_function_class

  FString Text
  ```

```
------------------------------------------------------------
use ?rs_destination_db!sys_raw?
```

使用法

- *function_name* を指定しないと、**rs_helpclassfstring** は、ファンクション文字列クラスのすべてのファンクションに対して定義されたすべてのファンクション文字列を表示します。
- *function_name* を指定すると、**rs_helpclassfstring** は *function_name* と一致するファンクション文字列を表示します。たとえば、**rs_insert**、**rs_delete**、**rs_update**、**rs_select**、またはユーザ定義のファンクションなどです。
- 派生ファンクション文字列クラスの場合、カスタマイズされていない継承ファンクション文字列は表示されません。

rs_helpcounter

カウンタの情報を表示します。

構文

```
rs_helpcounter [{sysmon | duration | observer | monitor
    | must_sample | no_reset | keep_old}
    | module_name [, {short | long}] | keyword [, {short | long}]]
```

パラメータ

- **sysmon –** これらのカウンタを指定して、パフォーマンスの評価および複写システムのプロファイル情報の収集における効率性を最大限に高めます。
- **duration –** 100 分の 1 秒単位で表される時間間隔で実行時間を測定するすべてのカウンタを指定します。
- **observer –** イベントの発生回数を記録するカウンタを指定します。たとえば、キューからメッセージが読み取られる回数を記録します。
- **monitor –** 現在の値を記録するカウンタを指定します。たとえば、キューから最後に読み取られたメッセージのサイズをバイト数で記録します。
- **must_sample –** サンプリングがオンに設定されているかどうかにかかわらず、サンプリングを保持する必要があるカウンタを指定します。
- **no_reset – admin stats, reset** の実行時に値がリセットされないカウンタを指定します。
- **keep_old –** 現在の値と前回の値を保持するカウンタを指定します。
- **module_name –** モジュールの名前です。*dsi*、*dsiexec*、*sqt*、*cm*、*dist*、*rsi*、*sqm*、*repagent* などがあります。

- **short –** 指定したカウンタの表示名、モジュール名、カウンタ説明を出力するように、Replication Server に指示します。
- **long –** *rs_statcounters* テーブルのすべてのカラムの値を出力するように、Replication Server に指示します。
- **keyword –** 検索キーワードです。カウンタの長い名前、カウンタの表示名、カウンタの説明を検索します。

例

- **例 1 –** すべてのモジュール名と、**rs_helpcounter** を使用して詳細を表示するための構文を表示します。

```
1> rs_helpcounter
2> go

ModuleName
------------------------------
CM
DIST
DSI
DSIEXEC
REPAGENT
RSH
RSI
RSIUSER
SERV
SQM
SQMR
SQT
STS
SYNC
SYNCELE
(12 rows affected)
```

```
How to Use rs_helpcounter
-----------------------------------------------------------------
rs_helpcounter  -> Shows module names and help.
rs_helpcounter [ sysmon | duration | observe | monitor
| must_sample | no_reset | keep_old ]
rs_helpcounter ModuleName[, {short | long }]
rs_helpcounter keyword[, { short | long }]
where “keyword” is part of the counter name, display name or
description
(return status = 0)
```

- **例 2 –** SQM リーダの表示名、モジュール名、カウンタの説明をリストします。

```
rs_helpcounter sqmr, short
```

```
Display NameModule NameCounter Description
-----------------------------------------------------------------
---
BlocksReadSQMRNumber of 16K blocks read from a stable
queue by an SQM Reader thread.
```

```
ClocksReadCachedSQMRNumber of 16K blocks from cache read by
an SQM Reader thread.
CmdsReadSQMRCommands read from a stable queue by an
SQM Reader thread.
SQMRReadTimeSQMRThe amount of time taken for SQMR to read
a block.
SleepsStartQRSQMRsrv_sleep() calls by an SQM Reader
client due to waiting for SQM thread to
start.
SleepsWriteQSQMRsrv_sleep() calls by an SQM read client
due to waiting for the SQM thread to
write.
XNLInterruptedSQMRNumber of interruptions so far when
reading large messages with partial
read. Such interruptions happen due to
time out, unexpected wakeup, or nonblock
read request, which is marked as
READ_POSTED.
XMLPartialsSQMRPartial large messages read so far.
XNLReadsSQMRLarge messages read successfully so
far. This does not count partial
messages, or timeout interruptions.
(return status = 0)
```

使用法

- **rs_helpcounter** を使用して、*rs_statcounters* システムテーブルの内容を検索できます。
- パラメータを指定しないで **rs_helpcounter** を使用すると、モジュール名の一覧と構文が出力されます。
- カウンタステータスと RSSD に格納されているその他のカウンタ情報の詳細については、rs_statcounters システムテーブルを参照してください。

パーミッション

このコマンドは、すべてのユーザが実行できます。

rs_helpdb

Replication Server が認識しているデータベースについての情報を提供します。

構文

```
rs_helpdb [data_server, database]
```

パラメータ

- **data_server –** 情報を表示するデータベースのあるデータサーバです。

- **database –** 情報を表示するデータベースの名前です。

例

- **例 1 –**

```
rs_helpdb
```

```
dsnamedbnameconn_id   dbid
-------------------------------------     -------   ----
TOKYO_DSTOKYO_RSSD101       101
SYDNEY_DSSYDNEY_RSSD102       102
TOKYO_DSpubs2105       105
TOKYO_DS                        pubs2_conn2          106       105

controlling_prserrorclass
-----------------------------------------------------
TOKYO_RSrs_sqlserver_error_class
SYDNEY_RSrs_sqlserver_error_class
TOKYO_RSrs_sqlserver_error_class
TOKYO_RSrs_sqlserver_error_class

repserver_errorclassfuncclass
-----------------------------------------------------------
rs_repserver_error_classrs_sqlserver_function_class
rs_repserver_error_classrs_sqlserver_function_class
rs_repserver_error_classrs_sqlserver_function_class
rs_repserver_error_classrs_sqlserver_function_class

status
-------------------------------------------------------------------------
--
Log Transfer is ON, Distribution is ON
Log Transfer is ON, Distribution is ON
Log Transfer is ON, Distribution is ON
Log Transfer is ON, Distribution is ON
```

使用法

- *data_server* と *database* パラメータを指定しないと、**rs_helpdb** は *rs_databases* システムテーブルにあるすべてのデータベースの情報を返します。
- **rs_helpdb** は、Replication Server の RSSD で実行されます。
- 各データベースに対し、**rs_helpdb** は次の情報を表示します。

  dsname - データベースのあるデータサーバの名前です。

  dbname - データベースの名前です。

  connid - Multi-Path Replication を有効にした場合、複写システム全体でデータベースをユニークに識別するために割り当てられた ID 番号です。

  dbid - 複写システム全体でデータベースをユニークに識別するために割り当てられた ID 番号です。

  controlling_prs - データベースを管理している Replication Server です。

errorclass - このデータベースのデータサーバから返されるエラーを処理するために、Replication Server が使用するエラークラスです。
repserver_errorclass - このデータベースの Replication Server から返されるエラーを処理するエラークラスです。
funcclass - データベースで使用されるファンクション文字列クラスです。
status - データベースに対してログ転送と分配がオンかオフかを示します。
ltype - データベースコネクションのタイプです (論理コネクションまたは物理コネクション)。
ptype - データベースのタイプです (アクティブデータベース、スタンバイデータベース、または論理コネクション)。

rs_helpdbrep

現在の Replication Server に関連するデータベース複写定義についての情報を表示します。

構文

```
rs_helpdbrep [db_repdef[, data_server[, database]]]
```

パラメータ

- **db_repdef** – データベース複写定義の名前を指定します。
- **data_server** – データベース複写定義を表示するデータサーバの名前を指定します。
- **database** – データベース複写定義を表示するデータベースの名前を指定します。

例

- **例 1** – 次の例では、Adaptive Server は現在の Replication Server で見つかったすべてのデータベース複写定義の情報を表示します。

```
rs_helpdbrep
```

```
DB Rep.Def.NamePrimary DS.DBPrimary RSRep.DDLRep.Sys. Rep.Tab
Rep.Func.
------------------------------------------------------------------
---
db_rep1PDS.pdb1PRSYesOut-ListAllAll
db_rep2PDS.pdb2PRSYesOut-ListAllAll

Rep.Tran. Rep.Upd. Rep.Del. Rep.Ins. Rep.Sel. Creation Date
--------- -------- -------- -------- -------- -------------------
AllAllAllAllAllNov 26 2008 6:58AM
AllAllAllAllAllDec 2 2008 6:12PM
```

- **例 2 –** 次の例では、Adaptive Server は 1 つのデータベース複写定義 *db_rep1* に関する情報を表示します。

```
rs_helpdbrep db_rep1
```

```
DB Rep.Def.Name Primary DS.DB Primary RS  Rep.DDL  Rep.Sys.
Rep.Tab Rep.Func.
--------------- ------------  ---------  ------- -------  -------
--------
db_rep1         PDS.pdb1      PRS        Yes     Out-List  All    All

Rep.Tran. Rep.Upd. Rep.Del. Rep.Ins. Rep.Sel. Creation Date
--------- -------- -------- -------- -------- -------------------
AllAllAllAllAllNov 26 2008 6:58AM


Rep.Type      Owner     Name
------------  --------  -------------
Not Rep.Sys.  . sp_setrepproc

DBRep.Def.Name DBSub.Name ReplicationDS.DB ReplicateRS Creation
Date
-------------- ---------- ---------------- -----------
------------------
db_rep1db_sub1RDS1.rdb1RRS1Nov 26 2008 6:58AM
db_rep1db_sub2RDS2.rdb2RRS2Nov 26 2008 6:59AM
```

使用法

- Adaptive Server は、指定されたデータベース複写定義に関する詳細情報のみ表示します。
- パラメータにはワイルドカード '%' を含めることができます。このワイルドカードは任意の文字列を表します。たとえば、文字列 'abc%' が *db_repdef* に割り当てられている場合、**rs_helpdbrep** はデータベース複写定義名の先頭に 'abc' が付いているすべてのデータベース複写定義をリストします。

参照：

- rs_helpdbsub (703 ページ)

rs_helpdbsub

レプリケートデータサーバに関連するデータベースサブスクリプションについての情報を表示します。

構文

```
rs_helpdbsub [db_sub[, data_server[, database]]]
```

パラメータ

- **db_sub** – データベースサブスクリプションを指定します。
- **data_server** – データベースサブスクリプションを表示するデータサーバの名前を指定します。
- **database** – データベースサブスクリプションを表示するデータベースの名前を指定します。

例

- **例 1** – 次の例では、Adaptive Server は 1 つのデータベースサブスクリプション *db_sub1* に関する情報を表示します。

```
rs_helpdbsub db_sub1, RDS1, rdb1
```

```
DBSub.NameReplicateDS.DBReplicateRSStatus at RRSDBRep.Def.Name
---------------------------------------------------------------
db_sub1RDS1.rdb1RRS1Validatedb_rep

PrimaryDS.DBPrimaryRSMethodTrunc.Table Creation Date
---------------------------------------- -----------------
PDS.pdb1PRSBulk CreateYesMay 2 2003 3:38PM
```

使用法

- パラメータを何も指定しないと、**rs_helpdbsub** は Replication Server に定義されているデータベースサブスクリプションをリストします。
- *db_sub* パラメータのみを指定すると、**rs_helpdbsub** はデータベースサブスクリプション名が *db_sub* と一致する Replication Server に定義されているすべてのデータベースサブスクリプションを表示します。
- パラメータにはワイルドカード '%' を含めることができます。このワイルドカードは任意の文字列を表します。たとえば、文字列 'abc%' が *db_sub* に割り当てられている場合、**rs_helpdbsub** はデータベースサブスクリプション名の先頭に 'abc' が付いているすべてのデータベースサブスクリプションをリストします。

参照：

- rs_helpdbrep (702 ページ)

rs_helperror

指定したデータサーバまたは Replication Server のエラー番号に割り当てられている、Replication Server のエラーアクションを表示します。

構文

```
rs_helperror server_error_number [, v]
```

パラメータ

- **server_error_number –** データサーバのエラー番号です。
- **v –** このオプションを指定すると、Adaptive Server のエラーメッセージがある場合、その内容が表示されます。

例

- **例 1 –**

```
rs_helperror 2601, v
```

```
DS Error NumError ActionError Class
-----------------------------------
----------------------------
2601Stop Replicationrs_sqlserver_error_class

Adaptive Server Error Message
------------------------------------------------------------------------
-----
Attempt to insert duplicate key row in object '%.*s' with unique
index
 '%.*s'%S_EED

RS Error NumError ActionReplication Server Error Class
------------------------------------------------------
```

使用法

- すべてのエラークラスについて、割り当てられているエラーアクションが表示されます。
- データサーバのエラー番号にエラーアクションを割り当てるには、**assign action** コマンドを使用します。

参照：

- assign action (226 ページ)

rs_helpexception

例外ログにあるトランザクションを表示します。

構文

```
rs_helpexception [transaction_id, [, v]]
```

パラメータ

- **transaction_id** – 表示するトランザクションの番号です。
- **v** – トランザクションのテキストを詳細に表示します。

例

- **例 1** – 例外ログにあるすべてのトランザクションの情報を表示します。

  ```
  rs_helpexception
  ```

- **例 2** – トランザクション番号 1234 の詳細情報と、そのトランザクションのテキストを表示します。

  ```
  rs_helpexception 1234, v
  ```

使用法

- パラメータを何も指定しないと、**rs_helpexception** は例外ログにあるトランザクションの情報を、すべてのトランザクションの番号も含めて表示します。
- 有効な *transaction_id* を指定すると、**rs_helpexception** はトランザクションの詳細情報を表示します。
- 例外ログにあるトランザクションを削除するには、**rs_delexception** を使用します。

参照：

- rs_delexception (684 ページ)

rs_helpfstring

複写定義に関連したファンクションのパラメータとファンクション文字列テキストを表示します。

構文

```
rs_helpfstring replication_definition
[, function_name]
```

パラメータ

- **replication_definition –** 表示するファンクションを持つテーブルまたはファンクション複写定義です。
- **function_name –** ファンクション名に対応する文字列です。文字列は、ファンクション名の全体または最初の部分と一致させてください。

例

- **例 1 –** 複写定義 *authors_rep* のすべてのファンクションに対するパラメータとファンクション文字列テキストを表示します。

  ```
  rs_helpfstring authors_rep
  ```

- **例 2 –** 複写定義 *authors_rep* の **rs_insert** ファンクションに対するパラメータとファンクション文字列テキストを表示します。

  ```
  rs_helpfstring authors_rep, rs_insert
  ```

  ```
  Function String information for Replication Definition.
   'authors_rep'
  Valid Parameters are:
   Parameter NameDatatype
   ------------------------------ ------------------------------
   @au_idvarchar
   @au_lname varchar
   @au_fname varchar
   @phonechar
   @addressvarchar
   @city varchar
   @statechar
   @countryvarchar
   @postalcode char

  Rep.Def.NameFunction Name FString NameFSClass Name
  -------------- -------------- ------------
  ---------------------------
   authors_reprs_insert rs_insert rs_sqlserver_function_class

  --- Begin FString Text ---
  ```

```
------------------------------------------------------------
*** System-Supplied Transact-SQL Statement ***
--- End FString Text ---
```

使用法

- *function_name* を指定しないと、**rs_helpfstring** は複写定義のすべてのファンクションに対して定義されたすべてのファンクション文字列を表示します。
- *function_name* を指定すると、**rs_helpfstring** は *function_name* と一致するファンクション文字列を表示します。たとえば、**rs_insert**、**rs_delete**、**rs_update**、**rs_select**、またはユーザ定義のファンクションなどです。
- システムが生成するデフォルトのファンクション文字列のテキストは、RSSD には格納されていません。これらのファンクション文字列に対しては、**rs_helpfstring** は “System-Supplied Transact-SQL Statement” というメッセージを表示します。

rs_helpfunc

Replication Server または特定の複写定義で使用されているファンクションの情報を表示します。

構文

```
rs_helpfunc [replication_definition [, function_name]]
```

パラメータ

- **replication_definition –** ファンクション情報を表示する複写定義です。
- **function_name –** ファンクション名に対応する文字列です。文字列は、ファンクション名の全体または最初の部分と一致させてください。

例

- **例 1 –** 使用しているすべてのファンクション、複写定義、プライマリ Replication Server を表示します。各ファンクションのクラススコープも表示します。

  ```
  rs_helpfunc
  ```

- **例 2 –** 複写定義 *authors_rep* のすべてのファンクションに対し、ファンクション名、パラメータ、データ型などのファンクション情報を表示します。

  ```
  rs_helpfunc authors_rep
  ```

  ```
  Functions and Parameters for Replication Definition:
   'authors_rep'
  System Function Names
  ```

```
 ---------------------
 rs_insert
 rs_delete
 rs_update
 rs_select
 rs_select_with_lock

Parameter(s)DatatypeLength
 --------------- -------- ------
 @statechar2
 @postalcode char 10
 @au_idvarchar11
 @phonechar 12
 @countryvarchar12
 @city varchar20
 @au_fname varchar20
 @addressvarchar40
 @au_lnamevarchar40
```

- **例 3 –** 複写定義 *authors_rep* のファンクション **rs_insert** のパラメータとデータ型を表示します。

```
rs_helpfunc authors_rep, rs_insert
```

使用法

- パラメータを何も指定しないと、**rs_helpfunc** は Replication Server に定義されているすべてのファンクションを表示します。
- *replication_definition* を指定すると、その複写定義に指定されているファンクションだけを表示します。*function_name* も指定すると、**rs_helpfunc** は *function_name* と一致する名前のファンクションを表示します。
- ユーザ定義ファンクションが重複していて、非同期トランザクションに支障をきたす可能性がある場合には、**rs_helpfunc** はそのことを表示します。

rs_helpobjfstring

ターゲットのスコープファンクション文字列のパラメータとファンクション文字列テキストを表示します。

構文

```
rs_helpobjfstring data_server, database, [owner.]object_name[,
function_name]
```

パラメータ

- ***data_server* –** ターゲットスコープファンクション文字列を使用するレプリケートまたはスタンバイデータサーバを指定します。

- ***database*** **–** ターゲットスコープファンクション文字列を使用するレプリケートまたはスタンバイデータベースを指定します。
- **[owner.]object_name –** カスタムファンクション文字列を表示するためのテーブルまたはストアドプロシージャ。テーブルに所有者がいる場合は、所有者を指定します。
- **function_name –** 入力する必要がある完全なファンクション名に対応する文字列です。たとえば、関数名が **rs_writetext** の場合は、"rs_write" を入力しないでください。

例

- **例 1 – upd_datetime** ストアドプロシージャのターゲットスコープファンクション文字列を作成するとします。

```
create function string upd_datetime.upd_datetime
     for database NY_DS.rdb1
     with overwrite
          output language
          'update datetime set
          row_num = ?row_num!param?,
          datecol = ?datecol!param?,
          timecol = ?timecol!param?,
          ndatecol = ?ndatecol!param?,
          ntimecol = ?ntimecol!param?,
          comment = ?comment!param?
          where
          row_num = ?row_num!param?'
```

次のように入力します。

- `rs_helpobjfstring NY_DS,rdb1,upd_datetime`

または

- `rs_helpobjfstring NY_DS,rdb1,upd_datetime,upd_datetime`

次のようなメッセージが表示されます。

```
Function String information for Target Object: 'upd_datetime'.

Object Name     Object Type         Function Name
-----------     -----------         --------------
upd_datetime    stored procedure    upd_datetime

Function String Name   Output Type Option       System Generated
--------------------   -------------------      -----------------
upd_datetime           language not applicable  no


         --- Beginning of Function String Text ---

FString Text
-------------------------------------------------------------------
 update datetime set
```

```
         row_num = ?row_num!param?,
         datecol = ?datecol!param?,
         timecol = ?timecol!param?,
         ndatecol = ?ndatecol!param?,
         ntimecol = ?ntimecol!param?,
         comment = ?comment!param?
         where
         row_num = ?row_num!param?

         --- End of Function String Text ---


(return status = 0)
```

- **例 2 –** は dbo テーブルのターゲットスコープファンクション文字列を作成します。

```
create function string dbo.datetime.rs_insert
     for database NY_DS.rdb1
     with overwrite
     output language
     'insert datetime values (
             ?row_num!new? ,
             ?datecol!new? ,
             ?timecol!new? ,
             ?ndatecol!new? ,
             ?ntimecol!new? ,
             ?comment!new?)
       update fn_monitor set insert_count = insert_count + 1'
```

次のように入力します。

```
rs_helpobjfstring NY_DS,rdb1,'dbo.datetime',rs_insert
```

次のようなメッセージが表示されます。

```
Function String information for Target Object: 'dbo.datetime'.

Object Name       Object Type   Function Name
-------------     -----------   --------------
datetime          table         rs_insert


Function String Name   Output Type Option        System Generated
--------------------   -------------------       ----------------
rs_insert              language not applicable   no



         --- Beginning of Function String Text ---

FString Text
-----------------------------------------------------------------
     insert datetime values (
                          ?row_num!new? ,
                          ?datecol!new? ,
                          ?timecol!new? ,
```

```
                        ?ndatecol!new? ,
                        ?ntimecol!new? ,
                        ?comment!new?)
                        update fn_monitor
                        set insert_count =
                        insert_count + 1

        --- End of Function String Text ---
(return status = 0)
```

この例では、**create function string** コマンドのオブジェクト名にテーブルの所有者 dbo が含まれています。

注意： dbo.datetime には引用符を付ける必要があります。

ファンクション文字列の作成時にテーブルの所有者を省略し、次のように入力します。

```
rs_helpobjfstring NY_DS,rdb1,datetime,rs_insert
```

次のようなメッセージが表示されます。

```
Target Object 'datetime' does not have customized function string.
(return status = -1
```

- **例 3 –** dbo.tbl1 テーブルのターゲットスコープファンクション文字列を作成します。

```
create function string dbo.tbl1.rs_writetext; unitext_fld1 for
NY_DS.rdb1
        output RPC
        'exec update_repl_unitext
                @p_key       = ?p_key!new?,
                @unitext_fld       = ?unitext_fld1!new?,
                @last_chunk         = ?rs_last_text_chunk!sys?'
```

次のように入力します。

```
rs_helpobjfstring NY_DS,rdb1,'dbo.tbl1',rs_writetext
```

次のようなメッセージが表示されます。

```
Function String information for Target Object: 'dbo.tbl1'.

Object Name         Object Type    Function Name
-------------       -----------    --------------
tbl1                table          rs_writetext

Function String Name     Output Type Option    System Generated
----------------------   ------------------    ----------------
unitext_fld1             RPC  not applicable   no

         --- Beginning of Function String Text ---

 FString Text

-------------------------------------------------------------------
        exec update_repl_unitext
```

```
             @p_key = ?p_key!new?,
             @unitext_fld = ?unitext_fld1!new?,
             @last_chunk = ?rs_last_text_chunk!sys?

         --- End of Function String Text ---


(return status = 0)
```

- **例 4 –** dbo.tbl1 テーブルのターゲットスコープファンクション文字列を作成するとします。

```
create function string dbo.tbl1.rs_datarow_for_writetext
for NY_DS.rdb1
        output RPC
        'exec update_txtimg_stat
                @p_key       = ?p_key!new?,
                @txtfld_stat = ?unitext_fld1!text_status?'
```

次のように入力します。

```
rs_helpobjfstring NY_DS,rdb1,'dbo.tbl1',rs_datarow_for_writetext
```

次のようなメッセージが表示されます。

```
Function String information for Target Object: 'dbo.tbl1'.

Object Name          Object Type    Function Name
-------------        -----------    --------------
tbl1                 table          rs_datarow_for_writetext

Function String Name      Output Type Option     System Generated
--------------------      ------------------     ----------------
rs_datarow_for_writetext  RPC not applicable     no


         --- Beginning of Function String Text ---

 FString Text
 ------------------------------------------------------------------------
 exec update_txtimg_stat
                @p_key = ?p_key!new?,
                @txtfld_stat = ?unitext_fld1!text_status?


         --- End of Function String Text ---


(return status = 0)
```

次のように入力します。

```
rs_helpobjfstring NY_DS,rdb1,'dbo.tbl1'
```

両方のファンクション文字列情報が表示されます。

- この例の

```
rs_helpobjfstring
NY_DS,rdb1,'dbo.tbl1',rs_datarow_for_writetext
```

と

- 例 3 の

```
rs_helpobjfstring NY_DS,rdb1,'dbo.tbl1',rs_writetext
```

使用法

- *function_name* を指定しないと、**rs_helpobjfstring** はオブジェクトのすべてのファンクション文字列を表示します。
- *function_name* を指定すると、**rs_helpobjfstring** は *function_name* と一致するファンクション文字列を表示します。たとえば、**rs_insert**、**rs_delete**、**rs_update**、**rs_select**、またはユーザ定義のファンクションなどです。
- システムが生成するデフォルトのファンクション文字列のテキストは、RSSD には格納されていません。これらのファンクション文字列の場合、**rs_helpobjfstring** は "System-Supplied Transact-SQL Statement" を表示します。

rs_helppartition

Replication Server のパーティションに関する情報を表示します。

構文

```
rs_helppartition [partition_name]
```

パラメータ

- ***partition_name*** – パーティション名に対応する文字列です。 文字列は、パーティション名の全体または最初の部分と一致させてください。

例

- **例 1** – Replication Server で使用できるすべてのパーティションについての要約情報を表示するには、次のように入力します。

```
rs_helppartition
```

次のような内容が表示されます。

```
Displaying all partitions known to 'TOKYO_RS'.
Logical Name Size (MB) Segments Allocated (MB)
------------  ---------  -----------------------
partition_1       20  3
auto_winp           200                        3
```

- **例 2 –** partition_1 パーティションについての詳細情報を表示するには、次のように入力します。

```
rs_helppartition partition_1
```

次のような内容が表示されます。

```
Information for stable device: 'partition_1' on 'TOKYO_RS'.
This device is active.

Physical NamePartition ID
---------------------------------------  ------------
/remote/tyrell2/app/dev/tokyo_rs_p1.dat101


Partition Size (MB)Segments Allocated (MB)
-------------------  ----------------------
                 20      5


Inbound Database Queue(s) on this partition:
Connection Name      Number of Segments
-------------------  ------------------
LDS.pubs2     1
TOKYO_RS.TOKYO_RSSD 1


Outbound Database Queue(s) on this partition:
Connection Name Number of Segments
-------------------  ------------------
LDS.pubs2 1
TOKYO_RS.TOKYO_RSSD1


Outbound Replication Server Queue(s) on this partition:
Connection Name Number of Segments
-------------------  ------------------
SYDNEY_RS 1
```

使用法

- パラメータを指定しない場合、**rs_helppartition** はすべての Replication Server パーティションに関する要約情報を表示します。これには、手動で作成したパーティションと、自動サイズ変更可能なパーティションが含まれます。
- *partition_name* を指定すると、**rs_helppartition** は *partition_name* と一致する名前のすべてのパーティションに関する情報を表示します。
- *partition_name* がパーティション名と完全に一致する場合、そのパーティションの詳細情報を表示します。この情報には、論理名、物理名、合計サイズ、各パーティションから割り付けられる 1MB セグメントの数、パーティション上のキューが含まれます。

- *partition_name* が特定のパーティション名と完全に一致しない場合、名前の最初の部分が *partition_name* と一致するすべてのパーティション、または認識されているすべてのパーティションの情報を表示します。

参照：
- admin auto_part_path (53 ページ)
- alter auto partition path (133 ページ)
- create auto partition path (284 ページ)
- drop auto partition path (405 ページ)

rs_helppub

パブリケーションについての情報を表示します。

構文

```
rs_helppub [publication_name, primary_dataserver, primary_db,
    article_name]
```

例

- **例 1 –**

```
rs_helppub
```

```
 Publication NamePRSPrimary DS.DB
 ---------------- --------------------
funcpub prim_rsP_DS.pdb1
pub1 prim_rsP_DS.pdb1
pub2 prim_rsP_DS.pdb1

Num Articles Status Request Date
------------------- --------------------
3ValidMar 23 1998 11:51AM
7ValidMar 24 1998 10:41AM
3ValidMar 24 1998 11:50AM

(return status = 0)
```

- **例 2 –**

```
rs_helppub funcpub:
```

```
Publication NamePRS Primary DS.DB
---------------------- --------------
funcpubprim_rsP_DS.pdb1

Num Articles StatusRequest Date
-------------------------------------
3ValidMar 23 1998 11:51AM
```

```
Article Name Replication Definition Type
----------------------------------------
authorsauthors
authorsauthors
publisherspublishers

Primary Object NameReplicate Object NameRequest Date
----------------- ----------------------------------------
many_rows_datamany_rows_dataMar 23 1998 10:01AM
 Mar 23 1998 11:51AM

Sub NameReplicate DS.DBOwnerReq. Date
---------- ---------------------------------------------
funcsub1R_DS.rdb1saMar 24 1998 11:12AM

(return status = 0)
```

- **例 3 –**

```
rs_helppub funcpub, P_DS, pdb1, publishers:
```

```
Article NamePublication Name Replication Definition
--------------------------- ------------------------
publishers funcpubpublishers

Primary Object NameReplicate Object Name
----------------------------------------
publisherspublishers

TypeRequest DateStatus
----------------- ----------
TableMar 23 1998 11:51AMValid

Where clauses
-----------------------------------------------------------------

where
pub_id = "0736"

Sub. Name Replicate DS.DBOwnerReq Date
----------------------------------------------
funcsub1R_DS.rdb1saMar 24 1998 11:12AM

(return status = 0)
```

使用法

- プライマリサイトで **rs_helppub** を実行すると、そのサイトで作成されたすべてのパブリケーションの情報が表示されます。
- レプリケートサイトで **rs_helppub** を実行すると、そのサイトでサブスクリプションが作成されたパブリケーションについての情報だけが表示されます。

- パブリケーションまたはアーティクルへのサブスクリプションに関する情報を表示するには、**rs_helppubsub** を使用します。
- サブスクリプションステータスの最も正確なレポートを取得するには、**check_subscription** を使用します。

参照：

- rs_helppubsub (718 ページ)

rs_helppubsub

パブリケーションサブスクリプションおよびアーティクルサブスクリプションについての情報を表示します。

構文

```
rs_helppubsub subscription_name, publication_name,
primary_dataserver,
    primary_db, replicate_dataserver, replicate_db
```

例

- **例 1 –** このサイトで認識されているパブリケーションサブスクリプションをすべてリストします。

```
rs_helppubsub
```

```
Subscription NamePublication Name
------------------ ----------------
funcsub1funcpub

Primary DS.DB Replicate DS.DBPRS Status RRS Status
--------------------------------------------------
P_DS.pdb1R_DS.rdb1UnknownValid

Owner Request Date
------ -------------------
sa Mar 24 2007 11:12AM
(1 row affected)

Subscription Name Article NameReplication Definition
----------------------------------------------------
funcsub1authorsauthors

PRS StatusRRS StatusRequest DateAutocorrection
--------------------------------------- --------------
UnknownValid Mar 24 2007 11:11AMoff

Subscribe to Truncate TableDynamic SQL
--------------------------------------
```

```
UnknownOn
(1 row affected, return status = 0)
```

- **例 2 –** *sub* という名前のパブリケーションサブスクリプションをすべてリストします。

```
rs_helppubsub sub
```

- **例 3 –** *pub* という名前のパブリケーションに対する *sub* という名前のパブリケーションサブスクリプションをすべてリストします。

```
rs_helppubsub sub, pub
```

- **例 4 –** 指定したパブリケーションに対する sub という名前のサブスクリプションをすべてリストします。

```
rs_helppubsub sub, pub, primary_dataserver, primary_db
```

- **例 5 –** グループ内のパブリケーションサブスクリプションおよびアーティクルサブスクリプションをリストします。

```
rs_helppubsub sub, pub, primary_dataserver, primary_db,
replicate_dataserver, replicate_db
```

```
Subscription NamePublicationNamePrimaryDS.DB
--------------------------------------------------
subpubost_cardhu_2.pdb1

ReplicateDS.DBPRSStatusRRSStatusOwner
----------------------------------------------
ost_cardhu_2.rdb1UnknownValidrdb1_owner

RequestDateSubscriptionNameArticleName
------------------------------------------
February251998subarticle1
article2
subarticle3
subarticle4
subarticle5

PRSStatusRRSStatusRequestDateReplicationDefinition
------------------------------------------------------
UnknownVALIDFeb25,1998repdef1
repdef2
UnknownVALIDFeb25,1998repdef3
UnknownVALIDFeb25,1998repdef4
UnknownVALIDFeb25,1998repdef5

AutocorrectionSubscribetoTruncateTableDynamic SQL
-----------------------------------------------------
onoffon
offonon
offoffon
offoffon
```

使用法

- あるアーティクルまたはパブリケーションに対するすべてのサブスクリプションを調べるには、**rs_helppub** を使用します。
- サブスクリプションステータスの最も正確なレポートを取得するには、**check_subscription** を使用します。

参照：

- rs_helppub (716 ページ)

rs_helprep

複写定義についての情報を表示します。

構文

```
rs_helprep [replication_definition]
```

パラメータ

- **replication_definition –** 複写定義名に対応する文字列です。文字列は、複写定義名の全体または最初の部分と一致させてください。

例

- **例 1 –** `rs_helprep`

Rep def	PRS	Primary DS.DB	Primary Table	Replicate Table	Type
authors	cardhu_11	cardhu_10.pdb1	authors	ling.authors_r1	Tbl
authors1	cardhu_11	cardhu_10.pdb1	authors	authors_r2	Tbl
discounts	cardhu_11	cardhu_10.pdb1	discounts	discounts	Tbl
publishers	cardhu_11	cardhu_10.pdb1	publishers	ling.publishers_r1	Tbl
publishers1	cardhu_11	cardhu_10.pdb1	publishers	publishers_r2	Tbl
roysched	cardhu_11	cardhu_10.pdb1	roysched	roysched	Tbl
rs_classes	cardhu_11	cardhu_10.emb	rs_classes	Tbl	
rs_columns	cardhu_11	cardhu_10.emb	rs_columns	Tbl	
rs_databases	cardhu_11	cardhu_10.emb	rs_databases	Tbl	

Rep def	PRS	Primary DS.DB	Primary Table	Replicate Table	Type
rs_erroractions	cardhu_11	cardhu_10.emb	rs_erroractions	Tbl	
rs_funcstrings	cardhu_11	cardhu_10.emb	rs_functstrings	Tbl	
rs_functions	cardhu_11	cardhu_10.emb	rs_functions	Tbl	
rs_objects	cardhu_11	cardhu_10.emb	rs_objects	Tbl	
rs_routes	cardhu_11	cardhu_10.emb	rs_routes	Tbl	
rs_systext	cardhu_11	cardhu_10.emb	rs_systext	Tbl	

- **例 2 – create function replication definition** を使用して作成された authors 複写定義に関する情報を表示します。

```
rs_helprep authors
```

```
Replication Definition NamePRSType Creation Date
--------------------------- -------------------- ----
-------------------
authorsprimary_rsTblNov 26, 2008 1:48PM

PDS.DBPrimary OwnerPrimary Table
------------------- -------------------- --------------------
pds.pdbauthors

Replicate OwnerReplicate Table
-------------------- --------------------
authors

Send Min Cols. Used by Standby Min Vers Dynamic SQL SQL Stmt.Rep.
-------------- --------------- -------- ----------- -------------
NoNo1000OnUD

Col. NameRep. Col. NameDatatypeLen.Pri. Col.Searchable
------------------------------------------------------
au_idau_idvarchar1111
au_lnameau_lnamevarchar4001
au_fnameau_fnamevarchar2001
```

- **例 3 – create applied function replication definition** を使用して作成された R1_app 複写定義に関する情報を表示します。

```
rs_helprep R1_app
```

```
Replication Definition Name PRSTypeCreation Date
------------------------------------------------------------------
---
R1_appost_replnx4_12FuncFeb 22 2008 12:15PM

PDS.DBPrimary FunctionReplicate FunctionUsed by StandbyFunc_type
-----------------------------------------------------------------
PDS.pdb1R1R1_repNoApplied
```

```
ParameterDatatypeLengthSearchable
---------------------------------
aint40

Function NameFString ClassFString SourceFString Name
------------------------------------------------------------------------
--
R1rs_sqlserver_function_classClass DefaultR1

Subscriptions known at this Site 'ost_replnx4_12'.

Subscription NameReplicate DS.DBOwnerCreation Date
--------------------------------------------------

(return status = 0)
```

- **例 4 – create request function replication definition** を使用して作成された R1_req 複写定義に関する情報を表示します。

```
rs_helprep R1_req
```

```
Replication Definition NamePRSTypeCreation Date
-----------------------------------------------------------------------
R1_reqost_replnx4_12FuncFeb 22 2008 12:15PM

PDS.DBPrimary FunctionReplicate FunctionUsed by StandbyFunc_type
------------------------------------------------------------------------
-
PDS.pdb1R2R2_repNoRequest

ParameterDatatypeLengthSearchable
---------------------------------
aint4 0

Function NameFString ClassFString SourceFString Name
------------------------------------------------------------------------
--
R2rs_sqlserver_function_classClass Default R2

Subscriptions known at this Site 'ost_replnx4_12'.


Subscription NameReplicate DS.DBOwnerCreation Date
--------------------------------------------------------

(return status = 0)
```

- **例 5 –** 次のようなテーブルおよび複写定義を前提とします。

```
create table t1 (c1 int, c2 int)
```

```
create replication definition r1
with primary at ost_wasatch_08.pdb1
with all tables named t1
(c1 int, "c2" int quoted)
primary key (c1)
```

`rs_helprep r1` により、*c2* が引用符付き識別子として表示されます。

```
Replication Definition NamePRSType Creation Date
--------------------------- -------------------- ----
-------------------
r1ost_wasatch_09TblNov 11, 2008 2:28PM

PDS.DBPrimary OwnerPrimary Table
------------------- -------------------- --------------------
ost_wasatch_08.pdb1t1

Replicate OwnerReplicate Table
------------------- --------------------
t1

Send Min Cols. Used by Standby Min Vers Dynamic SQL SQL Stmt. Rep.
-------------- --------------- -------- -----------
--------------
NoNo1000OnNone

Col. NameRep. Col. NameDatatypeLen.Pri. Col.Searchable
------------------------------------------------------
c1c1int410
"c2""c2"int400

Function NameFString ClassFString SourceFString Name
------------------------------------------------------------------------
-
rs_deleters_sqlserver_function_classClass Defaultrs_delete
rs_insertrs_sqlserver_function_classClass Defaultrs_insert
rs_selectrs_sqlserver_function_classClass Defaultrs_select
rs_select_rs_sqlserver_function_classClass Defaultrs_select_
with_lockwith_lock
rs_truncaters_sqlserver_function_classClass Defaultrs_truncate
rs_updaters_sqlserver_function_classClass Defaultrs_update

Subscriptions known at this Site 'ost_wasatch_09'.
Subscription NameReplicate DS.DBOwnerCreation Date
--------------------------------------------------
(return status = 0)
```

- **例 6 –** 前の例で定義されたテーブルおよび複写定義を前提とし、*t1* を引用符付き識別子として定義します。

```
alter replication definition r1
alter replicate table name "t1" quoted
```

`rs_helprep r1` により、*c2* と *t1* が引用符付き識別子として表示されます。

```
Replication Definition NamePRSType Creation Date
--------------------------- -------------------- ----
-------------------
r1ost_wasatch_09TblNov 11, 2008 2:28PM

PDS.DBPrimary OwnerPrimary Table
------------------- -------------------- --------------------
ost_wasatch_08.pdb1"t1"

Replicate OwnerReplicate Table
```

```
-------------------- --------------------
"t1"

Send Min Cols. Used by Standby Min Vers Dynamic SQL SQL Stmt. Rep.
-------------- --------------- -------- -----------
--------------
NoNo1000OnNone

Col. NameRep. Col. NameDatatypeLen.Pri. Col.Searchable
------------------------------------------------------
c1c1int410
"c2""c2"int400

Function NameFString ClassFString SourceFString Name
-------------------------------------------------------------------
-
rs_deleters_sqlserver_function_classClass Defaultrs_delete
rs_insertrs_sqlserver_function_classClass Defaultrs_insert
rs_selectrs_sqlserver_function_classClass Defaultrs_select
rs_select_rs_sqlserver_function_classClass Defaultrs_select_
with_lockwith_lock
rs_truncaters_sqlserver_function_classClass Defaultrs_truncate
rs_updaters_sqlserver_function_classClass Defaultrs_update

Subscriptions known at this Site 'ost_wasatch_09'.
Subscription NameReplicate DS.DBOwnerCreation Date
--------------------------------------------------
(return status = 0)
```

- **例 7 –** 前の例で定義された複写定義を前提とし、*c2* を引用符付きではないとして定義します。

```
alter replication definition r1
alter columns c2 not quoted
```

`rs_helprep r1` により、*t1* のみが引用符付き識別子として表示されます。

```
Replication Definition NamePRSType Creation Date
--------------------------- -------------------- ----
-------------------
r1ost_wasatch_09TblNov 11, 2008 2:28PM

PDS.DBPrimary OwnerPrimary Table
-------------------- -------------------- --------------------
ost_wasatch_08.pdb1"t1"

Replicate OwnerReplicate Table
-------------------- --------------------
"t1"

Send Min Cols. Used by Standby Min Vers Dynamic SQL SQL Stmt. Rep.
-------------- --------------- -------- -----------
--------------
NoNo1000OnNone

Col. NameRep. Col. NameDatatypeLen.Pri. Col.Searchable
------------------------------------------------------
```

```
c1c1int410
c2c2int400

Function NameFString ClassFString SourceFString Name
-------------------------------------------------------------------
-
rs_deleters_sqlserver_function_classClass Defaultrs_delete
rs_insertrs_sqlserver_function_classClass Defaultrs_insert
rs_selectrs_sqlserver_function_classClass Defaultrs_select
rs_select_rs_sqlserver_function_classClass Defaultrs_select_
with_lockwith_lock
rs_truncaters_sqlserver_function_classClass Defaultrs_truncate
rs_updaters_sqlserver_function_classClass Defaultrs_update

Subscriptions known at this Site 'ost_wasatch_09'.
Subscription NameReplicate DS.DBOwnerCreation Date
-----------------------------------------------
(return status = 0)
```

- **例 8 – create function replication definition** を使用して作成した “authors” 複写定義に関する情報を表示するには、次のように入力します。

```
rs_helprep authors
```

出力の Ref Objowner カラムおよび Ref Objname カラムを確認します。

```
Replication Definition NamePRSType Creation Date
--------------------------- -------------------- ----
-------------------
authorsprimary_rsTblNov 26, 2008 1:48PM

PDS.DBPrimary OwnerPrimary Table
------------------- -------------------- --------------------
pds.pdbauthors

Replicate OwnerReplicate Table
------------------- --------------------
authors

Send Min Cols. Used by Standby Min Vers Dynamic SQL SQL Stmt.Rep.
-------------- --------------- -------- ----------- -------------
NoNo1000OnUD

Col. NameRep. Col. NameDatatypeLen.Pri. Col.Searchable
-----------------------------------------------------
au_idau_idvarchar1111
au_lnameau_lnamevarchar4001
au_fnameau_fnamevarchar2001

Ref. ObjownerRef. Objname
-------------------------


table2
```

使用法

- パラメータを指定しないと、**rs_helprep** は Replication Server にあるすべての複写定義の要約情報を表示します。
- *replication_definition* を指定すると、**rs_helprep** は *replication_definition* と一致する名前のすべての複写定義に関する情報を表示します。
- *replication_definition* が 1 つの複写定義の名前と完全に一致する場合、その複写定義の詳細情報を表示します。表示される情報には、プライマリ Replication Server、データサーバとデータベース、複写定義のカラム、複写定義に対して定義されているファンクション、Replication Server が認識している複写定義に対するサブスクリプションが含まれます。
- 表示される詳細情報は、テーブル複写定義、ファンクション複写定義、システムテーブル複写定義では多少異なります。
- *replication_definition* が 1 つの複写定義の名前と完全に一致しない場合、名前の最初の部分が *replication_definition* と一致するすべての複写定義の情報が表示されます。
- 引用符付き識別子は二重引用符で囲まれて表示されます。
- **rs_helprep** は、Real-Time Loading (RTL) および High Volume Adaptive Replication (HVAR) のテーブル参照に関する情報を表示します。
- **rs_helprep** はデータベース複写定義を表示しません。データベース複写定義を表示するには **rs_helpdbrep** を使用します。

rs_helprepdb

現在の Replication Server 内の、複写定義に対するサブスクリプションがあるデータベースに関して情報を表示します。

構文

```
rs_helprepdb [data_server, database]
```

パラメータ

- **data_server –** 情報を表示するデータベースのあるデータサーバです。
- **database –** 情報を表示するデータベースの名前です。

例

- **例 1 –** 現在の Replication Server 内の、複写定義に対するサブスクリプションがあるすべてのデータベースに関する情報を表示します。

  ```
  rs_helprepdb
  ```

```
dsnamedbname dbid controlling_prs
---------------- ------------- --------------------------
SYDNEY_DSSYDNEY_RSSD102SYNDEY_RS
```

- **例 2 –** 指定されたデータサーバとデータベースに関する情報を表示します。

```
rs_helprepdb SYDNEY_DS, pubs2
```

```
dsnamedbname dbid controlling_prs
---------------- ------------- --------------------------
SYDNEY_DS pubs2104 SYDNEY_RS
```

使用法

- **rs_helprepdb** は、プライマリ Replication Server の RSSD で実行してください。
- *data_server* と *database* パラメータを指定しないと、**rs_helprepdb** は、Replication Server のすべての複写定義に対するサブスクリプションがあるデータベースをすべて表示します。それぞれのデータサーバとデータベースに対し、データベース ID と管理している Replication Server が表示されます。
- *data_server* と *database* を指定すると、**rs_helprepdb** はそのデータベースに関する情報だけを表示します。

rs_helpreptable

プライマリテーブルに対して作成された複写定義についての情報を表示します。

構文

```
rs_helpreptable database, [owner,] {table | '*' }
```

パラメータ

- ***database* –** テーブルが作成されているデータベースです。
- ***owner* –** テーブルの所有者。
- **{*table* | '*' } –**
  - *table* - テーブルの名前。
  - * ワイルドカード文字 - すべてのテーブル。 同じ *owner* を持つテーブルを含む複写定義をすべてリストするには、ワイルドカードを使用します。

例

- **例 1 –** pdb1 データベース内の authors テーブルの複写定義に関する情報を表示します。

```
rs_helpreptable pdb1, authors
```

次のような内容が表示されます。

Replication Definition Name	Primary Owner	Primary Table	Replicate Owner	Replicate Table	Used By	Standby	Min Vers
authors	John	authors	John	authors_r1		Yes	1000
authors1	Bob	authors	Bob	authors_r2		No	1000

- **例 2 –** `pdb1` データベース内のテーブルについて、テーブル所有者が Mary である複写定義をすべてリストします。

```
rs_helpreptable pdb1, Mary, *
```

次のような内容が表示されます。

Replication Definition Name	Primary Owner	Primary Table	Replicate Owner	Replicate Table	Used By	Standby	Min Vers
r1	Mary	t1	Mary	t1		No	1150
r2	Mary	t2	Mary	t2		No	1150

使用法

- ユーザが定義したテーブル複写定義だけが表示されます。
- * ワイルドカード文字を使用すると、現在同じ *owner* を持つテーブル所有者を変更するために、**alter replication definition** でワイルドカードを使用した場合に変更される複写定義が、すべてリストされます。

rs_helprepversion

現在の Replication Server の複写定義バージョンについての情報を表示します。

構文

```
rs_helprepversion {repdef_name | repdef_version_id}
```

パラメータ

- **repdef_name –** 複写定義名です。
- **repdef_version_id –** 複写定義バージョン ID です。

例

- **例 1 –** 複写定義名 types11_pdb1 を指定すると、すべてのバージョンの複写定義に関する情報を表示します。

```
rs_helprepversion types11_pdb1
```

出力は次のようになります。

```
Repdef Version NameRepdef Version IDActiveActive
InboundOubound
------------------------------------------------------------------
-----
types11_pdb10x0107006500000067YesNo
rs_drp01060065000000674a955c450x0106006500000067NoYes
rs_drp01050065000000674a955c400x0105006500000067NoNo
rs_drp01040065000000674a955c3f0x0104006500000067NoNo
rs_drp01030065000000674a955c3d0x0103006500000067NoNo
rs_drp01020065000000674a955c3c0x0102006500000067NoNo
rs_drp01010065000000674a955c3b0x0101006500000067NoNo
rs_drp01000065000000674a955c3a0x0100006500000067NoNo
(return status = 0)
```

- **例 2 –** 複写定義バージョンを複写定義バージョン ID (この例では 0x0106006500000067) で指定する場合、**rs_helprepversion** は、複写定義バージョンの一般情報およびカラム情報を表示します。

```
rs_helprepversion 0x0106006500000067
```

出力は次のようになります。

```
Repdef Version NameRepdef Version IDActiveActive
InboundOubound
------------------------------------------------------------------
------
rs_drp01060065000000674a955c450x0106006500000067NoYes
```

```
ColumnReplicateDatatypeLenPriSearchableRefRef
NameColNameColObjownerObjname
----------- ----------- --------- ----- ---- ----------- ---------
-------
charcolcharcolvarchar25500
floatcolfloatcolfloat800
datecoldatecoldatetime800
smdatecolsmdatecolsmalldatet400
moneycolmoneycolmoney800
smmoneycolsmmoneycolsmallmoney400
intcolintcolint400
smintcolsmintcolsmallint200
tinyintcoltinyintcoltinyint100
row_numrow_numint410
(return status = 0)
```

使用法

rs_helprepversion は、複写定義バージョンについての情報を表示します。

- アクティブなインバウンド複写定義バージョン - エグゼキュータが、インバウンドキューにデータを配置するために使用します。
- アクティブなアウトバウンド複写定義バージョン - ディストリビュータが、アウトバウンドキューにデータを配置するために使用します。

参照：
- alter replication definition (199 ページ)
- alter applied function replication definition (129 ページ)
- alter request function replication definition (210 ページ)
- rs_helprep (720 ページ)
- rs_send_repserver_cmd (735 ページ)

rs_helproute

ルートのステータス情報を表示します。

構文

```
rs_helproute [replication_server]
```

パラメータ

- **replication_server –** ルートのステータス情報を表示する Replication Server の名前です。

例

- **例 1 –** TOKYO_RS から SYDNEY_RS へのルートは現在アクティブです。

```
rs_helproute
```

```
route route_status
-------------------------------------
TOKYO_RS -----> SYDNEY_RS Active
```

使用法

- *replication_server* を指定しないと、**rs_helproute** は現在の Replication Server で認識されているすべてのルートの情報を表示します。
- *replication_server* を指定すると、その Replication Server に接続されているルートの情報だけを表示します。
- Replication Server は、定義された手順を使用して、送信元と送信先の Replication Server 間のルートを作成したり削除したりします。この手順の実行

中、ルートの状態はさまざまに変化します。送信元または送信先の Replication Server の RSSD で **rs_helproute** を実行すると手順の現在の状態が表示されます。

- 各ルートに対し、**rs_helproute** は次の 2 種類の情報を表示します。
  - ルートのステータス
    ステータスにはルート処理手順の状態が反映されます。各ルートの情報は、**rs_helproute** をルートの送信元と送信先どちらの Replication Server で実行するかによって異なります。
  - システムテーブルサブスクリプションのリスト
    ルートを作成している場合は、作成中のシステムテーブルのサブスクリプションについての情報を表示します。ルートを削除している場合は、どのシステムテーブルのサブスクリプションが削除されているかがこのリストに表示されます。
    ルート処理手順では、通常、システムテーブルのサブスクリプションが処理されます。この情報は、どのサブスクリプションが原因で次のステップに移行できないのかを調べるのに便利です。システムテーブルのサブスクリプションが 1 つも表示されない場合、現在処理手順の障害となっているシステムテーブルサブスクリプションはありません。
    システムテーブルサブスクリプションのマテリアライゼーションまたはマテリアライゼーション解除が完了せずに問題が発生した場合も、同じようにこのコマンドを利用できます。たとえば、ルートの作成、削除、変更中に問題が発生したときは、**rs_helproute** を実行してサブスクリプションのステータスに関する情報を検証してください。

rs_helpsub

サブスクリプションについての情報を表示します。

構文

```
rs_helpsub
 [subscription_name [, replication_definition
    [, data_server, database]]]
```

パラメータ

- **subscription_name –** サブスクリプション名に対応する文字列です。文字列は、サブスクリプション名の全体または最初の部分と一致させてください。
- **replication_definition –** サブスクリプションに関連する複写定義の名前です。
- **data_server –** サブスクリプションのデータを含むデータベースが格納されているデータサーバです。
- **database –** サブスクリプションのデータが格納されているデータベースです。

例

- **例 1 –** 使用できるすべてのサブスクリプションの情報を表示します。RRS カラムの “Unknown” というステータスは、該当する Replication Server (PRS = プライマリ Replication Server) でのサブスクリプションのステータスを現在の Replication Server が認識していないことを示しています。

```
rs_helpsub
```

```
** This Site is primary_rs **
 Status at
Subscription Name Rep. Def. NameReplicate DS.DBA/C RRSPRS
----------------- --------------- ---------------- --- --------
------
authors_1authorsRDS.rdb0UnknownValid
many_rows_1many_rowsRDS.rdb0UnknownValid
publishers_1publishersRDS.rdb0UnknownValid
titleauthor_1titleauthorRDS.rdb0UnknownValid
titles_1titlesRDS.rdb0UnknownValid

Dynamic SQL
-----------
On
On
On
On
On
(return status = 0)
```

- **例 2 –** *authors_sub* サブスクリプションについての詳細情報を表示します。

```
rs_helpsub authors_sub
```

```
Subscription Name Rep. Def. NameReplicate DS.DBA/C RRSPRS
----------------- --------------- ---------------- --- --------
------
authors_subauthors_repRDS.rdb0 DefinedUnknown

Dynamic SQLOwner Creation Date
------------------------------------------
OnsaOct 2 2007

SubscriptionText
------------------------------------------------------------------
----
 create subscription authors_sub
 for authors_rep
 with replicate at RDS.rdb
 where
 state = "CA"
(return status = 0)
```

使用法

- パラメータを何も指定しないと、**rs_helpsub** は Replication Server に定義されているすべてのサブスクリプションの要約情報を表示します。この情報には、複写定義、レプリケートデータサーバとレプリケートデータベース、オートコレクションのステータス、レプリケート Replication Server とプライマリ Replication Server でのサブスクリプションマテリアライゼーションのステータスが含まれます。
- *subscription_name* を指定すると、**rs_helpsub** は *subscription_name* と一致する名前のサブスクリプションに関する情報を表示します。
- *subscription_name* が 1 つのサブスクリプションの名前と完全に一致する場合、そのサブスクリプションの所有者、作成日、テキストも表示します。
- *subscription_name* が 1 つのサブスクリプション名と完全に一致しない場合、名前の最初の部分が *subscription_name* と一致するすべてのサブスクリプションの情報を表示します。
- *replication_definition* も指定すると、**rs_helpsub** はその複写定義に対するサブスクリプションの情報だけを表示します。
- **rs_helpsub** はサブスクリプション複写定義を表示しません。サブスクリプション複写定義を表示するには **rs_helpdbsub** を使用します。

rs_helpuser

Replication Server が認識しているユーザログイン名についての情報を表示します。

構文

```
rs_helpuser [user]
```

パラメータ

- **user –** 情報を表示するユーザのログイン名です。

例

- **例 1 –** すべてのユーザについての情報を表示します。

```
rs_helpuser
```

```
 Users and Privileges Known at Site repl_rs
Primary Users
User NamePermission(s) Name
---------------------- ------------------------------
TOKYO_RS_id_user no grants
sa sa
TOKYO_RS_raconnect source
TOKYO_RS_rsi connect source
```

```
repusercreate object
TOKYO_RSSD_primconnect source, primary subscr

Maintenance Users
User nameDestination DS.DB
------------------------------------------------------------------
---
TOKYO_RSSD_maintTOKYO_DS.TOKYO_RSSD
pubs2_maintTOKYO_DS.pubs2
pubs2_maintSYDNEY_DS.pubs2sb
```

- **例 2 –** *pubs2_maint* ユーザについての情報を表示します。

```
rs_helpuser pubs2_maint
```

```
 Users and Privileges Known at Site TOKYO_RS
Primary User(s)
User NamePermission Name
---------------------- ------------------------------
pubs2_maintTOKYO_DS.pubs2
pubs2_maintSYDNEY_DS.pubs2sb
```

使用法

- パラメータを指定しないと、**rs_helpuser** は現在の Replication Server で認識されているすべてのユーザログイン名についての情報を表示します。
- *user* にログイン名を指定すると、**rs_helpuser** はそのユーザログイン名の情報だけを表示します。

rs_init_erroractions

新しいエラークラスを初期化します。

注意： **rs_init_erroractions** は今後廃止されます。 新しいクラスを初期化するには、**set template to** オプションを指定して **create error class** を使用することをおすすめします。

構文

```
rs_init_erroractions new_error_class, template_class
```

パラメータ

- **new_error_class –** 作成した新しいエラークラスの名前です。
- **template_class –** 新しいエラークラスのテンプレートとして使用するエラークラスの名前です。

例

- **例 1 –** テンプレートエラークラス *rs_sqlserver_error_class* をベースにしてエラークラス *new_class* を作成します。

```
rs_init_erroractions new_class, rs_sqlserver_error_class
```

使用法

- テンプレートエラークラスには、ユーザ定義のエラークラスまたは *rs_sqlserver_error_class* などのシステムが提供するエラークラスを指定できます。
- 新しいエラークラスをそのエラークラスのプライマリ Replication Server に作成するには、**create error class** コマンドを使用してください。そのうえで、**rs_init_erroractions** を使用してそのエラークラスを初期化します。

参照：

- create error class (308 ページ)

rs_send_repserver_cmd

プライマリデータベースで複写定義の変更要求を直接実行します。

構文

```
rs_send_repserver_cmd 'rs_api'
```

パラメータ

- **rs_api – rs_send_repserver_cmd** のために指定する、複写定義の複写コマンド言語 (RCL) のコマンドおよびパラメータが格納されます。*rs_api* は、*varchar* パラメータで、最大長は 16370 バイト (Adaptive Server)、4000 バイト (Oracle)、および 8000 バイト (Microsoft SQL Server) です。

  rs_api を一重引用符で囲み、文字列内の各一重引用符を二重引用符で置き換えます。

  rs_api のパラメータの長さが create または alter replication definition 要求に対して短すぎる場合は、要求を 2 つ以上に分割できます。

例

- **例 1 –** “authors” の **alter replication definition** 要求をプライマリデータベースで実行し、住所、都市、州、郵便番号のカラムを削除します。

```
exec rs_send_repserver_cmd 'alter replication
definition authors drop address, city, state, zip'
```

- **例 2 –** 複写定義 RCL が *rs_api* で使用できる最大長を超える場合は、要求を 2 つ以上に分割できます。

```
exec rs_send_repserver_cmd 'alter replication
definition authors drop address, city'
go
exec rs_send_repserver_cmd 'alter replication
definition authors drop state, zip'
```

- **例 3 –** この例では、"authors" を二重引用符で囲み、'off' を 2 つの一重引用符で囲む必要があります。

```
exec rs_send_repserver_cmd 'alter replication definition
"authors" replicate sqldml ''off'''
```

使用法

- プライマリデータベースで **rs_send_repserver_cmd** を使用する前に、**admin verify_repserver_cmd** を使用して複写定義要求を Replication Server で正常に実行できることを確認します。
- Replication Server では、次の複写定義コマンドの **rs_send_repserver_cmd** がサポートされています。
  - **alter replication definition**
  - **create replication definition**
  - **drop replication definition**
  - **alter applied function replication definition**
  - **create applied function replication definition**
  - **alter request function replication definition**
  - **create request function replication definition**

  注意： Adaptive Server の他に、Replication Server では、**rs_send_repserver_cmd** のサポートが、これらの ASE 以外のデータベースのサポートされているバージョン(Microsoft SQL Server および Oracle) まで拡張されています。サポートされているデータベースバージョンについては、Replication Agent の『リリースノート』を参照してください。
- プライマリデータベースで **rs_send_repserver_cmd** を実行する場合、Replication Agent は *rs_api* に格納された RCL コマンドを Replication Server に送信し、RCL コマンドを実行します。これにより、Replication Server は適切な複写定義バージョンを使用してプライマリデータを複写します。つまり、**rs_send_repserver_cmd** より前のプライマリデータは古い複写定義バージョンで複写され、**rs_send_repserver_cmd** より後のプライマリデータは新しい複写定義バージョンで複写されます。

- 必ずしも、プライマリデータサーバから直接、複写定義の変更要求を発行する必要はありません。たとえば以下のような状況では、プライマリ Replication Server から直接、**alter replication definition** 要求を実行できます。
  - 複写定義へのサブスクリプションがない
  - 複写定義へのサブスクリプションはあるが、プライマリデータベースログにテーブルまたはストアドプロシージャのデータがない
  - テーブル複写定義との間でサーチャブルカラムを追加または削除する
  - ファンクション複写定義との間でサーチャブルパラメータを追加または削除する
  - 動的 SQL をオンまたはオフにするために複写定義を変更する

警告！ Replication Server は、Replication Agent によって Replication Server に送信されるすべてのコマンドを受け入れるため、プライマリデータベースで **rs_send_repserver_cmd** へのアクセスを制御する必要があります。

参照：
- admin verify_repserver_cmd (101 ページ)
- alter replication definition (199 ページ)
- create replication definition (346 ページ)
- drop replication definition (419 ページ)
- alter applied function replication definition (129 ページ)
- create applied function replication definition (274 ページ)
- alter request function replication definition (210 ページ)
- create request function replication definition (362 ページ)
- sysadmin skip_bad_repserver_cmd (505 ページ)

rs_ticket

Replication Server のパフォーマンス、モジュールのハートビート、複写の正常性、テーブルレベルのクワイスをモニタする、プライマリデータベースのストアドプロシージャです。

構文

```
rs_ticket h1 [, h2 [, h3 [, h4]]]
```

パラメータ

- **h1 [, h2 [, h3 [, h4]]]** – 短い *varchar* 文字列のヘッダ情報です。

例

- **例 1 –** 定期的に **rs_ticket** を実行します。

```
Exec rs_ticket 'heartbeat', 'beat-sequence-number'
```

- **例 2 –** パフォーマンスを測定するには、プライマリデータベースから次のコマンドを実行します。

```
Exec rs_ticket 'start'
Execute replication benchmarks
Exec rs_ticket 'stop'
```

使用法

- **rs_ticket** ストアドプロシージャのチケットバージョン番号は V=2 で、チケットサイズは 1024 バイトです。
- アプリケーションがバージョン 1 のチケットのみを認識する場合は、**rs_ticket_v1** を呼び出してバージョン 1 フォーマットのチケットを生成します。**rs_ticket_v1** の構文は次のとおりです。

```
rs_ticket_v1 h1 [, h2 [, h3 [, h4]]]
```

- **rs_ticket** は次のコマンドを実行します。

```
rs_marker 'rs_ticket rs_ticket_param'
```

  不正なフォーマットの **rs_marker** を発行することを避け、*rs_ticket_param* 標準を実行するには、**rs_marker** ではなく **rs_ticket** を呼び出す必要があります。**rs_marker** を直接呼び出して無効な **rs_marker** サブコマンドを作成すると、Replication Server は **rs_marker** を拒否して RepAgent のコネクションを停止します。この場合は、データが失われる可能性があるトランザクションログから **rs_marker** をスキップする必要があります。
- Replication Server EXEC、DIST、RSI、DSI モジュールは **rs_ticket** サブコマンドを解析および処理します。
  - EXEC は、**rs_ticket** を処理するときに、*rs_ticket_param* の後ろにタイムスタンプと RepAgent から受信した総バイト数を追加します。EXEC タイムスタンプの形式は "EXEC(spid)=mm/dd/yy hh:mm:ss.ddd" です。バイト情報は "B(spid)=ddd" です。EXEC はインバウンドキューに **rs_ticket** を書き込みます。
  - DIST は、**rs_ticket** を処理するときに、*rs_ticket_param* に別のタイムスタンプを追加します。DIST タイムスタンプの形式は "DIST(spid)=hh:mm:ss.ddd" です。
  - RSI は、**rs_ticket** を処理するときに、*rs_ticket_param* に別のタイムスタンプを追加します。RSI タイムスタンプの形式は "RSI(spid)=mm/dd/yy hh:mm:ss.ddd" です。

    - DSI は、**rs_ticket** を処理するときに、*rs_ticket_param* に別のタイムスタンプを追加します。DSI タイムスタンプの形式は "DSI(spid)=hh:mm:ss.ddd" です。
- **rs_ticket** についてのサブスクリプションはありません。レプリケートサイトから少なくとも 1 つのサブスクリプションがないかぎり、DIST は DSI に対して **rs_ticket** を送信しません。
- **rs_ticket** は低負荷で非介入型であるため、テスト環境と運用環境の両方で使用できます。
- **rs_ticket** によって、Replication Server をクワイスせずに、データがいつ複写パスから完全にフラッシュされたのかを把握できます。
- **rs_ticket** の移動は、RSTicket カウンタを介して EXEC、DIST、RSI、DSI スレッドによって追跡されます。各スレッドには、対応するスレッドが **rs_ticket** を受信するたびに 1 つずつ増加する RSTicket カウンタが 1 つあります。このカウンタはリセットされません。
  RSTicket カウンタをサンプリングして、**rs_ticket** が到達したモジュールをモニタできます。 SAP Control Center® for Replication またはその他の SAP Replication Server のモニタツールは、これらのカウンタを使用して EXEC、DIST、RSI、DSI ハートビートを生成します。
  プライマリで **rs_ticket** を送信して RSTicket カウンタを確認しても、複写パスの状態をモニタできます。モジュールの RSTicket カウンタが増加していないときは、この段階で複写パスが中断されています。
- **rs_ticket** を複写するようマーク付けしないでください。
- SAP Replication Server が 15.0 以降の場合にのみ、**rs_ticket** を使用します。

参照：

- rs_ticket_report (583 ページ)
- rs_ticket_history (839 ページ)

rs_zeroltm

データベースのロケータ値をゼロ (0) にリセットします。このストアドプロシージャは、Adaptive Server コマンドの **dbcc settrunc** でセカンダリトランケーションポイントを無効にしてログをトランケートした後、Replication Server を再起動する前に実行します。

構文

```
rs_zeroltm data_server, database
```

パラメータ

- **data_server –** ロケータ値をリセットするデータベースが格納されているデータサーバです。
- **database –** ロケータ値をリセットするデータベースです。

例

- **例 1 –** TOKYO_DS データサーバの *pubs2* データベースのロケータ値を 0 にリセットします。

```
rs_zeroltm TOKYO_DS, pubs2
```

使用法

- このコマンドは、RepAgent が有効なデータベースに使用します。
- **rs_zeroltm** を使用する前に、**dbcc settrunc** でセカンダリトランケーションポイントを無効にしてログをトランケートしてください。
- 複写データベースのロケータ値は Replication Server が管理し、*rs_locater* テーブルに格納されています。ロケータ値は、通常 Adaptive Server に格納されているセカンダリトランケーションポイントの値と同じです。
  トランザクションログが満杯になると、**dbcc settrunc** コマンドでセカンダリトランケーションポイントを無効にして、ログをトランケートしなければならないことがあります。**dbcc settrunc** によってセカンダリトランケーションポイントがリセットされるため、ロケータ値とセカンダリトランケーションポイントは一致しなくなります。その場合、**rs_zeroltm** を実行して 2 つの値を再び同期化させます。**rs_zeroltm** でロケータ値をゼロに設定すると、Replication Server によって Adaptive Server から新しいセカンダリトランケーションポイントが取得され、ロケータがその値に設定されます。

参照：

- dbcc settrunc (602 ページ)

実行プログラム

Replication Server の実行プログラムについて説明します。実行プログラムには、Replication Server および **rs_subcmp** プロシージャが含まれます。

repserver

Replication Server の実行プログラムです。

構文

```
{repserver | repsrvr} [-C config_file] [-i id_server]
[-S rs_name] [-I interfaces_file]
[-E errorlog_file] [-M] [-v] [-K keytab_file]
[-k rs_principal_name]
[-upgr] [-Aerssd_release_dir] [-purgeq]
[nodb {all|dbid_1[,dbid_2[,dbid_3[,…]]]}
[-e]
```

パラメータ

- **-C *config_file*** – Replication Server 設定ファイルの名前とロケーションを指定します。 **rs_init** プログラムは、デフォルトでは *Rep_Server_name*`.cfg` (*Rep_Server_name* は Replication Server の名前) という名前の設定ファイルを作成します。 このファイル名は、**-C** フラグを使用して指定できます。 **-C** フラグを指定しないと、**repserver** は Replication Server を起動したディレクトリの `config.rs` という名前の設定ファイルを検索します。
- **-i *id_server*** – その複写システムで使用する ID サーバの名前を指定します。 ID サーバは、最初に起動する Replication Server にしてください。 ID サーバが実行中でアクセス可能でないと、新しい Replication Server は起動できません。 ID サーバの名前は設定ファイルに格納されます。 別の ID サーバを指定するには、**-i** オプションを使用してください。
- **-S *rs_name*** – 使用する現在の Replication Server の名前を指定します。 ネットワークベースのセキュリティと統一化ログインが有効になっている場合は、プリンシパルユーザの名前を指定します。
- **-I *interfaces_file*** – Replication Server が定義されている interfaces ファイルの名前とロケーションを指定します。 interfaces ファイルには、現在の Replication Server が通信するデータサーバと他の Replication Server に関するエントリも含まれていなければなりません。 レプリケートサイトの interfaces ファイルには、プライマリ Replication Server とプライマリデータサーバのエントリを登録しま

す。 **-I** フラグを指定しないと、Replication Server はリリースディレクトリ内のデフォルトの Interfaces ファイルを検索します。

使用しているプラットフォームのデフォルト interfaces ファイルを含む interfaces ファイルの詳細については、各プラットフォーム用の『Replication Server インストールガイド』と『Replication Server 設定ガイド』を参照してください。

- **-E *errorlog_file* – repserver** がエラーメッセージを書き込む Replication Server エラーログファイルの名前とロケーションを指定します。 **-E** フラグを指定しないと、デフォルトのエラーログファイルの名前とロケーションは、それぞれ、`repserver.log` と Replication Server を起動したディレクトリになります。
- **-M –** Replication Server をスタンドアロンモードで起動します。これは、リカバリ処理を開始するために使用します。スタンドアロンモードでの Replication Server の実行の詳細については、『Replication Server 管理ガイド第 2 巻』を参照してください。
- **-v –** Replication Server のバージョン番号を出力します。
- **-K *keytab_file* –** サーバにログインするユーザのセキュリティクレデンシャルが格納されている、DCE keytab ファイルの名前とロケーションを指定します。 keytab ファイルは、DCE の **dcecp** ユーティリティを使用して作成できます。 詳細については DCE のマニュアルを参照してください。

注意： **-K***keytab_file* オプションは、Windows プラットフォームと DCE ネットワークセキュリティを使用する場合にのみ指定します。

- **-k rs_principal_name –** Kerberos キー配布センター (KDC) によって認証される Replication Server プリンシパル名を指定します。

注意： Kerberos セキュリティメカニズムを有効にして Replication Server を起動すると、Replication Server では最初に **k rs_principal_name** オプションで指定されているプリンシパル名を Kerberos 認証に使用します。 **-k rs_principal_name** オプションが指定されていない場合、Replication Server は SYBASE_RS_PRINCIPAL 環境変数で設定されたプリンシパル名を探します。 いずれも指定されていない場合、Replication Server はサーバ名を認証に使用します。

『Replication Server 管理ガイド: 第 1 巻』の「Replication Server のプリンシパル名の指定」を参照してください。

- **-upgr –** Replication Server にアップグレードモードで開始するよう指示します。
- **-A*erssd_release_directory* –** Replication Server が ERSSD を使用している場合に、アップグレードする ERSSD のリリースディレクトリのロケーションを指定します。 たとえば、Replication Server をバージョン 15.5 から 15.7.1 SP100 にアップグレードする場合は、次のようになります。

- UNIX - `/sybase/REP-15_5/ASA11`
- Windows - `c:¥sybase¥REP-15_5¥ASA11`

-A オプションを含めなかった場合に、設定ファイルに情報が含まれる場合は、Replication Server は Replication Server 設定ファイルからリリースディレクトリのロケーションを取得します。**-A** オプションを指定した場合は、**repserver** または **repsrvr.exe** コマンドで手動で指定した内容によって構成ファイルの設定がオーバーライドされるため、Replication Server は設定ファイルのリリースディレクトリのロケーションを無視します。

- **-purgeq –** インバウンドキューからトランザクションをパージします。15.5 より前のバージョンの Replication Server からアップグレードする場合は、このオプションを使用する必要があります。
- **-nodb all –** アップグレードプロセスからすべてのユーザデータベースを除外します。
- **-nodb *dbid_1*[,*dbid_2*[,*dbid_3*[,...]]] –** アップグレードプロセスから特定のデータベースを除外します。複数のデータベース ID はコンマで区切り、ID 間にはスペースを入れないでください。例：

```
repserver upgr . . .  -A . . .  -nodb 101,102,105
```

- **-e –** アップグレードのために **-upgr** パラメータを入力したときに、Replication Server がデータサーバに送信する SQL 文を記録します。**-e** オプションを使用しない場合、生成された SQL 文は記録されません。**-e** オプションを使用するかどうかに関係なく、アップグレードプロセスでは、Replication Server エラーログファイルを使用して、アップグレードが開始される前に `rs_config` テーブルに格納された現在の設定パラメータの設定、アップグレードプロセス時に発生したエラーと、ユーザデータベースがアップグレードされなかった理由が記録されます。ダウングレードする必要がある場合に、前の設定をリストアするには、エラーログファイルを確認します。

例

- **例 1 –** TOKYO_RS という名前の Replication Server を、設定ファイル *TOKYO_RS.cfg* を使用して開始します。

```
repserver -STOKYO_RS -CTOKYO_RS.cfg
```

- **例 2 –** SYDNEY_RS という名前の Replication Server を、設定ファイル *SYDNEY_RS.cfg* を使用して開始します。*TOKYO_RS* は複写システムの ID サーバです。

```
repserver -SSYDNEY_RS -CSYDNEY_RS.cfg -iTOKYO_RS
```

- **例 3 –** Replication Server を起動し、interfaces ファイル `my_newinterfaces` を指定します。このファイルは、デフォルトの interfaces ファイルまたは LDAP ディレクトリサービスよりも優先します。

```
repserver -STOKYO_RS _CTOKYO_RS.cfg
    -I$SYBASE/SYBASE_RS/my_newinterfaces
```

- **例 4 –** NY_RS Replication Server を起動し、`/sybase/REP-15_5/ASA11` ERSSD リリースディレクトリのロケーション、RSSD *ny_rs.cfg* 設定ファイル、`my_newinterfaces` Interfaces ファイル、および `ny_rs_errorlog` エラーログファイルを使用してアップグレードします。

```
repserver upgr SNY_RS A/sybase/REP-15_5/ASA11 Cny_rs.cfg
Imy_newinterfaces E ny_rs_errorlog
```

使用法

- Unix で **repserver** を使用するか、または Windows で **repsrvr.exe** を使用して、Replication Server 実行プログラムを起動します。通常は、**rs_init** で作成されたファイルを実行することによって、Replication Server を開始します。便宜上、**repserver** は両方のプラットフォーム上のコマンドを参照します。
- **repserver** 実行プログラムは、リリースディレクトリの `bin` サブディレクトリにあります。 詳細については、使用しているプラットフォーム用の『Replication Server インストールガイド』および『Replication Server 設定ガイド』を参照してください。
- **repserver** コマンドは “sybase” ユーザが実行してください。これにより、Replication Server がそのディスクパーティションにアクセスできます。
- interfaces ファイルには、現在の Replication Server が通信する他の Replication Server とデータサーバに関する定義も含まれていなければなりません。レプリケートサイトの interfaces ファイルには、プライマリ Replication Server とプライマリデータサーバのエントリを登録します。
- パスワードが暗号化された形式で保存されている場合、Replication Server 設定ファイルで直接このパスワードを編集することはできません。 このファイル内の暗号化パスワードを変更するには、**rs_init** プログラムを使用してください。 詳細については、使用しているプラットフォーム用の『Replication Server インストールガイド』および『Replication Server 設定ガイド』を参照してください。
- *RSSD_primary_user* と *RSSD_maint_user* は、Replication Server の設定時に自動的に *rs_systabgroup* グループに (**rs_init** を使用して) 割り当てられます。このため、これらのユーザはシステムテーブルを変更できます。Adaptive Server システムプロシージャ **sp_changegroup** を使用して、このグループにその他のユーザログイン名を追加できます。詳細については、『Adaptive Server Enterprise システム管理ガイド』を参照してください。
- RSSD に対してネットワークベースセキュリティパラメータが指定されている場合、**use_security_services** パラメータが “on” に設定され、ネットワークベースセキュリティが自動的に起動されます。

- Replication Server をアップグレードする場合は、**-upgr** パラメータを使用します。**-upgr** オプションを使用する場合は、**-A**、**-purgeq**、**-nodb**、および **-e** オプションのみを使用できます。『Replication Server 設定ガイド』の「repserver を使用した RSSD または ERSSD およびユーザデータベースのアップグレード」を参照してください。
- **-k rs_principal_name** オプションを使用して Replication Server プリンシパル名を指定します。

表 50 : Replication Server の設定ファイルパラメータ

設定パラメータ	説明
CONFIG_charset	Replication Server 設定ファイルを記述するために使用する文字セット。このパラメータは、Replication Server とは違う文字セットを使用する場合にだけ設定する。Replication Server の文字セットと互換性のある文字セットを使用できる。
erssd_backup_dir	ERSSD バックアップディレクトリ。
erssd_dbfile	ERSSD データベースファイル。
erssd_errorlog	ERSSD エラーログ。
erssd_logmirror	ERSSD トランザクションログミラーファイル。
erssd_ping_cmd	ERSSD に ping を発行するために別のコマンドを指定できる。デバッグ目的のみ。
erssd_port	ネットワークリスナの ERSSD ポート番号。ポート番号は interfaces ファイルから取得される。
erssd_release_dir	別のリリースディレクトリを指定できる。デバッグ目的のみ。デフォルトは `$SYBASE/$SYBASE_REP/ASA11`。
erssd_ra_release_dir	ERSSD Replication Agent に別のリリースディレクトリを指定できる。デバッグ目的のみ。
erssd_ra_start_cmd	ERSSD Replication Agent を起動するために別のコマンドを指定できる。デバッグ目的のみ。
erssd_start_cmd	ERSSD を起動するために別のコマンドを指定できる。デバッグ目的のみ。
erssd_translog	ERSSD トランザクションログファイル。
ID_pw	ID サーバユーザ (*ID_user*) のパスワード。
ID_pw_enc	ID サーバユーザ (*ID_user*) の暗号化されたパスワード。

設定パラメータ	説明
ID_server	複写システムの ID サーバに指定されている Replication Server の名前。
ID_user	他の Replication Server が使用する ID サーバのログイン名。
RS_charset	Replication Server が使用する文字セット。 SAP がサポートしている文字セットを指定できます。 複写システムを設定する場合には、同じ Replication Server サイトにあるすべてのサーバで、できるだけ同じ文字セットを使用することを推奨 (ただし必須ではない)。また、複写システムにあるすべての Replication Server で互換性のある文字セットを使用することを推奨。 詳細については、『Replication Server デザインガイド』を参照してください。
RS_language	Replication Server がエラーログファイルとクライアントにメッセージを出力するときに使用する言語。Replication Server がローカライズされている言語で、使用している文字セットと互換性のある言語を指定できる。
RS_send_enc_pw	RSSD との最初のコネクションを除く、すべての Replication Server クライアントコネクションで暗号化パスワードが使用されるようにする。値は on または off。 デフォルト値は off
RS_sortorder	Replication Server が使用するソート順。ソート順は、文字データを扱った **where** 句を含むサブスクリプションで、テーブルのどのローが選択されるかを制御する。また、ユーザが入力する識別子の認識方法も制御する。 SAP がサポートするソート順のうち、指定した文字セットと互換性のあるソート順を指定できます。 複写システムのすべてのソート順を同じにする必要があります。
RS_unicode_sort_order	Replication Server が使用する Unicode ソート順。 SAP がサポートしている Unicode ソート順を指定できます。 デフォルト値は binary
RSSD_database	RSSD の名前。
RSSD_embedded	RSSD が埋め込まれているかどうかを示す。
RSSD_ha_failover	高可用性フェールオーバが許可されているかどうかを示す。 デフォルト値は No

設定パラメータ	説明
RSSD_maint_pw	RSSD メンテナンスユーザのパスワード。
RSSD_maint_pw_enc	RSSD メンテナンスユーザの暗号化されたパスワード。
RSSD_maint_user	RSSD メンテナンスユーザのログイン名。このログイン名は、システムテーブルを変更できる *rs_systabgroup* グループに自動的に登録される。 Adaptive Server システムプロシージャ **sp_changegroup** を使用して、このグループにその他のユーザログイン名を追加できます。詳細については、『Adaptive Server Enterprise システム管理ガイド』を参照してください。
RSSD_msg_confidentiality	Replication Server が暗号化データを送受信するかどうかを指定する。“required” に設定すると、送信データと受信データの暗号化が必要になる。“not_required” に設定すると、送信データは暗号化されず、受信データは暗号化されていてもされていなくてもよい。このオプションは実装されていない。 デフォルト値は not_required
RSSD_msg_integrity	データの改ざんをチェックするかどうかを指定する。有効な入力は、“required” および “not_required”。このオプションは実装されていない。 デフォルト値は not_required
RSSD_msg_origin_check	データのオリジンをチェックするかどうかを指定する。有効な入力は、“required” および “not_required”。このオプションは実装されていない。 デフォルト値は not_required
RSSD_msg_replay_detection	データが読み取られたり傍受されていないことをチェックするかどうかを指定する。有効な入力は、“required” および “not_required”。このオプションは実装されていない。 デフォルト値は not_required
RSSD_msg_sequence_check	データの順番が変更されていないことをチェックするかどうかを指定する。有効な入力は、“required” および “not_required”。このオプションは実装されていない。 デフォルト値は not_required
RSSD_mutual_auth	Replication Server がコネクションを確立する前に、RSSD が身元の確認を行うかどうかを指定する。有効な入力は、“required” および “not_required”。このオプションは実装されていない。 デフォルト値は not_required

設定パラメータ	説明
RSSD_primary_user	RSSD プライマリユーザのログイン名。インストール時に **rs_init** は、このユーザに自動的に *rs_systabgroup* グループを割り当てる。 他のユーザログイン名をこのグループに追加するには、Adaptive Server のシステムプロシージャ **sp_changegroup** を使用する。詳細については、『Adaptive Server Enterprise システム管理ガイド』を参照してください。
RSSD_primary_pw	RSSD プライマリユーザのパスワード。
RSSD_primary_pw_enc	RSSD プライマリユーザの暗号化されたパスワード。
RSSD_sec_mechanism	Replication Server が起動時に、RSSD と初めて通信するときに使用するセキュリティメカニズム。その後、RSSD との通信のためのネットワークセキュリティ情報は `rs_config` ファイルから読み込まれる。このオプションは実装されていない。
RSSD_server	RSSD のある Adaptive Server の名前。
RS_ssl_identity	SSL ID ファイル。
RS_ssl_pw	SSL プライベートキーのパスワード。
RS_ssl_pw_enc	SSL プライベートキーの暗号化されたパスワード。
RSSD_unified_login	Replication Server が起動時に、RSSD との接続にクレデンシャルを要求するかどうかを指定する。その後、RSSD との通信のためのネットワークセキュリティ情報は `rs_config` ファイルから読み込まれる。有効な入力は、“required” および “not_required”。このオプションは実装されていない。 デフォルト値は not_required

trace	Replication Server のトレースをオンにする。このパラメータの複数のインスタンスを使用して、異なるトレースを使用可能に設定できる。スペースは使用できない。例：
	`trace=DSI,DSI_BUF_DUMP`
	`trace=DIST,DIST_TRACE_COMMANDS`

trace_file	Replication Server ログファイルの名前を示す。

rs_subcmp

複写テーブルのデータをテーブルのプライマリバージョンと比較する実行プログラムです。**rs_subcmp** は、複写テーブルとプライマリテーブル間および複写デー

タベースとプライマリデータベース間のスキーマ比較も実行します。これらの機能は、ローとスキーマの欠落、孤立、不整合を検索し、オプションでそれらを調整するのに役立ちます。UNIX システムでは、この実行プログラムは **rs_subcmp** という名前です。Windows システムでは、この実行プログラムは **subcmp** という名前です。

rs_subcmp プログラムは、リリースディレクトリの `bin` サブディレクトリにあります。 詳細については、使用しているプラットフォームの『Replication Server インストールガイド』と『Replication Server 設定ガイド』を参照してください。

rs_subcmp を機能させるには、SYBASE 環境変数、ライブラリパス環境変数、および SYBASE_JRE6 を設定する必要があります。 **rs_subcmp** をスキーマ比較に使用する場合は、**rs_subcmp** を使用して **ddlgen** 実行ファイルを特定できること、および Replication Server 環境で **ddlgen** を正常に実行できることを確認します。 手順については「使用法」を参照してください。

rs_subcmp は SAP データベースだけを整理統合します。 Adaptive Server 15.7 SP100 以降のスキーマ比較に **rs_subcmp** を使用することはできません。 Adaptive Server 15.7 SP100 以降のスキーマ比較には、SAP® Replication Server® Data Assurance オプションを使用してください。 Replication Server Data Assurance オプションのマニュアルを参照してください。

構文

```
rs_subcmp [-R | -r] [-v] [-V] [-z[1 | 2] [-g] [-h]]
[-f config_file] [-F]
-S primary_ds [-D primary_db]
-s replicate_ds [-d replicate_db]
-t table_name [-T primary_table_name]
-c select_command [-C primary_select_command]
-u user [-U primary_user]
[-p passwd] [-P primary_passwd]
[-B primary_init_batch]
[-b replicate_init_batch]
[-n num_iterations] [-w wait_interval]
[-e float_precision] [-E real_precision]
[-k primary_key_column [-k primary_key_column]...]
[-i identity_column]
[-l text_image_column_name
[-l text_image_column_name]...]
[-L text_image_length_in_kilobytes]
[-N text_image_column_name
[-N text_image_column_name]...]
[-Z language]
[-o sort_order]
[-O sort_order]
[-J rs_subcmp_charset]
[-j rep_charset]
[-a replicate_column_name primary_column_name
[-a replicate_column_name primary_column_name]...]
[-q unicode_sort_order]
```

```
[-Q unicode_sort_order]
[-x schema_flag]
[-X filter_flag]
[-I interface_file]
[-H normalization_option]
```

パラメータ

- **-R –** プライマリデータベースで最終的にデータの不整合を検証してから、レプリケートデータをプライマリデータに合わせて調整します。**rs_subcmp** は、レプリケートデータがプライマリデータと一致するように、レプリケートデータベースでローを挿入、削除、更新します。
- **-r –** レプリケートデータをプライマリデータに合わせて調整しますが、**-R** で実行するような、プライマリデータベースでのデータの不整合の最終的な検証は行いません。**rs_subcmp** は、レプリケートデータがプライマリデータと一致するように、レプリケートデータベースでローを挿入、削除、更新します。
- **-v –** バージョン情報を出力します。
- **-V –** 比較結果を画面 (標準出力) に出力します。**-V** フラグを指定しない場合、**rs_subcmp** はローの相違点をレポートしません。*text*、*unitext*、または *image* データの値は出力されませんが、**rs_subcmp** は、データの不整合が *text*、*unitext*、または *image* のカラムにあるのか、他のデータ型のカラムにあるのかをレポートします。
- **-z –** トレースを有効にします。**-z1** (デフォルト) では、カラム見出しの比較など、基本的なトレース情報を生成します。また、**-z1** は、数値の精度の違いについての情報も出力します。**-z2** は、すべてのローとコマンドを比較して、トレース情報を生成します。
- **-f *config_file* – rs_subcmp** の設定ファイル名を指定します。
- **-F –** *config_file* で使用するフォーマット (構文) を表示します。設定ファイルには **-F** オプションで表示される構文を使用し、必須の構文パラメータが含まれている必要があります。
- **-S *primary_ds* –** サブスクリプションのプライマリデータが格納されているデータサーバの名前です。
- **-D *primary_db* –** サブスクリプションのプライマリデータが格納されているデータベースの名前です。
- **-s *replicate_ds* –** データのレプリケートコピーが格納されているデータサーバの名前です。
- **-d *replicate_db* –** データのレプリケートコピーが格納されているデータベースの名前です。
- **-t *table_name* –** データを比較する、プライマリデータベースとレプリケートデータベースのテーブルの名前です。データベース間でテーブル名が違う場合

には、**-T** オプションでプライマリデータベースのテーブル名を指定します。テーブル所有者名情報を入れることができます。

- **-T *primary_table_name* –** プライマリデータベースのテーブル名です。このオプションは、プライマリデータベースとレプリケートデータベースでテーブル名が違う場合に使用します。テーブル所有者名情報を入れることができます。
- **-c select_command –** データのプライマリコピーとレプリケートコピーの両方から、サブスクリプションのデータを検索する **select** コマンドです。 プライマリデータには別のコマンドを指定する場合は、**-C** を使用します。 **select** コマンドは、プライマリキーに基づいてローの順序を定める必要があります。

  次の条件を満たせば、`text`、`unitext`、または `image` データ型のカラムを **select** コマンドに含めることができます。

  - `text`、`unitext`、または `image` データ型のカラムは、プライマリキーには使用できません。
  - `text`、`unitext`、または `image` データ型のカラムは **select** リストの最後に指定しなければなりません。
  - デフォルトでは、レプリケートテーブルの `text` または `image` カラムには null 値を格納できません。 レプリケートテーブルの `text`、`unitext`、または `image` カラムに null 値を使用できるようにするには、**rs_subcmp** 実行プログラムで **-N** フラグを指定する必要があります。
- **-C *primary_select_command* –** データのプライマリコピーからサブスクリプションデータを検索する **select** コマンドです。 プライマリデータベースとレプリケートデータベースに異なる **select** コマンドを使用する必要がある場合は、このオプションと **-c** を使用します。 **select** コマンドは、プライマリキーに基づいてローの順序を定める必要があります。
- **-u *user* –** プライマリデータサーバとレプリケートデータサーバへのログインに使用するログイン名です。別々のログイン名が必要な場合には、**-U** オプションでプライマリデータサーバのログイン名を指定します。
- **-U *primary_user* –** プライマリデータサーバへのログインに使用するログイン名です。プライマリデータサーバとレプリケートデータサーバで違うログイン名が必要な場合には、このオプションと **-u** オプションを使用してください。
- **-p *passwd* –** *user* ログイン名と *primary_user* ログイン名 (指定されている場合) で使用するパスワードです。このオプションを省略すると、**rs_subcmp** は null パスワードを使用します。*primary_user* ログイン名に異なるパスワードを指定する場合には、**-P** オプションを使用してください。
- **-P *primary_passwd* –** *primary_user* ログイン名で使用するパスワードです。
- **-B *primary_init_batch* –** プライマリデータベースに、初めて接続したときに実行されるコマンドバッチです。バッチは、独立性レベルの設定など、さまざまな

目的で使用できます。指定したバッチは、**rs_subcmp** がプライマリデータベースにログインした後に実行されます。

- **-b *replicate_init_batch* –** レプリケートデータベースに、初めて接続したときに実行されるコマンドバッチです。バッチは、ウォームスタンバイアプリケーションで **rs_subcmp** を実行しているときにトリガを無効にしたり、または独立性レベルを設定したりするなどの、さまざまな目的で使用できます。指定したバッチは、**rs_subcmp** がレプリケートデータベースにログインした後に実行されます。
- **-n *num_iterations* – rs_subcmp** が検出した不整合なローを検証する回数です。デフォルトは 10 回です。1 回目に不整合なローが大量に検出されることがありますが、これは複写時の正常な遅延時間により発生するものです。検証を繰り返すことで、**rs_subcmp** は、正常な複写処理が進んでも修正されない、実際に不整合なローを見つけ出します。
- **-w *wait_interval* – rs_subcmp** が次の検証処理を開始するまでの待機時間です。デフォルトは 5 秒です。
- **-e *float_precision* –** 浮動小数点値を表すとき、指数部で表現できる小数点以下の桁数を設定します。デフォルトでは、プラットフォームによってサポートされている最大値に設定されます。
- **-E *real_precision* –** 実数値を表すとき、指数部で表現できる小数点以下の桁数を設定します。デフォルトでは、プラットフォームによってサポートされている最大値に設定されます。
- **-k *primary_key_column* –** テーブルのプライマリキーを構成するカラム名です。プライマリキーはユニークでなければならず、`text`、`unitext`、または `image` カラムは使用できません。 プライマリキーを構成するカラムそれぞれに **-k** オプションを指定してください。 プライマリカラムとレプリケートカラムで名前が違う場合には、レプリケートカラムの名前を指定します。
- **-i *identity_column* –** レプリケートテーブルの *xidentity* カラムの名前です。
- **-l *text_image_column_name* –** レプリケート側の `text`、`unitext`、または `image` カラムに対する更新がログに記録されないようにします。 デフォルトでは、`text`、`unitext`、または `image` カラムへの更新はログに記録されます。
- **-L *text_image_length* –** `text`、`unitext`、または `image` カラムにデータサーバが返す最も長い値を設定します。 デフォルト値は 2048K です。
- **-N *text_image_column_name* –** レプリケートテーブルの `text`、`unitext`、または `image` カラムに null 値を指定できることを示します。 デフォルトでは、レプリケートテーブルの `text`、`unitext`、または `image` カラムには null 値を格納できません。

- **-Z *language* – rs_subcmp** がエラーおよび情報メッセージに使用する言語名です。このオプションを指定しないと、使用しているプラットフォームの “default” ロケールエントリに指定された言語が使用されます。
- **-o *sort_order* –** 複写システムで使用されているソート順の名前です。**rs_subcmp** は、プライマリキーの各カラムを比較するときにこの情報を使用します。
- **-O *sort_order* –** 複写システムで使用されているソート順の名前です。**rs_subcmp** は、すべてのカラムを比較するときにこの情報を使用します。
- **-J rs_subcmp_charset – rs_subcmp** のエラー、情報メッセージ、すべての設定パラメータまたはコマンドラインオプションで使用する文字セットの名前です。*rs_subcmp_charset* を指定しない場合、使用しているプラットフォーム用の “default” ロケールエントリに指定された文字セットが使用されます。
- **-j rep_charset –** レプリケートデータサーバが使用する文字セットの名前です。**rs_subcmp** は、テーブルのレプリケートバージョンとプライマリバージョンを比較または調整するときにこの文字セットを使用します。*rep_charset* を指定しないと、*rs_subcmp_charset* で指定した文字セットが適用されます。
- **-a *replicate_column_nameprimary_column_name* –** レプリケートカラムの名前と対応するプライマリカラムの名前を指定します。このオプションは、レプリケートカラムの名前がプライマリカラムの名前と違う場合に使用します。

  注意： **-a** オプションでは、レプリケートカラムの名前を対応するプライマリカラムより先に指定してください。
- **-q *unicode_sort_order* – rs_subcmp** が Unicode のプライマリキーカラムの比較に使用する Unicode ソート順を指定します。
- **-Q *unicode_sort_order* – rs_subcmp** がすべての Unicode カラムの比較に使用する Unicode ソート順を指定します。
- **-x *schema_flag* – rs_subcmp** 比較タイプを指定します。*schema_flag* に使用できる値は次のとおりです。
  - 0 - データ比較。これはデフォルトの値。
  - 1 - 2 つのデータベース間のデータベーススキーマ比較。
  - 2 - 2 つのテーブル間のテーブルスキーマ比較。
- **-X *filter* –** 比較に含めるまたは除外するスキーマタイプとサブタイプを指定します。値が “+” で始まる場合、比較のためにスキーマタイプだけが選択され、サブスキーマタイプは無視されます。それ以外の場合は、スキーマタイプとサブスキーマタイプはいずれも選択されず、比較に使用されません。rs_subcmp でサポートされているスキーマタイプとスキーマサブタイプのリストについては、「rs_subcmp でサポートされているスキーマタイプ」テーブルおよび「rs_subcmp でサポートされているスキーマサブタイプ」テーブルを参照してください。

- **-I *interface_file* –** interfaces ファイルのロケーションを指定します。interfaces ファイルの詳細については、使用しているプラットフォームの『Replication Server 設定ガイド』を参照してください。
- **-g –** 矛盾するデータの調整ファイルを作成します。
- **-h –** 高速比較を実行します。
- **-H *normalization_option* –** 高速比較の実行時にデータを正規化する方法を示します。**rs_subcmp** でサポートされている正規化オプションのリストについては、「rs_subcmp でサポートされている正規化オプション」テーブルを参照してください。

例

- **例 1 –** `titleauthor.cfg` という名前の設定ファイルを使用して、**rs_subcmp** を起動します。

```
rs_subcmp -ftitleauthor.cfg
```

この設定ファイルの内容は、次のとおりです。

```
# titleauthor.cfg - Reconcile
# SYDNEY_DS.pubs2.dbo.titleauthor with
# TOKYO_DS.pubs2.dbo.titleauthor.
#
PDS= TOKYO_DS
RDS= SYDNEY_DS
PDB= pubs2
RDB= pubs2
PTABLE= titleauthor
RTABLE= titleauthor
PSELECT= select au_id, title_id, au_ord,¥ royaltyper
from titleauthor order by au_id,¥ title_id
RSELECT= select au_id, title_id, au_ord,¥ royaltyper
from titleauthor order by au_id,¥ title_id
PUSER=repuser
RUSER=repuser
PPWD=piglet
RPWD=piglet
KEY=au_id
KEY=title_id
RECONCILE=Y
VISUAL=Y
NUM_TRIES=3
WAIT=10
```

rs_subcmp は、`titleauthor` という名前のプライマリテーブルとレプリケートテーブルを比較し、次のような出力を生成します。

```
$SYBASE/bin/rs_subcmp -f ttl_au.cmp
INCONSISTENT ROWS:

_________Replicate row________
```

```
au_idtitle_idau_ordroyaltyper
-----------------------------------------
672-71-3249TC77771 40

__________Primary row_________
au_idtitle_idau_ordroyaltyper
-----------------------------------------
672-71-3249TC77771 50
```

- **例 2 –** subcmp.cfg という名前の設定ファイルを使用して、**rs_subcmp** を起動します。 コマンドラインオプションは、設定ファイルの設定を上書きします。 これは、authors テーブルの最終的な検証を行って、プライマリバージョンとレプリケートバージョンの差異を調整します。

```
rs_subcmp -R -fsubcmp.cfg -STOKYO_DS -Dpubs2 ¥
 -sSYDNEY_DS -dpubs2 -tauthors
```

プライマリのデータサーバとデータベースは TOKYO_DS と pubs2 です。 レプリケートのデータサーバとデータベースは SYDNEY_DS と pubs2 です。

- **例 3 –** PASE および R2ASE という 2 つの異なるサーバにある authors テーブルのスキーマを比較します。この場合、pubs2 というデータベースを持つこれらの各サーバは config.cfg ファイルという設定ファイルを使用します。

```
rs_subcmp -f config.cfg
```

この設定ファイルの内容は、次のとおりです。

```
PDS = PASE
RDS = R2ASE
PDB = pubs2
PTABLE = authors
RTABLE = authors
PUSER = sa
RUSER = sa
PPWD =
RPWD =
SCHEMAFLAG = 1
```

- **例 4 –** 設定ファイルを使用しないで、2 つのデータベース間のスキーマを比較します。

```
rs_subcmp -Spds -srds -Dpdb -drdb -Usa -Psa_pwd -usa
          -psa_pwd -x1
```

- **例 5 –** インデックス、トリガ、データ型を除外して、2 つのデータベースのスキーマを比較します。

```
rs_subcmp -Spds -srds -Dpdb -drdb -Usa -Psa_pwd -usa
          -psa_pwd -x1 -XitD
```

- **例 6 –** すべてのテーブルスキーマとユーザスキーマを比較します。

```
rs_subcmp -Spds -srds -Dpdb -drdb -Usa -Psa_pwd -usa
          -psa_pwd -x1 -X+TU
```

使用法

- プライマリで変更が発生しない状態で、**rs_subcmp** を実行します。
- **rs_subcmp** の環境変数を正しく機能するように設定します。
  - SYBASE - リリースディレクトリに設定されます。
  - SYBASE_JRE6 - 次のように、32 ビットまたは 64 ビットの JRE のある場所に設定されます。

表 51 : プラットフォームと SYBASE_JRE6 変数の場所

プラットフォーム	SYBASE_JRE6 変数の場所
32 ビット UNIX	/$SYBASE(ASE)/shared/JRE-6_0_24_32BIT
64 ビット UNIX	/$SYBASE(ASE)/shared/JRE-6_0_24_64BIT
32 ビット Windows	¥%SYBASE%¥shared¥JRE-6_0_24_32BIT
64 ビット Windows	¥%SYBASE%¥shared¥JRE-6_0_24_64BIT

  - ライブラリパス変数 - `$SYBASE/$SYBASE_OCS/lib` (UNIX) または `%SYBASE%¥%SYBASE_OCS%¥lib` (Windows) に設定されます。

    サポートされるオペレーティングシステムごとに、次のように異なるライブラリパス変数が使用される場合があります。

表 52 : オペレーティングシステムとライブラリパス変数

オペレーティングシステム	
RS6000	LIBPATH
HP UX	SHLIB_PATH
Linux	LD_LIBRARY_PATH
Solaris	LD_LIBRARY_PATH
Windows	PATH

- **rs_subcmp** スキーマ比較を機能させるには、DDLGENLOC および SYBROOT 環境変数を設定します。

  スキーマ比較を行うには、**rs_subcmp** を使用して `ddlgen` 実行ファイルを検索し、正常に実行できる必要があります。 *DDLGENLOC* が設定されていない場合、**rs_subcmp** はそのデフォルトロケーションで `ddlgen` を検索します (デフォルトロケーションは `%SYBASE%¥ASEP¥bin¥ddlgen`)。 `ddlgen` を正常に実行するには、`ddlgen` が使用する環境変数を正しく設定する必要があります。

注意： ddlgen がデフォルトロケーションとして設定されていない場合は、*DDLGENLOC* を %SYBASE%¥ASEP¥bin¥ddlgen などのフルパスに設定する必要があります。

SYBROOT 環境変数を SYBASE 環境変数に設定します。

- 次の動作条件が **rs_subcmp** に適用されます。
  - 設定ファイルがあり、コマンドラインオプションも使用する場合、コマンドライン値は設定ファイルにある値を上書きします。
  - 小文字のオプション (**-d**、**-c**、**-u**、**-p**、**-t**) に指定した値は、プライマリデータとレプリケートデータの両方に適用されます。プライマリデータに対する値を上書きするには、大文字のオプションを使用してください。
  - 大文字のオプションのうち、**-S** は必須オプションです。
  - **-k** で指定するプライマリキーは、ユニークでなければなりません。**-k** オプションを使用したプライマリキーカラムを指定しない場合、すべてのカラムがプライマリキーとして扱われます。
  - **-L** の正の整数を使用して、text カラムと image カラムのバイト長に 26KB のデフォルト値を上書きする新しい値を指定します。

    ```
    -L = <new_value>
    ```

    たとえば、text カラムと image カラムを 65,536 バイトにするには、次のように入力します。

    ```
    -L = <64>
    ```
- 次のオプションは、デフォルト以外のテーブルの所有者や、プライマリ側とレプリケート側で別々のテーブル名やカラム名を指定する場合に使用できます。
  - **-t**、**-T**、**-c**、**-C** の各オプションでは、テーブルの所有者も指定できます (**ling.authors** など)。
  - **-c** オプションでは、レプリケートテーブルの所有者、テーブル名、カラム名を指定します。
  - **-C** オプションでは、プライマリテーブルの所有者、テーブル名、カラム名を指定します。
  - **-k** オプションでは、レプリケートテーブルのカラム名を指定します。
- **rs_subcmp** は、スキーマ比較を実行するたびにレポートファイルを作成します。このレポートファイルは、2 つのテーブル間またはデータベース間の比較結果の詳細を示します。レポートファイルには、reportPROCID.txt という名前が付けられます。矛盾がある場合、**rs_subcmp** は reconcilePROCID.sql という名前の調整スクリプトを作成します。レポートファイルと調整スクリプトは、**rs_subcmp** が実行されるディレクトリと同じディレクトリに保存されます。
- 調整ファイルの SQL 文に、*text*、*unitext*、または *image* を含めることはできません。

- **-g** オプションを指定した場合、**rs_subcmp** は調整ファイルを作成します。ファイルには `reconcile_file_PROCID.sql` という名前が付けられ、現在の作業ディレクトリに置かれます。

リターンコード

rs_subcmp によって返されるリターンコードは、次のとおりです。

表 53 : rs_subcmp のリターンコード

リターンコード	意味
0	複写テーブルとプライマリテーブルが同じである。
1	**rs_subcmp** の実行中にエラーが発生した。
2	複写テーブルとプライマリテーブルが異なる。

設定ファイル

rs_subcmp パラメータを記載したファイルを作成し、そのファイル名をコマンドラインの **-f** オプションで指定できます。設定ファイルの中の各行は、パラメータ名、等号 (=)、値から構成されます。

表 54 : rs_subcmp の設定ファイルパラメータ (758 ページ) は、**rs_subcmp** 設定ファイルで使用できるパラメータと各パラメータに対応するコマンドラインオプションを表示します。

表 54 : rs_subcmp の設定ファイルパラメータ

設定パラメータ	コマンドラインオプション	値
PDS	**-S**	プライマリデータサーバの名前。
RDS	**-s**	レプリケートデータサーバの名前。
PDB	**-D**	プライマリデータベースの名前。
RDB	**-d**	レプリケートデータベースの名前。
PTABLE	**-T**	プライマリテーブルの名前。
RTABLE	**-t**	レプリケートテーブルの名前。
PUSER	**-U**	プライマリユーザ名。
RUSER	**-u**	レプリケートユーザ名。
PPWD	**-P**	プライマリパスワード。

設定パラメータ	コマンドラインオプション	値
RPWD	**-p**	レプリケートパスワード。
KEY	**-k**	レプリケートテーブルでのプライマリキー要素。
PINITBATCH	**-B**	プライマリデータベースコネクションの初期化バッチ。改行文字の前に円記号 (“¥”) を指定すれば、次の行に続けることができる。1 行は 1024 文字まで、合計で 64K 文字が使用できる。
RINITBATCH	**-b**	レプリケートデータベースコネクションの初期化バッチ。改行文字の前に円記号 (“¥”) を指定すれば、次の行に続けることができる。1 行は 1024 文字まで、合計で 64K 文字が使用できる。
PSELECT	**-C**	プライマリ **select** コマンド。改行文字の前に円記号 (“¥”) を指定すれば、次の行に続けることができる。1 行は 1024 文字まで、合計で 64K 文字が使用できる。
RSELECT	**-c**	レプリケート **select** コマンド。改行文字の前に円記号 (“¥”) を指定すれば、次の行に続けることができる。1 行は 1024 文字まで、合計で 64K 文字が使用できる。
RECONCILE	**-r**	不整合を調整する (Y または N)。
RECONCILE_CHECK	**-R**	プライマリで検証後に不整合を調整する (Y または N)。
TRACE	**-z**	レベルを指定してトレースを有効にする (1 または 2)。
FPRECISION	**-e**	浮動小数点に求められる精度 (整数 - デフォルト値はプラットフォームの設定)。
RPRECISION	**-E**	実数に求められる精度 (整数 - デフォルト値はプラットフォームの設定)。
WAIT	**-w**	次の比較処理までの秒数 (整数 - デフォルトは 5 秒)。
NUM_TRIES	**-n**	比較処理の回数 (整数 - デフォルトは 10 回)。
VISUAL	**-V**	結果を出力 (Y または N)。
IDENTITY	**-i**	レプリケートテーブルの *identity* カラム名。

設定パラメータ	コマンドラインオプション	値
TXT_IMG_LEN	**-L**	*text*、*unitext*、または *image* カラムにデータサーバが返す最も長い値 (キロバイト単位)。
NO_LOG	**-l**	レプリケート側の *text*、*unitext*、または *image* カラムへの更新をログに記録しない。
NULLABLE	**-N**	レプリケートテーブルの *text*、*unitext*、または *image* カラムに null 値を指定できる。
LANGUAGE	**-Z**	**rs_subcmp** のエラーおよび情報メッセージの言語。
SORT_ORDER	**-o**	指定したソート順を使用して、プライマリキーカラムを比較する。
SORT_ORDER_ALL_COLS	**-O**	指定したソート順を使用して、すべてのカラムを比較する。
SCHARSET	**-j**	**rs_subcmp** の文字セット。
RCHARSET	**-J**	レプリケートデータサーバの文字セット。
REP_PRI_COLNAME	**-a**	レプリケートカラムとプライマリカラムの名前。
UNICODE_SORT_ORDER	**-q**	**rs_subcmp** が Unicode のプライマリキーカラムの比較に使用する Unicode ソート順。
UNICODE_SORT_ORDER_ALL_COLS	**-Q**	**rs_subcmp** がすべての Unicode カラムの比較に使用する Unicode ソート順。
SCHEMAFLAG	**-x**	**rs_subcmp** 比較タイプ。
FILTER	**-X**	スキーマ比較に含めるまたは除外するスキーマタイプとスキーマサブタイプを指定するために使用するフィルタ。**rs_subcmp** でサポートされているスキーマタイプとスキーマサブタイプのリストについては、表 55 : rs_subcmp でサポートされているスキーマタイプ (766 ページ)および表 56 : rs_subcmp でサポートされているスキーマサブタイプ (766 ページ) を参照してください。
IFILE	**-I**	interfaces ファイルのロケーション。

設定パラメータ	コマンドラインオプション	値
RECONCILE_FILE	**-g**	調整ファイルを作成するかどうかを示す。 値： • Y - 調整ファイルを作成する。 • N - 調整ファイルを作成しない。 デフォルト値は N
FASTCMP	-h	高速比較を実行するかどうかを示す。 値： • Y - 圧縮されたデータを使用して高速比較を実行する。 • N - 通常比較を実行する。 デフォルト値は N
HASH_OPTION	**-H**	高速比較に使用する正規化オプションを示す。このパラメータが設定ファイルに含まれていない場合、**rs_subcmp** はネイティブのバイト順序と文字セットを使用してデータを正規化する。**rs_subcmp** でサポートされている正規化オプションのリストについては、「rs_subcmp でサポートされている正規化オプション」テーブルを参照してください。

select コマンドの動作条件

- **-c** (RSELECT) および **-C** (PSELECT) によって指定された **select** コマンドは、プライマリデータベースとレプリケートデータベースの両方から名前とデータ型が同じカラムを返す必要があります。
- **select** コマンドのプライマリキーまたは **order by** 句には、クラスタードインデックスが必要です。**select** コマンドでは、プライマリキーを基準にしてローを並べ替えてください。**rs_subcmp** は正しい順序でローを受信しないと、レプリケートテーブルのローが削除されることがあります。
- *rs_address* データ型を **-c** または **-C** オプションでは選択しないでください。レプリケートテーブルに *rs_address* データ型のカラムが含まれている場合、これらのカラムはプライマリ側とレプリケート側で一致しないことがあります。Replication Server では、不必要な複写を避けるためにこれらのカラムへの更新が除外されるためです。

rs_subcmp の動作

- **rs_subcmp** は、プライマリデータベースとレプリケートデータベースにログインし、指定された **select** コマンドを実行します。返されたカラムの名前とデータ型から、それらが同じカラムかどうかが判断されます。同じであれば、**rs_subcmp** はプライマリローとレプリケートローを比較し、次の 3 種類のリストを作成します。
  - 脱落したロー - プライマリにあってレプリケートにはないロー。
  - 孤立したロー - レプリケートにあってプライマリにはないロー。
  - 一貫性のないロー - プライマリキーが一致するローがレプリケートとプライマリにあるが、他のカラムが一致しない。
- 3 つのリストを作成すると、**rs_subcmp** は指定された回数だけ比較を繰り返し、次の内容をチェックします。
  - 脱落したローがレプリケートに出現しないか
  - 孤立したローがレプリケートからなくならないか
  - 一貫性のないローが一致しないか
  - 新たに検出されたレプリケートローの値が、前回の処理でのプライマリローの値と一致していないか
- 指定された回数の処理を繰り返した後、**-V** オプションを指定している場合には、3 つのリストの内容が標準出力に出力されます。

不整合の調整

- **rs_subcmp** は、**-R** または **-r** オプションを指定すると、脱落したロー、孤立したロー、一貫性のないローを調整します。
- **-r** オプションを指定した場合、**rs_subcmp** はプライマリコピーとレプリケートコピーを調整します。具体的には、最終的なリストをもとに、次の方法でレプリケートテーブルを変更します。
  - 脱落したローのリストに残っているローを挿入する。
  - 孤立したローのリストに残っているローを削除する。
  - 一貫性のないローは、プライマリローに合わせて更新する。
- **-R** オプションを指定した場合、**rs_subcmp** は **-r** オプションと同じ方法で、レプリケートテーブルをプライマリテーブルに合わせて調整します。ただし、脱落したローを挿入したり孤立したローを削除したりする前に、プライマリデータベースにログインし、対象となるローに **select** コマンドを実行して、次の点を確認します。
  - ローがまだ存在するかどうか (レプリケートテーブルに脱落したローがある場合)
  - ローが本当に存在しないかどうか (レプリケートテーブルに孤立したローがある場合)

IDENTITY カラムの調整

- あるローの *identity* カラムの値が一致しない場合、**rs_subcmp** はレプリケートデータベースでローを削除してから、プライマリデータベースからローを挿入してこれらを整理統合します。

text、unitext、または image データ型の調整

- 他のデータ型と違って、*text*、*unitext*、または *image* の値の不整合はリストに記録されません。*text* または *image* の値を含む脱落したローまたは一貫性のないローを調整する場合、**rs_subcmp** はプライマリデータベースにもう一度ログインして **select** 文を再度実行します。一貫性のないローまたは脱落したローが見つかった場合、**rs_subcmp** はローを更新または挿入してレプリケートテーブルを変更します。プライマリテーブルで一貫性のないローまたは脱落したローが見つからなかった場合、**rs_subcmp** は次の処理を実行します。
  - 一貫性のないローに対して、**rs_subcmp** はレプリケートテーブルから該当するローを削除する。
  - 脱落したローに対して、**rs_subcmp** は何の処理も行わない。
- Adaptive Server のオプション **set textsize** を **select** 文の中で使用すると、比較するテキストの量を制限できます。たとえば、テキストサイズの設定を 10 に変えた場合の、出力結果の違いを次に示します。最初の **select** 文では、30 文字の *text* が返されます。

```
set textsize 30 select * from zetext

abc
-----------------------------------------------
abba apples odd one here
beta banana rotten
caro celery not carrots
```

次の select 文では、テキストの *size* を 10 に設定します。

```
1> set textsize 10 select * from zetext
2> go
abc
```

```
---------------------------
abbaapplesodd one here
betabananarotten
carocelerynot carrot

(3 rows affected)
```

国際的な環境での rs_subcmp の使用

- **rs_subcmp** は、国際的な環境をサポートするために、 **-Z***language*、**-o***sort_order*、**-O***sort_order*、**-q***unicode_sort_order*、 **-Q***unicode_sort_order*、**-J***rs_subcmp_charset*、**-j***rep_charset* の各オプションを提供しています。

- **rs_subcmp** は、テーブルのレプリケートバージョンとプライマリバージョンを比較または調整するときに文字セットを変換します。この変換方法は Replication Server での変換方法と似ているため、同じような結果が得られます。たとえば、プライマリデータサーバとレプリケートデータサーバの文字セットに互換性がない場合、変換はできません。文字セットに互換性があっても、プライマリデータサーバの特定の文字がレプリケートサーバの文字セットにない場合には、その文字は "?" に置き換えられ、処理は継続されます。
- **rs_subcmp** は、ユーザデータに関連するすべてのオペレーションに対して、レプリケートデータサーバの文字セットを使用します。レプリケートデータサーバの文字セットを指定するには、**-j** コマンドラインオプションまたは *RCHARSET* 設定ファイルパラメータを使用してください。

  注意： すべてのデータオペレーションはレプリケートデータサーバの文字セットで実行されるため、**rs_subcmp** にはプライマリデータサーバの文字セットを指定するパラメータはありません。プライマリデータサーバで、すべての文字データがレプリケートデータサーバの文字セットに変換されます。これは、サブスクリプションマテリアライゼーション中の Replication Server の動作に関連します。
- **rs_subcmp** にレプリケートデータサーバと別の文字セットを指定することもできます。その場合、**-J** コマンドラインオプションまたは *SCHARSET* 設定ファイルパラメータを使用します。文字セットを指定すると、**rs_subcmp** は、文字列型の設定パラメータを **rs_subcmp** の文字セットからレプリケートデータサーバの文字セットに変換します。

文字セットとソート順の動作条件

- **rs_subcmp** で文字セットとソート順を指定する場合、次の条件が適用されます。
  - オブジェクト名 (サーバ名、データベース名、テーブル名、カラム名など) のすべての文字に、*rs_subcmp_charset* および *rep_charset* 文字セットと互換性がなければならない。互換性がないと **rs_subcmp** は実行できない。
  - レプリケートデータサーバとプライマリデータサーバの文字セットが異なる場合、レプリケートデータサーバの文字セットがプライマリデータサーバにインストールされている必要がある。これにより、プライマリデータサーバが文字セットを変換できる。
  - レプリケートデータサーバとプライマリデータサーバのソート順が異なり、**select** 文の **where** 句に *character* または *text* のデータ型が含まれている場合、結果に混乱が生じることがあります。 これを防ぐには、あらかじめ **-r** または **-R** (調整) オプションは指定せず、**-V** (出力) オプションを指定して **rs_subcmp** を実行し、データへの影響を調査します。

ソート順の使用

- Unicode 以外のソート順は、**-o** オプションまたは **-O** オプションを使用して指定します。
- **-o** オプションを指定すると、**rs_subcmp** は次の処理を実行します。
  1. プライマリキーのカラムのバイナリ比較を実行します。
  2. プライマリキーが一致する場合、**rs_subcmp** は残りのカラムのバイナリ比較を実行します。プライマリキーが一致しない場合、矛盾するローがレポートされます。
  3. プライマリキーのカラムが一致しない場合、**rs_subcmp** は指定したソート順を使ってそれらのカラムを比較します。
     - プライマリキーが一致しない場合、ローは脱落または孤立としてレポートされる。
     - プライマリキーのカラムがソート順を使用して同じようにテストする場合、ローは矛盾しているとレポートされる。
- **-O** オプションを指定すると、**rs_subcmp** は次の処理を実行します。
  - 指定されたソート順を使用して、*char*、*varchar*、*text* データ型のすべてのカラムに対してカラムの比較を実行する。
  - バイナリ比較は行わない。
- ソート順を指定しないと、**rs_subcmp** はプライマリローとレプリケートローの各カラムに対してバイナリ比較を実行します。

Unicode ソート順の使用

- Unicode のソート順は、**-q** オプションまたは **-Q** オプションを使用して指定します。
- **-q** オプションを指定すると、**rs_subcmp** は次の処理を実行します。
  1. Unicode のプライマリキーカラムのバイナリ比較を実行します。
  2. プライマリキーが一致する場合、**rs_subcmp** は残りのカラムのバイナリ比較を実行します。プライマリキーが一致しない場合、矛盾するローがレポートされます。
  3. プライマリキーのカラムが一致しない場合、**rs_subcmp** は指定したソート順を使ってそれらのカラムを比較します。
     - Unicode のプライマリキーカラムが一致しない場合、ローは脱落または孤立としてレポートされる。
     - プライマリキーのカラムがソート順を使用して同じようにテストする場合、ローは矛盾しているとレポートされる。
- **-Q** オプションを指定すると、**rs_subcmp** は次の処理を実行します。
  - すべての Unicode カラムに対して、指定されたソート順を使用してカラムの比較を実行する。
  - バイナリ比較は行わない。

- ソート順を指定しないと、**rs_subcmp** はプライマリローとレプリケートローの各 Unicode カラムに対してバイナリ比較を実行します。

表 55 : rs_subcmp でサポートされているスキーマタイプ

タイプ	説明
A	データベース内のすべてのエイリアス。
D	データベース内のすべてのデフォルト値。
E	データベース内のすべてのユーザ定義データ型。
G	データベース内のすべてのグループ。
R	データベース内のすべてのルール。
T	データベース内のすべてのユーザテーブル。インデックス、キー、制約、トリガなどのテーブル要素を含む。
U	データベース内のすべてのユーザ。
V	データベース内のすべてのビュー。
P	データベース内のすべてのプロシージャ。

表 56 : rs_subcmp でサポートされているスキーマサブタイプ

タイプ	説明
c	制約
d	デフォルトのバインド
f	外部キー
g	付与
i	インデックス
m	プロシージャモード
p	プライマリキー
r	バインドルール
t	トリガ

表 57 : rs_subcmp でサポートされている正規化オプション

正規化オプション	説明
lsb	バイト順序に依存するすべてのデータを LSB-First (リトルエンディアン) のバイト順序に正規化する。
msb	バイト順序に依存するすべてのデータを MSB-First (ビッグエンディアン) のバイト順序に正規化する。
unicode	文字データを Unicode (UTF-16) に正規化する。
unicode_lsb	プラットフォームに依存しないようにするために、Unicode を使用して lsb を正規化する。
unicode_msb	プラットフォームに依存しないようにするために、Unicode を使用して msb を正規化する。

SAP Replication Server システムテーブル

SAP Replication Server システムデータベース (RSSD) または Embedded RSSD (ERSSD) に格納されているシステムテーブルについて説明します。 システムテーブルは、専用のデータベース (RSSD 用の Adaptive Server または SQL Anywhere ERSSD) に格納されます。

システムテーブルにアクセスできるのは、**sa** パーミッションを持つユーザ、または *rs_systabgroup* グループのメンバだけです。 システムテーブルは RCL コマンドで管理します。直接には修正しないでください。 *rs_config* テーブル内のサーバ値を変更するには、**configure replication server** コマンドを使用します。

rs_systabgroup グループの詳細については、「repserver」を参照してください。 **configure replication server** の詳細については、**「configure replication server」**を参照してください。

システムテーブルには、*binary(8)* として定義されたユーザ定義のデータ型 *rs_id* が含まれています。このデータ型は、オブジェクト名を格納するカラムで使用されます。識別子 (オブジェクト名) の詳細については、「識別子」を参照してください。

システムテーブル *rs_lastcommit* と *rs_threads* は、RSSD や ERSSD ではなく、各ユーザデータベースで作成、保存されますが、この 2 つのテーブルについても、この章で説明します。

rs_articles

この Replication Server で認識されているアーティクルについての情報を格納します。

カラム	データ型	説明
articlename	varchar(255)	アーティクルの名前。
articleid	rs_id	ユニークなアーティクル ID。
type	char(1)	• T - テーブル • P - プロシージャ
primaryname	varchar(255)	プライマリテーブルまたはプロシージャの名前。

カラム	データ型	説明
primaryowner	varchar(30)	プライマリテーブルの所有者名。
objid	rs_id	対応する複写定義の ID。
pubid	rs_id	このアーティクルが属するパブリケーションの ID。
requestdate	datetime	アーティクルがパブリケーションに追加された日時。
minvers	int	このアーティクルをサポートするのに必要な Replication Server の最小バージョン。

インデックス

- articlename、pubid にユニーククラスタードインデックス
- articleid にユニークインデックス
- objid にインデックス

rs_asyncfuncs

Replication Server の複写定義に対するユーザ定義関数についての情報を格納します。同じ情報が rs_objfunctions にも格納されます。

カラム	データ型	説明
prsid	int	このファンクションがプライマリであるサイト。
funcname	varchar(255)	ファンクション名
funcid	rs_id	ファンクション ID
objid	rs_id	ファンクションが適用されるオブジェクト。
conflicting	tinyint	ファンクションが矛盾している場合は 1、そうでない場合は 0。
userdefined	bit	ユーザ定義のファンクションの場合は 1、そうでない場合は 0。
rowtype	tinyint	ローが複写されている場合は 1、そうでない場合は 0。

インデックス

- funcname にクラスタードインデックス
- objid、funcname にユニークインデックス
- funcid にユニークインデックス

rs_autopartpath

Replication Server がステーブルメッセージキュー用に使用する自動でサイズ変更可能なパーティションについての情報を格納します。

カラム	データ型	説明
name	varchar(30)	自動でサイズ変更可能なパーティションの論理パーティションパスに割り当てる名前。
path	varchar(255)	自動でサイズ変更可能なパーティション内のパーティションファイルのオペレーティングシステム上の物理的パスロケーション。
size	int	Replication Server が自動でサイズ変更可能なパーティションに作成できる各パーティションファイルに対して設定するメガバイト単位のサイズ。
max_size	int	Replication Server が自動でサイズ変更可能なパーティションに対して作成できるすべてのパーティションファイルに割り当てるメガバイト単位の合計サイズ。
status	int	自動でサイズ変更可能なパーティションのステータス。 • 0x00 - オンライン • 0x01 - 削除保留中 • 0x02 - パーティションあり • 0x04 - フル

インデックス

- name に、ユニーククラスタードインデックス
- path にユニークインデックス

rs_classes

ファンクション文字列クラス名とエラークラス名を格納します。

カラム	データ型	説明
classname	varchar(30)	クラス名。
classid	rs_id	このクラスの ID。
classtype	char(1)	値は次のいずれかになる。 • R Replication Server エラークラス • F - ファンクション文字列クラス • E - データサーバのエラークラス • D - データ型クラス
prsid	int	このクラスがプライマリであるサイトの ID。
parent_classid	rs_id	これが派生クラスである場合、親クラスの ID。 これが基本クラスである場合、0 (デフォルト)。
attributes	int	0x01 - デフォルトクラス。 *rs_default_function_class* と *rs_db2_function_class* の場合、デフォルトは 1。その他の場合、デフォルトは 0。

インデックス

- classname、classtype にユニーククラスタードインデックス
- classid にユニークインデックス

rs_clsfunctions

クラス全体の関数についての情報を格納します。

カラム	データ型	説明
prsid	int	このファンクションがプライマリであるサイト。
funcname	varchar(255)	ファンクションの名前。

カラム	データ型	説明
funcid	rs_id	ファンクションの ID。
objid	rs_id	0x00000000
conflicting	tinyint	ファンクションが矛盾している場合は 1、そうでない場合は 0。
userdefined	bit	ユーザ定義のファンクションの場合は 1、そうでない場合は 0。
rowtype	tinyint	ローが複写されている場合は 1、そうでない場合は 0。

インデックス

- funcname にユニークインデックス
- funcid にユニークインデックス

rs_columns

複写定義のカラムに関する情報を格納します。

カラム	データ型	説明
prsid	int	このオブジェクトのプライマリ Replication Server。
objid	rs_id	このカラムが属するテーブル複写定義またはファンクション複写定義の ID、もしくはファンクション ID。
colname	varchar(255)	カラム名またはパラメータ名。
colnum	smallint	カラム番号。

カラム	データ型	説明
coltype	tinyint	カラムまたはパラメータのデータ型。 • 0 - char • 1 binary • 4 - text • 5 - image • 6 - tinyint • 7 - smallint • 8 - int • 9 - real • 10 - float • 11 - bit • 12 - datetime • 13 - smalldatetime • 14 - money • 15 - smallmoney • 16 - numeric • 17 - decimal • 18 - varchar • 19 - varbinary • 25 - unichar • 27 - date • 28 - time • 29 - unitext • 30 - bigint • 31 - usmallint • 32 - uint • 33 - ubigint • 35 - bigdatetime • 36 - bigtime • 110 - univarchar
length	int	宣言されたデータの長さ。
searchable	tinyint	サーチャブルキーの場合は 1、それ以外の場合は 0。

カラム	データ型	説明
primary_col	tinyint	プライマリキーの場合は 1、そうでない場合は 0。
fragmentation	tinyint	フラグメンテーションキーの場合は 1、それ以外の場合は 0。
rowtype	tinyint	ローが複写される場合は 1、そうでない場合は 0。
status	int	マスク。次のうち 1 つ以上。 • 0x01 - カラムは *identity* カラムとして宣言されている。 • 0x02 - カラムは *timestamp* カラムとして宣言されている。 • 0x04 - カラムのデータ型は *rs_address* である。 • 0x08 - カラムのステータスは **replicate_if_changed** に設定されている。 • 0x10 - レプリケートテーブルでこのカラムに null 値を許可する (*text*、*unitext*、または *image* カラムに対してのみ)。 • 0x20 - カラムはスタンバイコネクションに送信される (内部複写定義の場合のみ)。 • 0x40 - カラムは内部複写定義から削除されたとマーク付けされている (内部複写定義の場合のみ)。 • 0x200 - *identity* としてパブリッシュされた。 • 0x400 - *timestamp* としてパブリッシュされた。 • 0x1000 - Java カラムとして宣言された。 • 0x2000 - Java カラムとしてパブリッシュされた。
basecolnum	smallint	基本複写定義でのカラムの位置。デフォルトは *colnum* の値。
repl_colname	char(255)	レプリケートテーブルでのカラム名。デフォルトは *colname* の値。

カラム	データ型	説明
declared_dtid	rs_id	データ型 ID。ユーザ定義データ型の場合、これはテーブルへの外部キーとなる。
publ_dtid	rs_id	複写定義に指定されているパブリッシュデータ型。 パブリッシュデータ型が指定されていない場合、publ_dtid の値は declared_dtid と同じ。
publ_base_coltype	tinyint	パブリッシュデータ型の基本データ型。 パブリッシュデータ型が指定されていない場合、publ_base_coltype の値は coltype と同じ。
publ_length	int	パブリッシュデータ型の最大長。
version	rs_id	複写定義バージョンを確認する。
ref_objowner	varchar(30)	参照句で指定され、RI 制約に使用されるレプリケートオブジェクト所有者。 オブジェクト所有者は、RI 制約で指定されるレプリケートデータベースでのテーブル所有者である。
ref_objname	varchar(255)	参照句で指定され、RI 制約に使用されるレプリケートオブジェクト名。 オブジェクト名は、RI 制約で指定されるレプリケートデータベースでのテーブル名である。

インデックス

- version、colname にユニーククラスタードインデックス
- objid、basecolnum にユニークインデックス
- objid、colname にユニークインデックス
- objid、colnum にユニークインデックス
- version、colnum にユニークインデックス

rs_config

configure replication server コマンドを使用して変更できる、一連の設定パラメータのデフォルト値を格納します。特定のターゲット向けの一部のパラメータは、**alter connection**、**alter logical connection**、または **alter route** コマンドを使用して設定することもできます。

rs_config テーブルの設定パラメータの詳細については、『Replication Server 管理ガイド 第 1 巻』を参照してください。

カラム	データ型	説明
optionname	varchar(30)	パラメータの名前(**memory_max**、**cm_max_connections** など) パラメータの一覧とその内容を表示するには、*rs_config* テーブルに対して **select *** 文を実行する。
objid	rs_id	このオプションが参照するオブジェクトの ID。0 の場合はシステム全体に適用される。
charvalue	varchar(255)	パラメータの文字値。
status	tinyint	このカラムは使用されていない。
comments	varchar(255)	パラメータについての説明。

インデックス

optionname、objid にユニーククラスタードインデックス

rs_databases

Replication Server サイトで認識されているデータベースの名前を格納します。

カラム	データ型	説明
dsname	varchar(30)	データサーバ名。
dbname	varchar(30)	データベース名
dbid	int	データベースのユニークな識別子。
conn_id	int	データベース接続のユニークな識別子。 状況に応じて次のようになる。 • デフォルトのコネクション - connid は dbid と同じ値。 • 代替コネクション - connid は dbid とは異なる値。

カラム	データ型	説明
dist_status	cs_int	コネクションのステータス。値は次のとおり。 • 0x1 - 有効。 • 0x2 - サスペンドされている。 • 0x4 - スタンバイに関連するアクションによってサスペンドされている。 • 0x8 - マーカを待機中。 • 0x10 - **dbcc ('ltm', 'ignore')** を発行する。 • 0x20 - スタンバイデータベースを初期化するためにダンプマーカを待機中。 • 0x40 - 関連する重複検出を切り替え中 (*ltype* が 'P' の場合)。 • 0x40 - 切り替えを許可する (*ltype* が 'L' の場合)。 • 0x80 - 一時的に何のグループ化も行っていない。 • 0x100 - 再同期マーカを待機中。 • 0x400 - ユーザデータベースのアップグレードを待機中。
src_status	cs_int	送信元のステータス。 • 0x1 - 有効。 • 0x2 - サスペンドされている。 • 0x4 - スタンバイに関連するアクションによってサスペンドされている。 • 0x10 - DIST スレッドはサスペンドされている。
attributes	tinyint	値は次のいずれかになる。 • 1 - 分配 • 2 - 送信元
errorclassid	rs_id	このデータベースのエラークラス。
funcclassid	rs_id	このデータベースのファンクション文字列クラス。

カラム	データ型	説明
prsid	int	このデータベースを管理する Replication Server の ID。
rowtype	tinyint	ローのタイプを示す。 • 1 - ローは複写される。 • 0 - ローは複写されない。
sorto_status	tinyint	ソート順のチェックが終了しているかどうかを示す。次の値のいずれか。 • 0 - チェックされていない。 • 1 - チェック済み。
ltype	char(1)	このローが表すデータベースのタイプ。値は次のいずれかになる。 • P - 物理データベース • L - 論理データベースコネクション
ptype	char(1)	ウォームスタンバイアプリケーションでのデータベースのタイプ。値は次のいずれかになる。 • A - アクティブデータベース • S - スタンバイデータベース • L - 論理データベースコネクション
ldbid	int	データベースに関連する論理コネクションの *dbid*。論理コネクションがない場合、*ldbid* の値は *dbid* と同じ。
enable_seq	int	アクティブデータベースの切り替え時、またはスタンバイデータベースの作成時に使用されるシーケンス番号。
rs_errorclassid	rs_id	このデータベースの Replication Server エラークラス

インデックス

- dsname、dbname、ltype にユニーククラスタードインデックス
- ptype、ldbid にユニークインデックス

- dbid、ltype にユニークインデックス
- dsname、dbname、ptype にユニークインデックス

rs_datatype

複写定義内のすべてのユーザ定義データ型 (UDD) についての属性情報を格納します。

カラム	データ型	説明
prsid	int	値は次のとおり。 • プライマリ Replication Server の ID。 • グローバルに定義された UDD の場合は 0。
classid	rs_id	データ型が属するデータ型クラスの ID。
dtname	varchar(30)	データ型のユニークな名前。
dtid	rs_id	データ型のユニークな ID。

カラム	データ型	説明
base_coltype	tinyint	データ型の基本データ型の ID。値は次のとおり。 • 0 - char • 1 - binary • 2 - longchar (未使用) • 3 - longbinary (未使用) • 4 - text • 5 - image • 6 - tinyint • 7 - smallint • 8 - int • 9 - real • 10 - float • 11 - bit • 12 - datetime • 13 - smalldatetime • 14 - money • 15 - smallmoney • 16 - numeric • 17 - decimal • 18 - varchar • 19 - varbinary • 21 - sensitivity • 25 - unichar • 27 - date • 28 - time • 29 - unitext

カラム	データ型	説明
		• 30 - bigint • 31 - usmallint • 32 - uint • 33 - ubigint • 35 - bigdatetime • 36 - bigtime • 101 - numeric (リテラル) • 102 - money (リテラル) • 103 - real (リテラル) • 104 - float (リテラル) • 105 - identity (リテラル) • 106 - timestamp (リテラル) • 107 - sensitivity (リテラル) • 110 - univarchar
length	int	データ型の値の最大長。 decimal または money のマスクを使用している UDD では、値は最大精度より 4 桁多くなる。
status	int	ステータス。 (rs_columns テーブルの status カラムを参照)。

カラム	データ型	説明
length_err_act	tinyint	値が length の長さを超えた場合の処理。値は次のとおり。 • 1 - エラー。 • 2 - 継続する。 • 3 - 左端を切り詰める。 • 4 - 右端を切り詰める。 • 5 - 切り上げる。 • 6 - 切り上げて、エラー時には継続する。 • 7 - 切り上げて、エラー時にはデフォルト値を使用する。 • 8 - 切り上げて、エラー時には最小値を使用する。 • 9 - 切り上げて、エラー時には最大値を使用する。 • 10 - 切り捨てる。 • 11 - 切り捨てて、エラー時には継続する。 • 12 - 切り捨てて、エラー時にはデフォルト値を使用する。 • 13 - 切り捨てて、エラー時には最小値を使用する。 • 14 - 切り捨てて、エラー時には最大値を使用する。
mask	varchar(255)	データ型マスク。 null 以外のマスクの場合、基本データ型は char であることが必要。
scale	int	小数点以下の最大桁数。 マスクが money または decimal の場合のみ有効。
default_len	tinyint	default_val カラムの値の長さ。 default_val 訳文不要■
default_val	binary(255)	デフォルト値。このデータ型へ変換するとき、変換後のデータに値が欠落している場合に適用される値。
delim_pre_len	tinyint	delim_pre 値の長さ。

カラム	データ型	説明
delim_pre	binary(30)	Java 以外の値をファンクション文字列にマップする場合に使用されるポストフィクス文字または文字列。基本データ型のデリミタプレフィクスが使用されている場合は空の文字列。
delim_post_len	tinyint	delim_post の長さ。
delim_post	binary(30)	Java 以外の値をファンクション文字列にマップする場合に使用されるポストフィクス文字または文字列。基本データ型のデリミタプレフィクスが使用されている場合は空の文字列。
min_boundary_len	tinyint	min_boundary カラムの値の長さ。 • 1 - エラー。 • 2 - デフォルトを使用。 • 3 - 最小値を使用。 • 4 - 最大値を使用。
min_boundary	binary(255)	データ型に有効な最小値。
min_boundary_err_act	tinyint	値が min_boundary に設定されている最小値に満たない場合の処理。 値は次のとおり。 • 1 - エラー。 • 2 - デフォルトを使用。 • 3 - 最小値を使用。 • 4 - 最大値を使用。
max_boundary_len	tinyint	max_boundary の値の長さ。
max_boundary	binary(255)	データ型に有効な最大値。

カラム	データ型	説明
maximum_ boundary_err_act	tinyint	値が max_boundary で設定されている最大値を超えた場合の処理。 値は次のとおり。 • 1 - エラー。 • 2 - デフォルトを使用。 • 3 - 最小値を使用。 • 4 - 最大値を使用。
rowtype	tinyint	このローがローカルな Replication Server にだけ分配されるか、ドメイン内のすべての Replication Server に分配されるかを示す。値は次のとおり。 • 0 - ローカル • 1 - グローバル
canonic_type	tinyint	DSI は、動的 SQL 実行コマンドを送信するときに、canonic_type の値を使用して UDD を正しいデータ型に変換する。 255 は、データ型に動的 SQL との互換性がないことを示す。

インデックス

- dtid にユニークインデックス
- name にユニークインデックス
- classid にユニークでないインデックス
- prsid にユニークでないインデックス

rs_dbreps

名前セットを除き、データベース複写定義についてのすべての情報を格納します。このテーブルは、バージョン 12.6 以降のすべてのサイトに複写されます。

カラム	データ型	説明
dbrepid	rs_id	データベース複写定義の ID

カラム	データ型	説明
dbrepname	varchar (255)	データベース複写定義の名前
prsid	int	プライマリ Replication Server の ID
dbid	int	プライマリデータベースの ID
ownerid	rs_id	データベース複写定義を作成した Replication Server ユーザ
requestdata	datetime	データベース複写定義が作成された時間
status	int	サブセット内容のビットマップ • 0x0001 - テーブルリストが含まれる • 0x0002 - テーブルが無効である • 0x0004 - ファンクションリストが含まれる • 0x0008 - ファンクションが無効である • 0x0010 - トランザクションリストが含まれる • 0x0020 - トランザクションが無効である • 0x0040 - システムプロシージャリストが含まれる • 0x0080 - システムプロシージャが無効である • 0x0100 - DDL を複写しない • 0x0200 - **update** リストが含まれる • 0x0400 - このリストでは **update** 文の複写は無効である • 0x0800 - **delete** リストが含まれる • 0x1000 - このリストでは **delete** 文の複写は無効である • 0x2000 - **select into** リストが含まれる • 0x4000 - このリストでは **select into** 文の複写は無効である • 0x8000 - **insert select** リストが含まれる • 0x10000 - このリストでは **insert select** 文の複写は無効である
minvers	int	このテーブルを複写可能な Replication Server の一番古いバージョン

インデックス

dbrepid、dbid、dbrepname にユニークインデックス

rs_dbsubsets

データベース複写定義の名前セットを格納する。このテーブルは、バージョン 12.6 以降のすべてのサイトに複写される。

カラム	データ型	説明
dbrepid	rs_id	データベース複写定義の ID
prsid	int	プライマリ Replication Server の ID
type	char	名前の種類 • T - テーブル名 • F - ファンクション名 • X - トランザクション名 • P - システムプロシージャ名 • U - **update** コマンド • L - **delete** コマンド • I - **insert select** コマンド • S - **select into** コマンド
owner	varchar (30)	テーブルまたはファンクションの場合は所有者名、トランザクションまたはシステムプロシージャの場合は実行したユーザの名前。 * はすべての所有者またはユーザを表す。
name	varchar (255)	テーブル、ファンクション、トランザクション、またはシステムプロシージャの名前。 * はすべてのテーブル、ファンクション、トランザクション、システムプロシージャを表す。

カラム	データ型	説明
status	int	エントリのステータスを次のように示すビットマスク。 • 0x01 - マテリアライゼーションのみ • 0x02 - テーブルリストに追加 • 0x04 - テーブルリストから削除 • 0x08 - 例外リストに追加 • 0x10 - 例外リストから削除 • 0x20 - このデータベース複写定義の最終エントリ • 0x40 - dbsubsets の更新のみ
priority	int	テーブルの優先度。

インデックス

dbrepid、subtype、owner、name にユニーククラスタードインデックス

rs_dbversion

Replication Server システムデータベース (RSSD) のバージョンおよび Replication Server ユーザデータベースオブジェクトの最小互換性バージョンを格納します。

rs_dbversion テーブルは RSSD ではなく、各ユーザデータベースに保存されます。

カラム	データ型	説明
last_rssd_ver	int	最新のデータベースアップグレードの RSSD バージョンを取得する。
min_compat_ver	int	ユーザデータベースオブジェクトが現在のユーザデータベースと互換性がある、Replication Server の最小 RSSD バージョンを示す。

rs_dependtbls

データベース複写定義への追加または削除が保留中のテーブルを格納します。

カラム	データ型	説明
dbrepid	rs_id	このローのテーブルが処理を保留されている対象の複写定義 ID。
dbsubid	rs_id	このローのテーブルが処理を保留されている対象のサブスクリプション ID。
owner	varchar(30)	テーブル所有者名。
name	varchar(255)	テーブルの名前。
repdb	int	レプリケートデータベースの名前。
primdb	int	プライマリデータベースの名前。
status	int	ステータスフラグ: • 0x01 - 新規要求 • 0x02 - 進行中の要求 • 0x03 - 要求の処理エラー
action	int	テーブルに対する現在のアクションを示すフラグ: • 0x01 - 待機中 • 0x02 - 追加 • 0x03 - 削除中
request	int	テーブルに対して要求されたアクションを示すフラグ: • 0x01 - テーブルの追加 • 0x02 - テーブルの削除 • 0x03 - テーブルの再マテリアライズ

インデックス

dbsubid、owner、name にユニーククラスタードインデックス

rs_dictionary

パスワードで許可されていない文字列の組み合わせを格納します。

管理者は独自のスクリプトを使用して、文字と数値の組み合わせを dictionary テーブルに入力する必要があります。

カラム	データ型	説明
words	varchar(30)	許可されていない文字の組み合わせ

インデックス

- words にユニーククラスタードインデックス

rs_diskaffinity

ディスクパーティションとデータベースコネクションまたはルートの関係についての情報を格納します。

カラム	データ型	説明
partition_id	int	Replication Server が割り当てたパーティション ID。
dbid_or_siteid	int	Replication Server またはデータベースの ID。
status	int	関係のステータス。正しい値は、次のとおり。 • 0x01 - 有効 • 0x02 - 旧式

インデックス

dbid_or_siteid にユニーククラスタードインデックス

rs_diskpartitions

Replication Server がステーブルメッセージキュー用に使用するディスクパーティションについての情報を格納します。

カラム	データ型	説明
name	varchar(255)	ディスクデバイスの OS 上の名前。
logical_name	varchar(30)	ユーザがパーティションに指定した名前。
id	int	Replication Server が割り当てたパーティション ID。
num_segs	int	パーティションの総セグメント数。

カラム	データ型	説明
status	int	ディスクパーティションのステータスを示す。正しい値は、次のとおり。 • 1 - オンライン中。 • 2 - 削除中。 • 16 - パーティションは自動的にサイズ変更可能。
vstart	int	Replication Server がパーティションに書き込みを開始するオフセット位置 (MB 単位)。

インデックス

- logical_name にユニーククラスタードインデックス
- name にユニークインデックス

rs_encryptionkeys

Replication Server を使用して暗号化キーを格納します。

カラム	データ型	説明
name	varchar(30)	暗号化キーの名前。
value	binary(128)	暗号化キーの値。
status	int	暗号化キーのステータス。
ctime	datetime	作成日時または最終更新日時。

インデックス

- name に、ユニーククラスタードインデックス

rs_erroractions

データサーバのエラー番号に対して、Replication Server によって行われるアクションをマップします。

カラム	データ型	説明
ds_errorid	int	データサーバのエラー番号。
errorclassid	rs_id	エラークラス ID (「*rs_classes*」を参照)。
action	tinyint	エラー発生時の処理。 • 1 - エラーを無視する。 • 2 - 複写を停止する。 • 3 - 警告メッセージを出力する。 • 4 - 例外ログにエントリを書き込む。 • 5 - トランザクションをリトライし、再びエラーになった場合はトランザクションをログに記録する。 • 6 - トランザクションを一定の回数リトライし、引き続きエラーになる場合は複写を停止する。
prsid	int	このローがプライマリであるサイト。

インデックス

- ds_errorid, errorclassid にユニークインデックス
- errorclassid にクラスタードインデックス

rs_exceptscmd

例外ログからトランザクションのテキストを検索するために使用する情報を格納します。

テキストは rs_systext システムテーブルに格納され、次の情報が含まれます。

- ソースコマンド - Replication Server が受信したユーザトランザクションのテキスト。
- 出力コマンド - Replication Server がファンクション文字列をもとにデータベースに対して生成したトランザクションのテキスト。出力コマンドは、言語コマンドまたは RPC のいずれかになります。

rs_exceptscmd には、ソースコマンドまたは出力コマンドごとに 1 つのローが割り当てられます。

カラム	データ型	説明
sys_trans_id	rs_id	このトランザクションにシステムが割り当てたトランザクション ID
src_cmd_line	int	ログに記録されたトランザクション内での、ソースコマンドの行番号
output_cmd_index	int	ログに記録されたトランザクション内での、出力コマンドの行番号
cmd_type	char(1)	コマンドのタイプ。 • S - ソースコマンド • L - 言語出力コマンド • R - RPC 出力コマンド
cmd_id	rs_id	*rs_systext* へのインデックス

インデックス

cmd_id にユニークインデックス

rs_exceptshdr

失敗したトランザクション情報を格納します。 トランザクションのソースコマンドと出力コマンドは、システムテーブルの rs_exceptscmd と rs_systext に格納されます。 rs_exceptscmd と rs_exceptshdr では、トランザクションに対するローはすべて sys_trans_id カラムで識別されます。

カラム	データ型	説明
sys_trans_id	rs_id	このトランザクションにシステムが割り当てたトランザクション ID
rs_trans_id	binary(120)	Replication Server が生成したユニークなトランザクション ID。
app_trans_name	varchar(30)	ユーザが指定したトランザクション名。
orig_siteid	int	オリジンデータベースの ID。

カラム	データ型	説明
orig_site	varchar(30)	オリジンデータベースのデータサーバ名。
orig_db	varchar(30)	オリジンデータベースの名前。
orig_time	datetime	トランザクションが開始された時間。
orig_user	varchar(30)	オリジンサイトでトランザクションを実行したユーザ。
error_siteid	int	エラーが発生したサイトの ID。
error_site	varchar(30)	エラーが発生したデータサーバの名前。
error_db	varchar(30)	エラーが発生したデータベースの名前。
log_time	datetime	エラーが発生した時間。
ds_error	int	データサーバのエラー番号。
ds_errmsg	varchar(255)	データサーバのエラーメッセージ。
error_src_line	int	エラーが発生したコマンドの行番号。
error_proc	varchar(255)	エラーが発生したプロシージャ。
err_output_line	int	エラーが発生した出力コマンドの行番号。
log_reason	char(1)	トランザクションがログに記録された理由。 • O - DSI キューに孤立したトランザクションが存在する。 • E - データサーバのエラーが LOG または RETRY_LOG にマップされている。 • S - **skip transaction** オプションを指定した **resume connection** コマンドが実行されたため、トランザクションがスキップされた。 • D - **sysadmin log_first_tran** コマンドによってトランザクションがログに記録された。

カラム	データ型	説明
trans_status	smallint	トランザクションのステータス。次のうち 1 つ以上。 • 0x0001 - 孤立したトランザクション。 • 0x0002 - ログに記録されたトランザクションは、プライマリサイトに送信が予定されている。 • 0x0004 - 矛盾したトランザクション。
retry_status	smallint	トランザクションのリトライステータス。次のいずれか。 • 1 - リトライは成功した。 • 2 - トランザクションがコミットされていない。
app_usr	varchar(30)	エラーが発生したサイトでトランザクションを適用したユーザの名前。
app_pwd	varchar(30)	エラーが発生したサイトでトランザクションを適用したユーザのパスワード。

インデックス

sys_trans_id にユニークインデックス

rs_exceptslast

オリジン ID、セカンダリキュー ID、例外ログに書き込まれた最後のログトランザクションに関する情報が格納されます。

カラム	データ型	説明
error_db	int	エラーが発生したデータベース
origin	int	トランザクションのオリジンデータベース
origin_qid	binary(36)	このオリジンからの最後のトランザクションのキュー ID
secondary_qid	binary(36)	このオリジンから最後にログに記録されたトランザクションのセカンダリキュー ID

カラム	データ型	説明
status	tinyint	トランザクションのステータス • 0 - 有効。このオリジンで失われたトランザクションはない。 • 1 - ロスを検出。このオリジンでトランザクションが失われていないか確認が必要。 • 2 - ロスの検出後にメッセージを拒否。このオリジンに対してトランザクションが失われた可能性がある。
origin_time	datetime	トランザクションのオリジンでの時間。
log_time	datetime	トランザクションがログに記録された時間
lorigin	int	メッセージが生成された論理データベース

インデックス

- error_db、origin にユニークインデックス
- error_db、origin、status にユニークインデックス

rs_funcstrings

それぞれのファンクションに関連するファンクション文字列を格納します。

カラム	データ型	説明
prsid	int	このローがプライマリであるサイト。
classid	rs_id	ファンクション文字列が属するクラス。
funcid	rs_id	この文字列に対応するファンクション。
name	varchar(255)	ファンクション文字列名。
fstringid	rs_id	このファンクション文字列の ID。

カラム	データ型	説明
attributes	int	ファンクション文字列の属性。 • 0x01 - 矛盾したファンクション。 • 0x02 - RPC。 • 0x04 - 変更されている。 • 0x08 - **rs_writetext** 以外のすべてのファンクションに使用され、ファンクションが出力コマンドを持たず、レプリケートデータサーバに何も送信されないことを示す。 • 0x10 - デフォルト入力。 • 0x20 - デフォルト出力。 • 0x40 - **rs_writetext** ファンクション文字列に、**writetext** 出力が使用される。 • 0x80 - **rs_writetext** ファンクション文字列に、**with log** オプションを指定した **writetext** 出力が使用される。 • 0x100 - **rs_writetext**、**rs_textptr_init**、または **rs_get_textptr** ファンクションに対するファンクション文字列。 • 0x200 - **rs_writetext** ファンクション文字列に、**no log** オプションを指定した **writetext** 出力が使用される。 • 0x400 - ファンクション文字列に、非キーカラムの値にアクセスする変数が 1 つ以上含まれている。 • 0x800 - 出力言語テンプレートで *rs_default_fs* システム変数が使用された。 • 0x1000 - **none** 出力が **rs_writetext** ファンクション文字列に使用される。
parameters	smallint	このファンクション文字列内のパラメータ数。
param_hash	int	入力テンプレートのハッシュ値。
expiredate	datetime	ファンクション文字列の期限が切れる日付。これは動的ファンクション文字列の期限に対して使用される。
rowtype	tinyint	ローが複写されている場合は 1、そうでない場合は 0。

カラム	データ型	説明
minvers	int	ファンクション文字列をサポートするのに必要な最小バージョン。 つまり、ファンクション文字列に値が 15.0 の minvers がある場合、15.0 を下回るサイトには複写しない。

インデックス

- classid、funcid、name にユニーククラスタードインデックス
- fstringid にユニークインデックス
- funcid にユニークでないインデックス

rs_functions

Replication Server ファンクションに関する情報を格納します。

rs_functions は、Replication Server 15.7 より前のバージョンのシステムテーブルです。バージョン 15.7 以降では、rs_functions は rs_clsfunctions および rs_objfunctions システムテーブルの union のビューです。

カラム	データ型	説明
prsid	int	このファンクションがプライマリであるサイト。
funcname	varchar(255)	ファンクションの名前。
funcid	rs_id	ファンクションの ID。
objid	rs_id	ファンクションが適用されるオブジェクト。クラススコープのファンクションの場合、このカラムには NULL_OBJECT_ID (0x00000000) が格納される。
conflicting	tinyint	ファンクションが矛盾している場合は 1、そうでない場合は 0。
userdefined	bit	ユーザ定義のファンクションの場合は 1、そうでない場合は 0。
rowtype	tinyint	ローが複写されている場合は 1、そうでない場合は 0。

インデックス

注意： インデックスは、rs_functions が 15.7 より前のバージョンの Replication Server のテーブルである場合にのみ存在します。

- objid にクラスタードインデックス
- objid、funcname にユニークインデックス
- funcid にユニークインデックス

rs_idnames

ID サーバで認識されている Replication Server とデータベースの名前を格納します。rs_idnames テーブルは、ID サーバサイトだけに関連のあるテーブルです。

カラム	データ型	説明
name1	varchar(30)	Replication Server またはデータサーバの名前。
name2	varchar(30)	データベース名 (Replication Server の場合は “”)。
type	int	Replication Server またはデータベース。 • 8 - Replication Server • 9 - データベース
id	int	Replication Server またはデータベースに割り当てられたユニークな ID。
ltype	char(1)	データベースのタイプ。 • P - 物理データベース • L - 論理データベース

インデックス

name1、name2、ltype にユニーククラスタードインデックス

rs_ids

さまざまなタイプのオブジェクトに対して使われている最新の ID を格納します。

カラム	データ型	説明
typename	varchar(30)	このオブジェクトタイプの名前 (“subscriptions”、“objects” など)
objid	int	このオブジェクトタイプに使用された最新の ID

objtype	tinyint	• オブジェクトタイプ。
		  • 1 - サブスクリプション
		  • 2 - オブジェクト
		  • 3 - クラス
		  • 4 - ユーザ
		  • 5 - ファンクション
		  • 6 - ファンクション文字列
		  • 7 - エラーログ
		• 例外ログのタイプ
		  • 12 - トランザクションを拒否
		• サイト ID のタイプ
		  • 8 - Replication Server の ID
		  • 9 - データベース ID
		• ステーブルキューのパラメータ
		  • 10 - ディスクパーティション ID
		• サブスクリプションモジュールで使用されるカウンタ
		  • 13 - サブスクリプションモジュールのカウンタ
		• リカバリマネージャの ID
		  • 14 - リカバリ ID タイプ
		  • 15 - 再マテリアライゼーション ID
		  • 16 - パブリケーション ID
		  • 17 - アーティクル ID
		  • 18 - **where** 句の ID
		  • 19 - UDD の ID

インデックス

objtype にユニーククラスタードインデックス

rs_lastcommit

Replication Server は、rs_lastcommit テーブルの情報を使用して各データの送信元でコミットされた最新のトランザクションを検索します。

rs_lastcommit テーブルは RSSD ではなく、各ユーザデータベースに保存されます。

カラム	データ型	説明
origin	int	ローがあるプライマリデータベースの ID 番号。
origin_qid	binary	オリジンデータベースのステーブルキュー内にある最後にコミットされたトランザクションを示す。
secondary_qid	binary	サブスクリプションマテリアライゼーションキューがオリジンデータベースに存在する場合、このカラムには、そのキューにあるレプリケートデータベースにコミットされた最新のトランザクションが含まれる。
origin_time	datetime	トランザクションのオリジンでの時間。
dest_commit_time	datetime	トランザクションが送信先でコミットされた時間。
conn_id	int	コネクション ID。
pad1	binary(255)	1 データページに 1 ローが格納されるように、ローの長さを補うためのもの。
pad2	binary(255)	1 データページに 1 ローが格納されるように、ローの長さを補うためのもの。
pad3	binary(255)	1 データページに 1 ローが格納されるように、ローの長さを補うためのもの。
pad4	binary(255)	1 データページに 1 ローが格納されるように、ローの長さを補うためのもの。

カラム	データ型	説明
pad5	binary(255)	1 データページに 1 ローが格納されるように、ローの長さを補うためのもの。
pad6	binary(255)	1 データページに 1 ローが格納されるように、ローの長さを補うためのもの。
pad7	binary(255)	1 データページに 1 ローが格納されるように、ローの長さを補うためのもの。
pad8	binary(83)	1 データページに 1 ローが格納されるように、ローの長さを補うためのもの。

インデックス

origin、conn_id にユニーククラスタードインデックス

rs_locater

ステーブルキューで送信側から受信した最新のロケータフィールドを格納します。

カラム	データ型	説明
sender	int	送信側のサイト ID。
type	char(1)	このローの使用者。 • R - RSI (ルート) • D - サブスクリプションに使用されるディストリビュータのロケータ • E - Replication Agent のエグゼキュータ • U - 最新のシステムアップグレードでのロケータ • W - ウォームスタンバイアプリケーションで使用されるディストリビュータのロケータ • C - Replication Agent によって送られてくる、前回正常に実行した複写定義要求のロケータ。 • F - Replication Agent によって送られてくる失敗したコマンドのロケータ。 • S - Replication Server によってスキップされた失敗したコマンドのロケータ。
locater	binary(36)	この送信元から受信された最新のキュー ID。

インデックス

sender、*type* にユニーククラスタードインデックス

rs_maintusers

Replication Server が他の Replication Server またはデータサーバにアクセスするために使用する、ユーザのログイン名とパスワードを格納します。

カラム	データ型	説明
destid	int	ログインする Replication Server またはデータベースのサイト ID。
username	varchar(30)	Replication Server の RSI ユーザまたはデータベースのメンテナンスユーザのユーザ名。
password	varchar(30)	パスワード
use_enc_password	int	• 0 - 通常のパスワードを使用 • 1 - 暗号化されたパスワードを使用
enc_password	varchar(66)	現在の暗号化ユーザパスワードを格納する。
new_password	varchar(66)	現在の有効期間のために生成された、新規のランダムな暗号化パスワードを格納する。
attributes	int	アカウントとパスワードのステータスと設定。 • 0x0 - 最初のパスワード。 Replication Server パスワードデーモンは、デーモンの次回ウェイクアップ時に、パスワードのステータスを最初のパスワードから隠しパスワードに変更する。 • 0x01 - 隠しパスワード。 Replication Server は、最初のパスワードをランダムパスワードに変更した。
password_date	datetime	最後にメンテナンスユーザのパスワードを変更した日付。

カラム	データ型	説明
expiration_interval	smallint	メンテナンスユーザのパスワードの有効期限 (日数)。

インデックス

destid にユニーククラスタードインデックス

rs_msgs

インストール時やいくつかの Replication Server ストアドプロシージャの実行時に使用される、ローカライズされたエラーメッセージを保存します。

カラム	データ型	説明
msgnum	int	メッセージのユニークな ID 番号。
langname	char(30)	メッセージテキストのローカライズに使用されている言語。RSSD である Adaptive Server の *@@language* グローバル変数に対応した名前。
msgtxt	varchar(255)	メッセージのテキストを指定の言語でローカライズしたもの。

インデックス

msgnum、langname にユニーククラスタードインデックス

rs_objects

1 つのローごとに 1 つの複写定義を格納します。

カラム	データ型	説明
prsid	int	このオブジェクトが作成されたプライマリ Replication Server。
objname	varchar(255)	オブジェクト名。
objid	rs_id	オブジェクト ID。
dbid	int	データサーバとデータベースのユニークな ID。

カラム	データ型	説明
objtype	char(1)	次のいずれかのオブジェクトタイプ。 • R - テーブル複写定義 • F - ファンクション複写定義

カラム	データ型	説明
attributes	int	マスク。次のうち 1 つ以上。 • 0x01 - 動的なファンクション文字列を生成する。 • 0x02 - 複写定義に bigdatetime または bigtime カラムが含まれ、Replication Server 15.5 以降にのみ送信できる。 • 0x04 - 複写定義に対して最少カラムが有効になっている。 • 0x08 - 複写定義に *identity* カラムが含まれる。 • 0x10 - **replicate_if_changed** ステータス。 • 0x20 - 複写定義に保留中の削除操作がある。 • 0x40 - 複写定義に text、unitext、または image カラムが含まれる。 • 0x80 - 複写定義はスタンバイデータベースで使用される。 • 0x0100 - 複写定義のカラムはスタンバイデータベースに送信される。 • 0x0200 - 複写定義はバージョン 11.0.x 以前の Replication Server に送信される。 • 0x0400 - 複写定義はプライマリテーブルに対する基本複写定義として使用されている。 • 0x0800 - 複写定義は内部専用である。 • 0x1000 - プライマリテーブルとレプリケートテーブルで、オブジェクト名またはカラム名が異なる。 • 0x4000 - 複写定義にカラムレベルでの変換が含まれる。 • 0x8000 - 複写定義に UDD で宣言されたカラムが含まれる。 • 0x10000 - 複写定義に 255 バイトよりも大きい char、varchar、binary、または varbinary カラムが含まれ、

カラム	データ型	説明
		Replication Server 12.5 以降にのみ送信できる。 • 0x20000 - 複写定義に `unichar` または `univarchar` カラムが含まれ、Replication Server 12.5 以降にのみ送信できる。 • 0x40000 - 複写定義に `date` または `time` カラムが含まれ、Replication Server 12.6 以降にのみ送信できる。 • 0x80000 - 複写定義に `timestamp` カラムが含まれる。 Replication Server 15.1 には `timestamp`、Replication Server 15.0.1 以前には `varbinary` として送信される。 • 0x200000 - 適用ファンクション複写定義であり、Replication Server 15.1 にのみ送信できる。 • 0x400000 - 要求ファンクション複写定義であり、Replication Server 15.1 にのみ送信できる。 • 0x800000 - 動的 SQL はテーブルには使用されない。 • 0x2000000 - SQL の複写で **update** が有効である。 • 0x4000000 - SQL の複写で **delete** が有効である。 • 0x8000000 - SQL の複写で **insert select** が有効である。 • 0x10000000 - SQL の複写が **repserver** によって内部的に無効にされている。
`ownertype`	`char(1)`	このオブジェクトの所有者タイプ。 • U - ユーザ • S - システム
`crdate`	`datetime`	作成された日時。

カラム	データ型	説明
parentid	rs_id	今後のために予約済み。
ownerid	rs_id	このオブジェクトを作成したユーザの ID。
rowtype	tinyint	ローが複写される場合は 1、そうでない場合は 0。
phys_tablename	varchar(255)	プライマリテーブル名。このオブジェクトについてデータサーバと通信するときに使用する。
deliver_as_name	varchar(255)	レプリケートテーブルまたはストアドプロシージャの名前。
phys_objowner	varchar(30)	複写定義に指定されているプライマリテーブルの所有者名。 テーブル所有者が指定されていない場合は、ブランク。
repl_objowner	varchar(30)	複写定義に指定されているレプリケートテーブルの所有者名。 テーブル所有者が指定されていない場合は、ブランク。
has_baserepdef	rs_id	これが基本複写定義でない場合、has_baserepdef の値は基本複写定義の objid と同じ。 基本複写定義の場合は、次の値を持つ。 0x00 - 基本複写定義
minvers	int	複写定義の最低バージョン、つまり送信可能な Replication Server の最低バージョン。値は次のとおり。 • 1200 - バージョン 12 以降の Replication Server へ送信する。 • 1150 - バージョン 11.5 以降の Replication Server へ送信する。 • 1000 または 0 (ゼロ) - 任意の Replication Server へ送信する。 • 0 (ゼロ) - ファンクション複写定義またはシステム複写定義の場合。

カラム	データ型	説明
version	rs_id	複写定義バージョンをユニークに識別する。
active_inbound	int	エグゼキュータは、このカラムを使用して、使用する複写定義バージョンを決定する。エグゼキュータは、active_inbound カラムの値が 0 の複写定義バージョンを使用する。
attributes2	int	• 0x01 - ディストリビュータはこれを使用して、ディストリビュータが使用していない複写定義バージョンを識別する。 • 0x02 - スタンバイ DSI はこれを使用して、スタンバイ DSI が使用していない複写定義バージョンを識別する。

インデックス

- objname にユニーククラスタードインデックス
- dbid、phys_tablename、phys_objowner、objtype、has_baserepdef、active_inbound にユニークインデックス
- objid にユニークインデックス
- version にユニークインデックス

rs_objfunctions

複写定義のユーザ関数についての情報を格納します。

カラム	データ型	説明
prsid	int	このファンクションがプライマリであるサイト。
funcname	varchar(255)	ファンクションの名前。
funcid	rs_id	ファンクションの ID。
objid	rs_id	複写定義またはファンクションが適用されるターゲットオブジェクトのオブジェクト ID。
conflicting	tinyint	ファンクションが矛盾している場合は 1、そうでない場合は 0。

カラム	データ型	説明
userdefined	bit	ユーザ定義のファンクションの場合は 1、そうでない場合は 0。
rowtype	tinyint	ローが複写されている場合は 1、そうでない場合は 0。

インデックス

- objid にクラスタードインデックス
- objid、funcname にユニークインデックス
- funcid にユニークインデックス

rs_oqid

オリジンサイトから受信した最新のキュー ID を格納し、トランケーションポイントのリセットの調整にも使用されます。

カラム	データ型	説明
origin_site_id	int	オリジンサイトのサイト ID。
q_number	int	キュー番号。
q_type	int	キュータイプ。
origin_q_id	binary(36)	オリジンデータベースでのコマンド ID。
local_q_id	binary(36)	キューのローカル ID。
valid	int	確定化ステータス。 • 0 - 有効 • 1 - ロスを検出 • 2 - ロスの検出後にメッセージを拒否
origin_lsite_id	int	オリジンサイトの論理データベースのサイト ID。

インデックス

- origin_site_id、q_number、q_type にユニーククラスタードインデックス

rs_passwords

Replication Server にアクセスする各ユーザのパスワード履歴を格納します。

カラム	データ型	説明
uid	rs_id	ユーザ ID。
enc_password	varchar(66)	暗号化パスワード。
password_date	datetime	パスワードが最初に設定された日付。

インデックス

- uid のインデックス

rs_profdetail

Replication Server プロファイルと関連付けられた詳細を記録します。

カラム	データ型	説明
profid	rs_id	*rs_profile* テーブルの外部キーであるプロファイル ID。
name	varchar(255)	プロファイル名。空文字列でもかまわない。
pdetailtype	int	プロファイルに対して実行するアクションを指定する。 • 1 - コネクション設定を **create connection** コマンドに追加する。 • 2 - RSSD のクラスレベル変換の定義を実行する。 • 3 - レプリケートデータベースのレプリケートデータベースオブジェクトの定義を実行する。
pdetailid	rs_id	rs_systext テーブルの外部キーであるプロファイルの詳細 ID。

カラム	データ型	説明
sequence	int	プロファイル内でのプロファイルの詳細シーケンスを示す。Replication Server は、シーケンスを使用して、プロファイルの詳細アクションの実行順序を決定する。 **注意：** Replication Server の **create connection** オプションは、プロファイルの詳細のシーケンスで示される順序に関係なく、常に最初に実行されます。

インデックス

- profid、sequence にユニークインデックス
- id にユニークインデックス
- profid にユニークでないインデックス

rs_profile

現在定義されている Replication Server プロファイルを格納します。

カラム	データ型	説明
name	varchar(255)	プロファイル名。
vers	varchar(255)	プロファイルのバージョン。空文字列でもかまわない。
id	rs_id	プロファイルの ID。
type	char(1)	プロファイルタイプ • C - 接続プロファイル
comments	varchar(255)	プロファイルの説明と情報。

インデックス

- name、vers、type にユニークインデックス
- type にユニークでないインデックス

rs_publications

この Replication Server に認識されているパブリケーションについての情報を格納します。

カラム	データ型	説明
prsid	int	パブリケーションが作成されたプライマリ Replication Server。
pubname	varchar(255)	パブリケーションの名前。
pubid	rs_id	ユニークなパブリケーション ID。
pdbid	int	パブリケーションのプライマリデータサーバとデータベースに対するユニークな ID。
requestdate	datetime	最後にアーティクルがパブリケーションに追加された日時。
ownerid	rs_id	パブリケーションを作成したユーザの ID。
status	int	パブリケーションのステータス。 • 0x00 - 無効 • 0x01 - 有効
minvers	int	このパブリケーションをサポートするのに必要な Replication Server の最小バージョン。

インデックス

- pubname、pdbid にユニーククラスタードインデックス
- pubid にユニークインデックス

rs_queuemsg

Replication Server のキューを RSSD にダンプすると、キューエントリが rs_queuemsg に格納されます。 rs_queuemsg テーブルが特定セグメントのローをすでに持っている場合、これらのローは、そのセグメントの最も最近のローをダンプする前に、テーブルから削除されます。

カラム	データ型	説明
q_number	int	キュー番号。
q_type	int	キュータイプ。
q_seg	int	キューセグメント。
q_blk	int	キューブロック。
q_row	int	キューロー。
len	int	キューエントリの長さ。
origin_site_id	int	オリジンサイトの ID。
origin_q_id	binary(36)	オリジンによって割り当てられたキュー ID。
origin_time	datetime	トランザクションが開始された時間。
origin_user	varchar(30)	オリジンサイトでトランザクションを実行したユーザ。
tran_name	varchar(30)	トランザクション名。
local_q_id	binary(36)	ローカル Replication Server によって割り当てられたキュー ID。
status	int	メッセージのステータス。
reserved	int	今後のために予約済み
tran_len	smallint	*tran_id* の長さ。
txt_len	smallint	コマンドの長さ。
tran_id	binary(120)	トランザクション ID
lorigin_site_id	int	キューエントリの送信元である論理コネクションのサイト ID。
version	int	メッセージのリリースバージョン。

インデックス

q_number、q_type、q_seg、q_blk、q_rowにユニーククラスタードインデックス

rs_queuemsgtxt

ステーブルキューにあるメッセージのコマンドまたはテキスト部分を格納します。ステーブルキューの各エントリに対して、このテーブルに 1 つ以上のローが格納されます。複数のローが使用されるのは、ステーブルキューエントリのデータの長さがコマンドフィールドの最大長である 255 バイトを超える場合です。

カラム	データ型	説明
q_number	int	キュー番号。
q_type	int	キュータイプ。
q_seg	int	メッセージを含むセグメント。
q_blk	int	セグメント内のメッセージを含むブロック。
q_row	int	ブロック内のメッセージを含むロー。
q_seq	int	このエントリに対するローのシーケンス番号。
txt	varchar(255)	エントリのテキスト。
txtbin	binary(255)	バイナリでのテキスト。

インデックス

q_number、q_type、q_seq、q_seg、q_blk、q_row にユニークデフォルトインデックス

rs_queues

サイトリカバリを可能にする情報を格納します。このテーブルは、Replication Server のステーブルキューマネージャと、保証された配信システムによって使用されます。

カラム	データ型	説明
number	int	キュー ID。次のどちらかを表す数字を格納する。 • インバウンドキューの送信元データベース • アウトバウンドキューの送信先データベースまたは Replication Server データベースの場合は *rs_databases* システムテーブルの *dbid* カラムの値、Replication Server の場合は *rs_sites* システムテーブルの *id* カラムの値に対応。
type	int	キュータイプ。 • 0 - アウトバウンドキュー • 1 - インバウンドキュー • 大きな負の数字 - サブスクリプションマテリアライゼーションキュー
state	int	このキューの現在のステータス。 • 0 - 障害発生 • 1 - アクティブ。 • 2 - 削除中
twosave	int	セグメントにあるすべてのメッセージがターゲットに通知された後、Replcation Server が SQM セグメントを維持する秒数を指定する。**-1** の場合は strict 設定。
truncs	int	トランケーションポイントの数。

インデックス

number、type にユニーククラスタードインデックス

rs_recovery

障害が発生した場合、リカバリ中に Replication Server によって実行されるアクションのログを記録します。

カラム	データ型	説明
action	int	リカバリ可能なアクションを示す。 • 1 - create_route • 2 - drop_route • 3 - スタンドアロンモード • 4 - キューの再構築 • 5 - リカバリの記録 • 6 - ログの先頭にある LTM の再起動 • 7 - スタンバイの作成 • 8 - switch active • 9 - DSI キューまたはマテリアライゼーションキューの strict セーブインターバル • 10 - switch active 実行後の DSI の 2 次重複検出の終了 • 11 - スタンバイの削除 • 12 - ディストリビュータロケータの変更 • 13 - 複写定義のあるセグメントの削除 • 14 - 保留中の複写定義の削除 • 15 - ハイバネーションモード • 16 - 保留中のサブスクリプションの削除 • 17 - ERSSD RepAgent によるプロセス管理 • 18 - リファレンスカウンタのある保留中のテーブルまたはファンクション複写定義の削除 • 19 - ルートアップグレードプロセスの続行 • 20 - ルートアップグレードのリカバリ • 21 直接ロードサブスクリプション ID の格納 • 22 - 直接ロードサブスクリプションのターゲットテーブル名の格納
id	rs_id	ローごとにユニーク ID が割り当てられる。

カラム	データ型	説明
seqnum	int	複数ローのアクションでは、このカラムに各ローのシーケンス番号が格納される。
state	int	定型のステータス番号を持つリカバリ可能アクションに対する現在のステータス。
text	binary(512)	アクションを完了するために必要なデータ。
textlen	int	テキストデータの長さ。

インデックス

id にユニークインデックス

rs_repdbs

プライマリ Replication Server で認識されているすべてのデータベース情報が保存されています。この情報は、レプリケートサイトのデータベースに対してサブスクリプションが実行されたときに格納されます。

カラム	データ型	説明
dbid	int	ユニークなデータベース ID。
dsname	varchar(30)	データサーバ名。
dbname	varchar(30)	データベース名
controllerid	int	このデータベースを管理する Replication Server。

インデックス

- controllerid にクラスタードインデックス
- dbid にユニークインデックス
- dsname、dbname にユニークインデックス

rs_repobjs

レプリケート Replication Server にある複写定義に対するオートコレクションフラグを格納します。**set autocorrection** コマンドを使用して、フラグをオンまたはオフに設定できます。

カラム	データ型	説明
objid	rs_id	複写定義のオブジェクト ID。
dbid	int	レプリケートデータが格納されているデータベースの ID。
attributes	int	有効な値。 • 0x01 - オートコレクションフラグがオンの場合 • 0x02 - 動的 SQL は複写定義に使用されない。

インデックス

objid、dbid にユニーククラスタードインデックス

rs_routes

ネットワークトラフィックに関するルートの情報を格納します。

カラム	データ型	説明
dest_rsid	int	データサーバまたは Replication Server の ID。
through_rsid	int	ルートが通過する Replication Server。 直接ルートの場合、through_rsid の値は、dest_id と同じ。
source_rsid	int	このルートが定義されている Replication Server。

カラム	データ型	説明
status	tinyint	ルートのステータス。 • 1 - 初期化中。 • 2 - ルートはこのサイトで有効 (送信元と送信先の Replication Server でステータスが 2 の場合にルートは有効)。 • 3 - 所定の処理が終了した後でこのルートを削除する。 • 4 - ただちにこのルートを削除する。
suspended	tinyint	値は次のいずれかになる。 • 0 - ルートはアクティブ。 • 1 - ルートはサスペンドされている。 • 2 - ルートは再構築中。トランケーションポイントを設定中。 • 3 - ルートはサスペンドされている。トランケーションポイントを設定中。 • 8 (マスク) - RSI アウトバウンドキュー用。レプリケート Replication Server が、この送信元の Replication Server に対して rs_locater テーブルの locater フィールドを 0 に設定するよう指示する。
src_version	int	このルートの送信元 Replication Server のバージョン。このバージョンは RSI のバージョンであることに注意 (**rs_config** ストアドプロシージャの *current_rssd_version* に表示されるバージョンではない)。 • 1000 - バージョン 10.1 より前の Replication Server には、すべてこのバージョンが割り当てられる。 • 1010 - バージョン 10.1 • 1100 - バージョン 11.0 • 1150 - バージョン 11.5 • 1200 - バージョン 12.0 サポートされている他のバージョン番号については、Replication Server のリリースノートを参照。

インデックス

dest_rsid、source_rsid にユニーククラスタードインデックス

rs_routeversions

ルートの各終端にある Replication Server のバージョン情報を格納します。

カラム	データ型	説明
dest_rsid	int	送信先 Replication Server の ID。
source_rsid	int	このルートが定義されている送信元 Replication Server の ID。
dest_rssd_id	int	送信先 Replication Server の RSSD の ID。
route_version	int	送信先と送信元の Replication Server のうち低い方のサイトバージョン。
min_path_version	int	今後のために予約済み
marker_serial_no	int	内部用
status	int	ルートのステータス。 • 0x00 - 有効。 • 0x01 - ルートのアップグレード／リカバリが進行中、または必要。 • 0x02 - ルートのアップグレード/リカバリが完了。これは、**sysadmin upgrade, "route"** を実行する際にシームレスルートアップグレードによって使用される一時的なステータスである。
proposed_version	int	移行中の新しいルート値。

インデックス

dest_rsid、source_rsid にユニーククラスタードインデックス

rs_rules

サブスクリプションのルールを格納します。この rs_rules には、サブスクリプションの句にある要素ごとに 1 つのローが存在します。

カラム	データ型	説明
prsid	int	このオブジェクトのプライマリ Replication Server。
subid	rs_id	このルールが適用されるサブスクリプションの ID。または、アーティクルへのサブスクリプションの場合には、このルールが適用される **where** 句の ID。
objid	rs_id	このサブスクリプションに対するテーブルまたはファンクション複写定義の ID。
dbid	int	サブスクリプションが作成されたデータが格納されているデータベースの ID。
subtype	int	サブスクリプションのタイプ。 • 0x01 - 範囲サブスクリプション • 0x02 - 等価サブスクリプション • 0x80 - アーティクルサブスクリプション
primary_sre	int	設定すると、サブスクリプションはプライマリ Replication Server のサブスクリプションレゾリューションエンジンに組み込まれる。
replicate_sre	int	設定すると、サブスクリプションはレプリケート Replication Server のサブスクリプションレゾリューションエンジンに組み込まれる。
colnum	smallint	基本カラム番号の値。
valuetype	tinyint	演算子のデータ型。SYBCHAR など。

カラム	データ型	説明
low_flag	tinyint	下限値のタイプに対するビットマップ。 • 0x01 - exclusive • 0x02 - inclusive • 0x04 - infinity • 0x08 - equality • 0x20 - rs_address
high_flag	tinyint	上限値のタイプに対するビットマップ。 • 0x01 - exclusive • 0x02 - inclusive • 0x04 - infinity • 0x08 - equality • 0x20 - rs_address
low_len	int	下限値の長さ。
high_len	int	上限値の長さ。
low_value	binary(255)	下限値のバイナリ表記。
high_value	binary(255)	上限値のバイナリ表記。
dtid	rs_id	複写定義に定義されている、カラムについて宣言されたデータ型の ID。

インデックス

- subid、colnum、primary_sre、replicate_sre、subtype にユニークインデックス
- subid、colnum にユニークインデックス
- objid、subtype、dbid にクラスタードインデックス

rs_schedule

Replication Server で作成するスケジュールについての情報を格納します。

カラム	データ型	説明
sched_name	varchar(30)	スケジュールの名前。

カラム	データ型	説明
sched_time	varchar(255)	制限された UNIX クロンスタイルの日付と時刻の文字列。Replication Server が指定されたオペレーションをいつ実行するのかを示します。
status	int	スケジュールのオン/オフを切り替える。正しい値は、次のとおり。 • 0 - オフ • 1 オン
type	int	スケジュールで実行するコマンドのタイプ。値は次のとおり。 • 0 - シェルコマンド
ownerid	rs_id	スケジュールを作成したユーザの ID。

インデックス

sched_name にユニーククラスタードインデックス

rs_scheduletxt

Replication Server で作成するスケジュールのコマンド部分を格納します。スケジュールの各エントリに対して、rs_scheduletxt テーブルに 1 つ以上のローが格納されます。複数のローが使用されるのは、コマンドがコマンドフィールドの最大長である 255 バイトを超える場合です。

カラム	データ型	説明
sched_name	varchar(30)	スケジュールの名前。
sequence	int	スケジュールに対するローのシーケンス番号。
textval	varchar(255)	シェルコマンドのフルパス。

インデックス

- sched_name、sequence にユニーククラスタードインデックス
- sched_name に部分インデックス

rs_schemamap

プライマリデータベースとレプリケートデータベースとの間のスキーママッピングを格納します。

カラム	データ型	説明
pdsname	varchar(30)	プライマリデータサーバの名前。
pdbname	varchar(30)	プライマリデータベースの名前。
rdsname	varchar(30)	レプリケートデータサーバの名前。
rdbname	varchar(30)	レプリケートデータベースの名前
from_schema	varchar(30) または NULL	プライマリデータベースのスキーマ。
to_schema	varchar(30) または NULL	レプリケートデータベースのスキーマ。
attributes	int	属性マスク: 0x01 - クラスタ化解除

インデックス

pdsname、pdbname、rdsname、rdbname、from_schema にユニーククラスタードインデックス

rs_segments

セグメントごとの割り付け情報を保持します。Replication Server は、ローディスク領域を使用して、メッセージデータを格納します。

カラム	データ型	説明
partition_id	int	パーティションのユニークな ID。
q_number	int	このパーティションが属するキュー。
q_type	int	このキューのタイプ。

カラム	データ型	説明
partition_offset	int	パーティション内でのセグメントのオフセット。
logical_seg	int	キュー内でのセグメントのオフセット。
used_flag	int	セグメントの現在のステータス。 • 0 - アクティブでない。 • 1 - アクティブ。 • *n* - セーブインターバル。*n* は、このセグメントを削除できる実際の時間 (基準時からの秒数) を示す。
version	int	セグメントの現在のバージョン。バージョン番号は使用されるたびに増加する。
flags	int	switch active 実行後に、DSI キューの最後にあるセグメントに 1 を設定する。

インデックス

partition_id、partition_offset にユニーククラスタードインデックス

rs_sites

サイトで認識されている Replication Server の名前を格納します。

カラム	データ型	説明
name	varchar(30)	Replication Server 名。
id	int	この Replication Server に割り当てられたサイト ID。
status	tinyint	未使用。

インデックス

- name にユニークインデックス
- id にユニーククラスタードインデックス

rs_statcounters

各カウンタについての情報を格納します。値は常に同じです。

カラム	データ型	説明
counter_id	int	カウンタのユニークな ID 番号。
counter_name	varchar(60)	カウンタの内容を示す名前。
module_name	varchar(30)	カウンタが所属するモジュールの名前。
display_name	varchar(30)	RCL コマンドで使用されるカウンタ名。
counter_status	int	カウンタステータス。次のいずれかのビットマスク値を示す。 • 0x001 - 内部使用。表示されない。 • 0x002 - 内部使用。表示されない。 • 0x004 - sysmon。**admin statistics, sysmon** の出力としてフラッシュされるカウンタ。 • 0x008 - 必須サンプル。常にサンプリングされるカウンタ。 • 0x010 - リセットなし。カウンタは 1 度もリセットされない。 • 0x020 - 期間。あるアクションを終了するまでの時間を通常は .01 秒単位で記録するカウンタ。 • 0x040 - 内部使用。表示されない。 • 0x080 - 古い値を維持。通常は次の監視期間中の計算で使用するために、カウンタの前の値が維持される。 • 0x100 - 内部使用。表示されない。 • 0x200 - オブザーバ。 • 0x400 - モニタ。 • 0x800 - 内部使用。表示されない。
description	varchar(255)	カウンタの説明。

インデックス

counter_id にユニーククラスタードキー rs_key_statcounters

rs_statdetail

RSSD にフラッシュされたカウンタメトリックを格納します。

カラム	データ型	説明
run_id	rs_id	実行または監視期間に割り当てられた番号。
instance_id	int	モジュールインスタンスを識別する ID。 カウンタはモジュール別にグループ化される。モジュールのインスタンスは 1 つの場合と複数の場合がある。 このカラムには、定義されたモジュール ID があればその値が格納される。たとえば DSI モジュールの場合、instance_id には DSI に関連するデータベース ID が格納される。
instance_val	int	instance_id でモジュールインスタンスを特定できない場合にモジュールを識別するための ID。
counter_id	int	カウンタのユニークな ID 番号。
counter_obs	int	監視の数。
counter_total	int	実行または監視期間に対して監視された値の合計。
counter_last	int	実行または監視期間に対して最後に監視された値。
counter_max	int	実行または監視期間に対して監視された最大値。
label	varchar(255)	データサーバやデータベース名など、カウンタに関連するモジュールインスタンスについての情報。

インデックス

run_id、instance_id、instance_val、counter_idにユニークノンクラスタードキー rs_key_statdetail

rs_statrun

各監視期間または実行に関する情報を格納します。

カラム	データ型	説明
run_id	rs_id	監視期間または実行に割り当てられた番号。
run_date	datetime	監視期間または実行の日時。
run_interval	int	監視期間または実行の秒数。
run_user	varchar(30)	RSSD にカウンタをフラッシュしたユーザの名前。
run_status	int	実行のステータス。

インデックス

run_id にユニークノンクラスタードキー rs_key_statdetail

rs_status

マテリアライゼーションの進捗に関する情報を格納します。

rs_status テーブルは、RSSD ではなく SAP IQ の各ユーザデータベースまたは HANA DB インスタンスに保存されます。 *rs_status* テーブルは、Replication Server 接続の作成時に HANA DB で作成される 6 つのオブジェクトの 1 つです。

注意： 直接ロードマテリアライゼーションの場合、Replication Server はレプリケートデータベースに rs_mat_status という名前のテーブルを作成します。 このテーブルは内部使用のみであるため、マニュアルには記載されていません。

カラム	データ型	説明
schema	SAP IQ:varchar (255) HANA DB: NVARCHAR (128)	SAP IQ: マテリアライズされるテーブルの所有者 HANA DB: スキーマ名
tablename	SAP IQ:varchar (255) HANA DB: NVARCHAR (128)	SAP IQ: マテリアライズされるテーブルの名前 HANA DB: テーブル名

カラム	データ型	説明
action	SAP IQ:varchar (1) HANA DB: NVARCHAR (1)	• I - 初回ロード • A - オートコレクションフェーズ • C - オートコレクションフェーズを終了、現在キャッチアップフェーズです (HANA DB のみ) • R - 複写
starttime	timestamp	アクションが開始された時間
endtime	timestamp	アクションが完了した時間
status	SAP IQ:varchar (1) HANA DB: NVARCHAR (1)	• P - アクションが進行中 • X - 実行が完了した • E - 実行エラー • C - 複写が完了した (HANA DB のみ)
pid	SAP IQ:int HANA DB: INTEGER	予約済み
tabletype	NVARCHAR (1)	予約済み (HANA DB のみ)
comment	NVARCHAR(1000)	予約済み (HANA DB のみ)

rs_subscriptions

サブスクリプション、トリガ、フラグメントについての情報を格納します。

カラム	データ型	説明
subname	varchar(255)	サブスクリプション、トリガ、またはフラグメントの名前。
subid	rs_id	このサブスクリプションまたはフラグメントの ID。

カラム	データ型	説明
type	int	オブジェクトタイプ。 • 0x00 - サブスクリプション • 0x01 - 範囲サブスクリプション • 0x02 - 等価サブスクリプション • 0x04 - テーブル全体 • 0x08 - パブリケーション用サブスクリプション • 0x40 - データベースサブスクリプション • 0x80 - アーティクル用サブスクリプション
objid	rs_id	このサブスクリプションのテーブル複写定義、ファンクション複写定義、アーティクル、またはパブリケーションの ID。もしくは、フラグメントまたはこのトリガのイベントの ID。
dbid	int	このオブジェクトが属するデータベースの ID。
pdbid	int	システムテーブルの複写とパブリケーション、またはアーティクルサブスクリプションでは、*pdbid* の値が複写定義に対するプライマリデータベースの ID になる。これ以外の場合には、値は 0 になる。
requestdate	datetime	最後に DDL 要求 (**create**、**drop**、**alter**) が入力された日時。
pownerid	rs_id	プライマリ Replication Server でのユーザ ID。
rownerid	rs_id	レプリケート Replication Server でのユーザ ID。

カラム	データ型	説明

status	int	• バイト 1 には、レプリケートデータベースのマテリアライゼーションステータスが格納される。
		  • 0x01 - サブスクリプションは新規。
		  • 0x02 - バルクサブスクリプションがアクティブ化中。またはアトミック／ノンアトミックサブスクリプションがマテリアライゼーションキューの作成を完了した。
		  • 0x04 - バルク／ノンアトミックサブスクリプションがアクティブ。
		  • 0x08 - バルクサブスクリプションが確定化中、またはノンアトミックサブスクリプションがマテリアライズを完了した。
		  • 0x10 - サブスクリプションが確定化済み。
		  • 0x40 - サブスクリプションがスタンバイで確定化済み。
		  • 0x80 - サブスクリプションがスタンバイで削除された。
		• バイト 2 には、プライマリデータベースのマテリアライゼーション解除ステータスが格納される。
		  • 0x100 - 新規。
		  • 0x0200 - アクティブ化中。
		  • 0x0400 - アクティブ。
		  • 0x0800 - 有効。
		• バイト 3 には、レプリケートデータベースのマテリアライゼーション解除ステータスが格納される。
		  • 0x00010000 - レプリケートでマテリアライゼーション解除中。
		  • 0x00020000 －レプリケートで削除中。
		  • 0x00100000 - プライマリでマテリアライゼーション解除中。
		• バイト 4 には、パブリケーションサブスクリプションに対するサスペクトまた

カラム	データ型	説明
		は再マテリアライゼーションのステータスが格納される。 • 0x02000000 - switch active によるサスペクト状態。 • 0x04000000 - スタンバイでの削除時にサスペクト状態。 • 0x10000000 - このパブリケーションサブスクリプション内のアーティクルサブスクリプションが 1 つずつマテリアライズ中。 • 0x20000000 - 新しいアーティクルサブスクリプションの作成段階。 • 0x40000000 - **truncate table** を含む。 • 0x80000 - サブスクリプションは削除中。 • 0x800000 - DSI はサブスクリプションに有効なマーカーを検出した。 • 0x1000 - サブスクリプションにエラーが発生した。
recovering	int	サブスクリプションのリカバリステータス。 • 0x0 - サブスクリプションは正常。 • 0x1 - リカバリ中。 • 0x2 - 保留中。
error_flag	int	設定されている場合、サブスクリプションはリカバリできない。
materializing	int	設定されている場合、サブスクリプションはマテリアライズ中である。
dematerializing	int	設定されている場合、サブスクリプションはマテリアライズ解除中である。
primary_sre	int	設定すると、サブスクリプションはプライマリ Replication Server のサブスクリプションレゾリューションエンジンに組み込まれる。

カラム	データ型	説明
replicate_sre	int	設定すると、サブスクリプションはレプリケート Replication Server のサブスクリプションレゾリューションエンジンに組み込まれる。
released	int	データベースマテリアライゼーションに関するテーブルサブスクリプションのステータス。
materializa-tion_try	int	このアトミックマテリアライゼーションが試行された回数。
method	int	サブスクリプションをマテリアライズする方法。 • 0x00 - デフォルト • 0x01 - アトミック • 0x02 - バルク • 0x04 - サスペンド • 0x08 - インクリメンタル • 0x10 - ノンアトミック • 0x80 - サスペンドされたスタンバイ DSI でのバルクマテリアライゼーション • 0x800 - **direct_load** オプションを指定して作成されたサブスクリプション **注意：** ファンクション複写定義の場合、このカラムは常に 0x02 (バルク) に設定されます。
generation	binary(4)	マテリアライゼーションキューのオリジンキュー ID に対する世代番号。
parentid	rs_id	アーティクル用サブスクリプションの場合、パブリケーションに対するそのサブスクリプションの ID。

カラム	データ型	説明
security	int	セキュリティ設定。 • 0x001 - unified_login が “required”。 • 0x002 - mutual_auth が “required”。 • 0x004 - msg_confidentiality が “required”。 • 0x08 - msg_integrity が “required”。 • 0x10 - msg_origin_check が “required”。 • 0x20 - msg_reply_detection が “required”。 • 0x40 - msg_sequence_check が “required”。
mechanism	varchar(30)	セキュリティメカニズムの名前。 デフォルト: 空の文字列
tableowner	varchar(30)	テーブル所有者名。
tablename	varchar(255)	テーブル名。
prsid	int	プライマリ SAP Replication Server ID。 rs_subscriptions システムテーブルの prsid 値は、常に rs_dbsubsets テーブルの prsid 値と同じ。
num_selects	int	直接ロードオプションで使用される選択スレッドの数。

インデックス

- subid にユニーククラスタードインデックス
- objid、dbid、subname にユニークインデックス
- subid、recovering、error_flag、materializing、dematerializing、primary_sre、replicate_sre、released にユニークインデックス
- recovering、requestdate にインデックス
- subid、status にユニークインデックス
- objid にユニークインデックス
- pdbid にユニークインデックス

rs_systext

rs_funcstrings などのさまざまなテーブルに対する、繰り返しグループのテキストを格納します。

カラム	データ型	説明
prsid	int	オブジェクトが定義されている Replication Server。
parentid	rs_id	このテキストが使用されているオブジェクトの ID。
texttype	char(1)	このローが表すオブジェクトのタイプ。 • S - ファンクション文字列の入力テンプレート • O - ファンクション文字列の出力テンプレート • C - 例外ログに記録されているトランザクション内のコマンド • P - Replication Server プロファイル
sequence	int	テキストの順番。
textval	varchar(255)	テキスト。

インデックス

parentid、texttype、sequence にユニーククラスタードインデックス

rs_targetobjs

ターゲットのテーブルまたはストアドプロシージャの情報を格納します。

カラム	データ型	説明
dbid	int	データベースのユニークな識別子。
objname	varchar(255)	テーブルまたはストアドプロシージャの名前。
objowner	varchar(30)	テーブルまたはストアドプロシージャの所有者。
objid	rs_id	オブジェクト ID。

カラム	データ型	説明
objtype	char(1)	次のいずれかのオブジェクトタイプ。 • S - ストアドプロシージャ。 • T - テーブル。
attributes	int	このカラムには以下が表示される。 • 0x01 - **rs_writetext** のカスタムファンクション文字列がある。 • 0x02 - **rs_textptr_init** のカスタムファンクション文字列がある。 • 0x04 - **rs_get_textptr** のカスタムファンクション文字列がある。

インデックス

- dbid、objname、objowner、objtype にユニーククラスタードインデックス
- objid にユニークインデックス

rs_tbconfig

Replication Server は、rs_tbconfig テーブルの情報を使用して、参照制約をサポートします。

rs_tbconfig はレプリケートシステムテーブルではありません。

カラム	データ型	説明
optionname	varchar(30)	パラメータの名前(**memory_max**、**cm_max_connections** など) パラメータの一覧とその内容を表示するには、**select *** 文を rs_tbconfig テーブルに対して実行します。
dbid	int	データベースのユニークな識別子。
objname	varchar(255)	レプリケートデータベースで定義したオブジェクト名。

カラム	データ型	説明
objowner	varchar(30)	複写定義に指定されているレプリケートオブジェクトの所有者名。 所有者が指定されていない場合は、ブランク。
charvalue	varchar(255)	パラメータの文字値。
status	tinyint	このカラムは使用されていない。
comments	varchar(255)	パラメータについての説明。

インデックス

optionname、dbid、objname、objownerにユニーククラスタードインデックス

rs_threads

Replication Server は rs_threads テーブルの情報を使ってデッドロックを検出し、並列 DSI スレッド間でトランザクションの逐次化を実行します。 このテーブルのエントリは、トランザクションが開始されたときと、コネクションに対して 2 つ以上の DSI スレッドが定義されたときに更新されます。

rs_threads テーブルは RSSD ではなく、各ユーザデータベースに保存されます。

カラム	データ型	説明
id	int	エントリ ID 番号。並列 DSI スレッドごとに、2 つのエントリがある。
seq	int	このエントリに加えられた最後の更新のシーケンス番号。コネクションが再起動されるたびにシーケンス番号は 0 で開始される。
pad1	char(255)	1 データページに 1 ローが格納されるように、ローの長さを補うためのもの。
pad2	char(255)	1 データページに 1 ローが格納されるように、ローの長さを補うためのもの。
pad3	char(255)	1 データページに 1 ローが格納されるように、ローの長さを補うためのもの。

カラム	データ型	説明
pad4	char(255)	1 データページに 1 ローが格納されるように、ローの長さを補うためのもの。

インデックス

id にユニーククラスタードインデックス

rs_ticket_history

rs_ticket の情報を格納します。

カラム	データ型	説明
cnt	int identity	チケットのユニークなシーケンス。
h1	varchar(10)	チケットのヘッダ。ヘッダが存在しない場合は “-” として設定。
h2	varchar(10)	チケットのヘッダ。ヘッダが存在しない場合は “-” として設定。
h3	varchar(10)	チケットのヘッダ。ヘッダが存在しない場合は “-” として設定。
h4	varchar(50)	チケットのヘッダ。ヘッダが存在しない場合は “-” として設定。
pdb	varchar(30)	プライマリデータベースの名前。
prs	varchar(30)	プライマリ Replication Server の名前。プライマリ Replication Server を指定しなかった場合は “-” に設定。
rrs	varchar(30)	レプリケート Replication Server の名前。
rdb	varchar(30)	レプリケートデータベースの名前。
pdb_t	datetime	**rs_ticket** ストアドプロシージャがプライマリデータベースで実行された時刻。
exec_t	datetime	チケットが Replication Server エグゼキュータスレッドをパススルーした時刻。
dist_t	datetime	チケットが DIST スレッドをパススルーした時刻。
rsi_t	datetime	チケットが RSI スレッドをパススルーした時刻。

カラム	データ型	説明
dsi_t	datetime	チケットが DSI スレッドをパススルーした時刻。
rdb_t	datetime	チケットがレプリケートデータベースに到着した時刻。
exec_b	int	EXEC スレッドが受信したバイトの総数。
rsi_b	int	RSI スレッドが受信したバイトの総数。
dsi_tnx	int	DSI スレッドが監視したトランザクションの総数。
dsi_cmd	int	DSI スレッドが監視したコマンドの総数。
ticket	varchar(1024)	ローチケット
conn_id	int	コネクション ID。

インデックス

rs_ticket_history(cnt) にユニーククラスタードインデックス rs_ticket_idx

rs_translation

クラスレベルデータ型変換についての情報を格納します。

カラム	データ型	説明
prsid	int	プライマリ Replication Server の ID。
classid	rs_id	コネクションのファンクション文字列クラス ID。
type	char(1)	変換のタイプ。値は次のとおり。 • D - クラスレベル
source_dtid	rs_id	変換元データ型の ID。
target_dtid	rs_id	変換先データ型の ID。
target_length	int	変換先データ型の値の最大長。
target_status	int	rs_columns テーブルの status カラムを参照。

カラム	データ型	説明
rowtype	tinyint	このローがローカルな Replication Server にだけ適用されるか、ドメイン内のすべての Replication Server に適用されるかを示す。値は次のとおり。 • 0 - ローカル • 1 - グローバル

インデックス

- classid、source_dtid、target_status にユニークな複合インデックス
- classid、prsid にユニークでないインデックス

rs_users

Replication Server にアクセスする各ユーザについての情報をローを格納します。

カラム	データ型	説明
username	varchar(30)	ユーザ名。
uid	rs_id	ユーザ ID。
attributes	int	アカウントとパスワードのステータスと設定 • 0x000 - デフォルト。 • 0x0001 - 最初のパスワード。 • 0x0002 - アカウントはロックされている。 • 0x0004 - ロック試行の最大失敗回数。 • 0x0008 - このログイン時のパスワードをリセットする必要がある。 • 0x0010 - パスワードに有効期限がない。 • 0x0020 - ログインは使用されない。 属性値は、ユーザと付与されるパーミッションによって異なる。 0x0000 は有効。

カラム	データ型	説明
expiration_interval	smallint	ユーザのパスワードの有効期限 (日数)。
failed_attempts	smallint	ログイン試行の失敗回数。
lock_date	datetime	アカウントがロックされた日付。
last_login	datetime	前回のログインの日付。
password	varchar(30)	パスワード。
password_date	datetime	最後にユーザのパスワードを変更した日付。
permissions	smallint	ユーザが使用できるロールを示すマスク。 • 0x0001 - **sa** • 0x0002 - **connect source** • 0x0004 - **create object** • 0x0008 - **primary subscribe**
use_enc_password	int	• 0 - 通常のパスワードを使用。 • 1 - 暗号化されたパスワードを使用。
enc_password	varchar(66)	暗号化パスワード。

インデックス

- username にユニークインデックス
- uid にユニークインデックス

rs_version

複写システムのバージョン番号を格納します。ローカル Replication Server では、ローカルのバージョン番号とシステムワイドなバージョン番号だけが格納されます。ID サーバでは、複写システム内のすべての Replication Server のバージョン情報が格納されます。

カラム	データ型	説明
siteid	int	Replication Server の ID 番号。 • 0 - システムワイドなバージョン番号のサイト ID • 1 - サイトバージョン番号のサイト ID • *n* - 個々の Replication Server のサイト ID
version	int	バージョン番号。 • 1000 - バージョン 10.0 (バージョンが不明な Replication Server に割り当てられる) • 1003 - バージョン 10.0.3 • 1011 - バージョン 10.1.1 • 1100 - バージョン 11.0 • 1101 - バージョン 10.0.1 • 1102 - バージョン 11.0.2 • 1103 - バージョン 11.0.3 • 1150 - バージョン 11.5 • 1200 - バージョン 12.0 • 1210 - バージョン 12.1 • 1250 - バージョン 12.5 • 1260 - バージョン 12.6 • 1500 - バージョン 15.0、15.0.1 • 1510 - バージョン 15.1 • 1520 - バージョン 15.2 • 1550 - バージョン 15.5 サポートされている他のバージョン番号については、『Replication Server リリースノート』を参照。

システムワイドなバージョン番号の詳細については、**「admin security_property」**を参照してください。

インデックス

siteid にユニーククラスタードインデックス

rs_whereclauses

この Replication Server で認識されているアーティクルで使用されている **where** 句についての情報を格納します。

カラム	データ型	説明
articleid	rs_id	この **where** 句に含まれるアーティクルの ID。
wclauseid	rs_id	この **where** 句の ID。
type	int	• 0x01 - 範囲 • 0x02 - 等価

インデックス

wclauseid にユニーククラスタードインデックス

頭文字と略語

Replication Server のメッセージやマニュアルで使用される頭文字と略語をリストします。

用語の定義については、『Replication Server 管理ガイド 第 2 巻』の用語解説を参照してください。

表 58 : 頭文字のリスト

頭文字	意味
APC	Asynchronous Procedure Call (非同期プロシージャコール)
API	Application Program Interface (アプリケーションプログラミングインタフェース)
BM	Bitmap (ビットマップ)
C/SI	Client/Server Interfaces
CM	Connection Manager (コネクションマネージャ)
dAIO	Asynchronous I/O Daemon (非同期 I/O デーモン)
dALARM	Alarm Daemon (アラームデーモン)
DBO	Database Owner (データベース所有者)
dCM	Connection Manager Daemon (コネクションマネージャデーモン)
DDL	データ定義言語
DIST	ディストリビュータ
DML	データ操作言語
dREC	Recovery Daemon (リカバリデーモン)
DSI	Data Server Interface (データサーバインタフェース)
dSUB	Subscription Retry Daemon (サブスクリプションリトライデーモン)
ELM	Exceptions Log Manager (例外ログマネージャ)
ERSSD	Embedded Replication Server システムデータベース
EXC	Exception (例外)
EXEC	Executor (エグゼキュータ)

頭文字	意味
FSTR	Function String (ファンクション文字列)
HDS	Heterogeneous datatype support (異機種データ型サポート)
HTS	Hash Table (ハッシュテーブル)
LAN	Local Area Network (ローカルエリアネットワーク)
LL	Linked List (リンクリスト)
LTI	Log Transfer Interface (ログ転送インタフェース)
LTL	Log Transfer Language (ログ転送言語)
MD	Message Delivery (メッセージデリバリ)
MEM	Memory Management (メモリ管理)
MP	Multiprocessor (マルチプロセッサ)
MSA	Multi-Site Availability (マルチサイト可用性)
NRM	Normalization (正規化)
OQID	Origin Queue ID (オリジンキュー ID)
PDS	Primary Data Server (プライマリデータサーバ)
PRS	Primary Replication Server (プライマリ Replication Server)
PRS	Parser (パーサ)
QID	Queue ID (キュー ID)
RA	Replication Agent
RCL	Replication Command Language (複写コマンド言語)
RDS	Replicate Data Server (レプリケートデータサーバ)
REP AGENT	RepAgent スレッド、Adaptive Server 用の Replication Agent
RMI	Remote Method Invocation
RPC	Remote Procedure Call (リモートプロシージャコール)
RRS	Replicate Replication Server (レプリケート Replication Server)
RS	Replication Server
RSI	Replication Server Interface (Replication Server インタフェース)
RSP	Replicated Stored Procedure (複写ストアドプロシージャ)

頭文字	意味
RSA	Replication System Administrator (複写システム管理者)
RSI	Replication Server Interface (Replication Server インタフェース)
RSSD	Replication Server System Database (Replication Server システムデータベース)
SA	System Administrator (システム管理者)
SP	Stored Procedure (ストアドプロシージャ)
SQM	Stable Queue Manager (ステーブルキューマネージャ)
SQT	Stable Queue Transaction Interface (ステーブルキュートランザクションインタフェース)
SRE	Subscription Resolution Engine (サブスクリプションレゾルーションエンジン)
STS	System Table Services (システムテーブルサービス)
SUB	Subscription (サブスクリプション)
TD	Transaction Delivery (トランザクションデリバリ)
TDS	Tabular Data Stream™
WAN	Wide Area Network (広域ネットワーク)

SAP Replication Server のデザイン制限

では、さまざまな複写システムオブジェクト用の最大パラメータと最小パラメータについて説明します。

Replication Server の制限値

*For_Life_Of*変数は、1つの Replication Server で作成できるオブジェクトの総数を表します。途中でオブジェクトを削除してもその数は変わりません。

たとえば、制限値が 100,000 のときに 100,000 個のオブジェクトを作成した場合は、その中の一部またはすべてを削除しても、それ以上オブジェクトを作成できません。*For_Life_Of* カウントと制限値は、Replication Server のソフトウェアがインストールされているかぎり効力があります。*For_Life_Of* カウントを再起動するには、サーバ全体をシステムから削除し、再インストールします。

表 59 : Replication Server の制限値

オブジェクトの種類	数値
Replication Server の複写定義数 (*For_Life_Of*)	2^{24} (16,777,216)
Replication Server のユーザ数 (*For_Life_Of*)	2^{24} (16,777,216)
Replication Server の reject log コマンド数 (*For_Life_Of*)	2^{32} から 2^{29} (3,758,096,384)
Replication Server の reject log トランザクション数 (*For_Life_Of*)	2^{31} (2,147,483,648)
ID サーバあたりの Replication Server 数	2^{24} (16,777,216)
ID サーバあたりのデータベース数	2^{24} (16,777,216)
Replication Server あたりのデータベース数	2^{24} (16,777,216)
Replication Server のパーティション数 (*For_Life_Of*)	2^{16} (65,536)
初期パーティション (RS インストール用) の最小サイズ	20MB
追加パーティションの最小サイズ	1MB
パーティションの最大サイズ	1TB
Replication Server あたりのステーブルキュー数	2^{64}
Replication Server のサブスクリプション数 (*For_Life_Of*)	2^{31}

オブジェクトの種類	数値
Replication Server あたりのコネクション数 • 受信 (Replication Agent、DIST、RS、ユーザ) • 発信 (DSI、ルート)	2^{24} -1 2^{32} -1
Replication Server のファンクション文字列数 (*For_Life_Of*)	2^{24} (16,777,216)
Replication Server のエラークラス数 (*For_Life_Of*)	2^{24} (16,777,216)

プラットフォーム固有の制限値

プロセスごとのファイル記述子などのプラットフォームオペレーティングシステム固有の特定の制限値について説明します。これは Replication Server に影響を与える場合があります。

固有の制限値については、プラットフォームのリリースノートを参照してください。

複写定義とサブスクリプションの制限値

複写定義とサブスクリプションの制限値について説明します。

表 60 : 複写定義の制限値

オブジェクトの種類	数値
複写定義あたりのカラム数	1024 に制限される。
複写定義あたりのプライマリキーカラム数	複写定義内に指定されたカラム数に制限される。
複写定義あたりのサーチャブルカラム数	複写定義内に指定されたカラム数に制限される。
複写定義あたりのサブスクリプション数	制限なし。
サブスクリプションの **where** 句の文字列幅	255 バイトに制限される。

ファンクション文字列の制限値

Replication Server のファンクション文字列の制限値について説明します。

表 61 : ファンクション文字列の制限値

オブジェクトの種類	数値
ファンクション文字列クラスあたりのファンクション文字列	制限なし。
言語タイプのファンクション文字列テンプレートあたりのバイト数	64K
変数値代入後の言語タイプのファンクション文字列テンプレートあたりのバイト数	64K マイナス 1 バイト
ファンクション文字列あたりの埋め込み変数の数	制限なし。
ファンクション文字列入力テンプレート内のユーザ変数の数	1024

プログラミングの制限とパラメータ

プログラミングの制限とパラメータについて説明します。

表 62 : プログラミングの制限とパラメータ

オブジェクトの種類	数値
サブスクリプションの **where** 句内の項数	制限なし。
DSI トランザクショングループ内のトランザクション数	20
DSI コマンドバッチ内のソースコマンド数	50
Replication Server が処理する各コマンドのバイト数	16K
エラークラスあたりのアクション割り当て数	2^{31} (2,1474,836,448)
ステーブルキューに書き込まれるメッセージの最大サイズ	制限なし。

用語解説

複写システムで使用される用語を解説します。

- **アクティブデータベース** – ウォームスタンバイアプリケーションのスタンバイデータベースに複写されるデータベースです。「*ウォームスタンバイアプリケーション*」も参照してください。
- **アプリケーションプログラミングインタフェース (API)** – ユーザまたはプログラムが相互に通信するために使用する、事前に定義されたインタフェースです。 Open Client™ および SAP Open Server は、クライアント/サーバアーキテクチャで通信を行う API の 1 つです。 RCL (Replication Command Language) は、SAP Replication Server の API です。
- **適用ファンクション** – ファンクション複写定義に対応する複写ファンクションであり、SAP Replication Server によってプライマリデータベースからサブスクライブ元のレプリケートデータベースに配信されます。「*複写ファンクションの配信*」、「*要求ファンクション*」、「*ファンクション複写定義*」も参照してください。
- **アーティクル** – テーブルまたはストアドプロシージャの複写定義を拡張したもので、パブリケーションの要素となります。アーティクルには、レプリケートデータベースが受信するローのサブセットを指定した **where** 句が含まれている場合もあれば、含まれていない場合もあります。
- **非同期プロシージャ配信** – プライマリデータベースまたはレプリケートデータベースで複写するように指定されたストアドプロシージャを実行できる Replication Server システムの一部です。
- **非同期コマンド** – クライアントが SAP Replication Server に送信するコマンドです。クライアントは、完了ステータスの受信を待たずに、他のオペレーションを継続できます。 SAP Replication Server のコマンドの多くは、複写システム内で非同期コマンドとして動作します。
- **アトミックマテリアライゼーション** – マテリアライゼーションメソッドの 1 つです。select オペレーションを holdlock を指定して使用し、1 つのアトミックオペレーションでネットワークを介して、プライマリデータベースからレプリケートデータベースへサブスクリプションデータをコピーします。 データの転送が完了するまで、プライマリデータへの変更は行えません。「*ノンアトミックマテリアライゼーション*」、「*バルクマテリアライゼーション*」、「*非マテリアライゼーション*」も参照してください。
- **オートコレクション** – オートコレクションは、複写定義に適用する機能で、レプリケートテーブルに、消失ローや重複したローが発生して障害が起こることを防ぎます。**set autocorrection** コマンドを使用して設定します。 オートコレクションを有効にすると、SAP Replication Server は各更新オペレーションまたは

挿入オペレーションを削除と挿入の連続オペレーションに変換します。 オートコレクションは、サブスクリプションがノンアトミックマテリアライゼーションを使用している複写定義についてのみ有効にしてください。

- **基本クラス** – 親クラスからファンクション文字列を継承しないファンクション文字列クラスです。「*ファンクション文字列クラス*」も参照してください。
- **ビットマップサブスクリプション** – ビットマップの比較に基づいてローを複写するサブスクリプションの種類です。 int データ型のカラムを作成し、複写定義を作成するときには、カラムを `rs_address` データ型として指定します。
- **バルクコピーイン** – SAP ASE 12.0 以降で、大量の **insert** 文を同じテーブルで複写するときに SAP Replication Server のパフォーマンスを向上させる機能です。SAP Replication Server は、Open Client™ Open Server™ Bulk-Library を使用して、レプリケートデータベースにトランザクションを送信する SAP Replication Server モジュールであるデータサーバインタフェース (DSI) にバルクコピーインを実装します。

  バルクコピーインにより、サブスクリプションマテリアライゼーションのパフォーマンスも向上します。 **dsi_bulk_copy** を on にすると、各トランザクションの **insert** コマンドの数が **dsi_bulk_threshold** を超えた場合に、SAP Replication Server は、バルクコピーインを使用してサブスクリプションをマテリアライズします。
- **バルクマテリアライゼーション** – マテリアライゼーションのメソッドの 1 つです。これは、複写システム以外でレプリケートデータベースのサブスクリプションのデータを初期化します。 バルクマテリアライゼーションは、テーブル複写定義とファンクション複写定義のどちらのサブスクリプションにも使用できます。 たとえば、磁気テープ、フロッピーディスク、CD-ROM、または光磁気ディスクなどのメディアを使用して、プライマリデータベースからデータを転送できます。 バルクマテリアライゼーションでは、define subscription から始まる一連のコマンドを使用します。「*アトミックマテリアライゼーション*」、「*ノンアトミックマテリアライゼーション*」、「*非マテリアライゼーション*」も参照してください。
- **集中型データベースシステム** – 中央サイトに設置された 1 つのデータベース管理システムでデータを一元管理するデータベースシステムです。
- **クラス** –「*エラークラス*」と「*ファンクション文字列クラス*」を参照してください。
- **クラスツリー** – 派生クラスと親クラスの複数のレベルから構成されるファンクション文字列クラスのセットです。これは、同じ基本クラスから派生します。「*ファンクション文字列クラス*」も参照してください。
- **クライアント** – クライアント/サーバアーキテクチャにおいて、サーバに接続されたプログラムです。 ユーザが実行するフロントエンドアプリケーションプログラムの場合もあれば、システムの拡張機能として実行されるユーティリティプログラムの場合もあります。

- **Client/Server Interfaces (C/SI) –** クライアント/サーバアーキテクチャで実行するプログラムのための SAP のインタフェース標準です。
- **同時実行性 –** 複数のクライアントが、データまたはリソースを共有できることを示します。データベース管理システムにおける同時実行性は、あるクライアントが使用中のデータを別のクライアントが変更しようとするときに発生する競合からクライアントを保護するシステムに依存します。
- **接続 –** SAP Replication Server からデータベースへの接続です。「*データサーバインタフェース (DSI)*」と「*論理接続*」も参照してください。
- **接続プロファイル –** データベース接続を確立するために必要な情報です。
- **コーディネートダンプ –** 複写システムでファンクション rs_dumpdb または rs_dumptran を実行することによって、複数のサイト間で同期がとられているデータベースダンプセット、またはトランザクションダンプセットです。
- **データベース –** 相互に関連するデータテーブルとその他のオブジェクトが、特定の目的に合わせて編成、表現されたものです。
- **データベース世代番号 –** データベースおよびデータベースを管理する SAP Replication Server の RSSD に格納されます。データベース世代番号は、各ログレコードのオリジンキュー ID (*qid*) の最初の部分です。 オリジンキュー ID は、SAP Replication Server が重複したレコードを処理していないことを示します。リカバリオペレーション時には、データベースの再ロード後に送信されたレコードが SAP Replication Server に無視されないようにするために、データベース世代番号を増やす必要がある場合があります。
- **データベース複写定義 –** サブスクリプションを作成できる対象のデータベースオブジェクト (テーブル、トランザクション、ファンクション、システムストアドプロシージャ、DDL) を集めて記述したものです。

  テーブル複写定義とファンクション複写定義も作成できます。「*テーブル複写定義*」と「*ファンクション複写定義*」も参照してください。
- **データベースサーバ –** SAP ASE などのサーバアプリケーションです。クライアントにデータベース管理サービスを提供します。
- **データ定義言語 (DDL) –** Transact-SQL などのクエリ言語のコマンドセットであり、データとデータベース内でのデータの関係を記述します。Transact-SQL の DDL コマンドには、**create**、**drop**、**alter** キーワードを使用するものなどがあります。
- **データ操作言語 (DML) –** Transact-SQL などのクエリ言語のコマンドセットであり、データの操作を行います。Transact-SQL の DML コマンドには、**select**、**insert**、**update**、**delete** があります。
- **データサーバ –** SAP クライアント/サーバインタフェースに準拠のクライアントインタフェースによって、データベースのレプリケートテーブルの物理表現を管理するのに必要な機能を提供するサーバです。 データサーバは、通常、

データベースサーバと同じものですが、SAP Replication Server に必要なインタフェースと機能を備えたデータリポジトリの場合もあります。

- **データサーバインタフェース (DSI)** – SAP Replication Server とデータベース間のコネクションに対応している SAP Replication Server スレッドです。 DSI スレッドは、DSI アウトバウンドキューからレプリケートデータサーバへトランザクションを送信します。 DSI スレッドは 1 つのスケジューラスレッドと 1 つまたは複数のエグゼキュータスレッドで構成されます。 スケジューラスレッドは、トランザクションをコミット順にグループ分けしてから、それらをエグゼキュータスレッドにディスパッチします。 エグゼキュータスレッドは、ファンクションをファンクション文字列にマッピングし、レプリケートデータベースのトランザクションを実行します。 DSI スレッドは、データベースへの Open Client コネクションを使用します。「*アウトバウンドキュー*」と「*コネクション*」も参照してください。
- **データソース** – リレーショナルデータサーバまたはノンリレーショナルデータサーバなどのデータベース管理システム (DBMS) 製品、その DBMS にあるデータベース、および複写システムの他のコンポーネントから DBMS にアクセスするための通信方法の組み合わせからなる概念をデータソースといいます。「*データベース*」と「*データサーバ*」も参照してください。
- **意思決定支援アプリケーション** – アドホッククエリ、レポート、計算などの処理が行え、少数データの更新トランザクションを特徴とするデータベースクライアントアプリケーションです。
- **宣言したデータ型** – Replication Agent から SAP Replication Server に配信された値のデータ型です。
  - Replication Agent が `datetime` などの基本 SAP Replication Server データ型を SAP Replication Server に配信する場合、宣言したデータ型は基本データ型です。
  - 上記以外の場合、宣言したデータ型は、プライマリデータベースの元のデータ型に対する UDD でなければなりません。
- **デフォルトのファンクション文字列** – システム提供クラス `rs_sqlserver_function_class` と `rs_default_function_class`、およびこれらのクラスからファンクション文字列を直接的または間接的に継承するクラスに対してデフォルトで提供されるファンクション文字列です。「*ファンクション文字列*」も参照してください。
- **マテリアライゼーション解除** – サブスクリプションが削除されたときに、適宜実行される処理です。これによって、他のサブスクリプションに使用されていない特定のローが、レプリケートデータベースから削除されます。
- **派生クラス** – 親クラスからファンクション文字列を継承するファンクション文字列クラスです。「*ファンクション文字列クラス*」と「*親クラス*」も参照してください。

- **直接ルート** – 中間 SAP Replication Server を使用せずに、送信元 SAP Replication Server から送信先 Replication Server へ直接メッセージを送信するために使用するルートです。「*間接ルート*」と「*ルート*」も参照してください。
- **ディスクパーティション** –「*パーティション*」を参照してください。
- **分散データベースシステム** – データをネットワーク上にある複数のデータベースに格納するデータベースシステムです。
- **ディストリビュータ** – インバウンドキューにある各トランザクションの送信先を決定するために使用する SAP Replication Server スレッド (DIST) です。
- **ダンプマーカ** – ダンプの実行時に SAP ASE がデータベーストランザクションログに書き込むメッセージです。 ウォームスタンバイアプリケーションでは、アクティブデータベースのデータでスタンバイデータベースを初期化するときに、SAP Replication Server がダンプマーカを使用して、トランザクションストリームのどこからトランザクションをスタンバイデータベースに適用するかを決定するように指定できます。「*ウォームスタンバイアプリケーション*」も参照してください。
- **Embedded Replication Server システムデータベース (ERSSD)** – SAP Replication Server システムテーブルを格納する SAP SQL Anywhere データベースです。 SAP Replication Server システムテーブルを ERSSD と SAP ASE RSSD のどちらに格納するかを選択できます。「*Replication Server システムデータベース (RSSD)*」も参照してください。
- **Enterprise Connect Data Access (ECDA)** – LAN ベースの ASE 以外のデータソースやメインフレームのデータソースなど、異機種データベース環境にあるデータへのアクセスを可能にするソフトウェアアプリケーションと接続ツールを統合したセットです。
- **エラーアクション** – データサーバのエラーに対する SAP Replication Server の応答処理です。 SAP Replication Server のエラーアクションの種類としては、**ignore**、**warn**、**retry_log**、**log**、**retry_stop**、**stop_replication** があります。 エラーアクションは、特定のデータサーバエラーに割り当てられます。
- **エラークラス** – 指定したデータベースで使用するデータサーバのエラーアクションの集まりに名前を付けたものです。
- **例外ログ** – データサーバ上で失敗したトランザクションの情報を保存する SAP Replication Server の 3 つのシステムテーブルがセットになったものです。 例外ログに記録されたトランザクションは、ユーザまたはインテリジェントアプリケーションが処理しなければなりません。 例外ログのクエリを行うには、**rs_helpexception** ストアドプロシージャを使用します。
- **ExpressConnect for HANA データベース** – SAP Replication Server と SAP HANA データベース間の直接通信に使用できるライブラリのセットです。
- **ExpressConnect for Oracle** – SAP Replication Server と Oracle データベース間の直接通信に使用できるライブラリのセットです。

- **フェールオーバ** – SAP フェールオーバを使用すると、バージョン 12.0 以降の 2 つの SAP ASE をコンパニオンとして設定できます。 プライマリコンパニオンに障害が発生した場合、そのサーバのデバイス、データベース、コネクションをセカンダリコンパニオンが引き継ぐことができます。

  SAP ASE における SAP フェールオーバの動作の詳細については、『高可用性システムにおける SAP フェールオーバの使用』を参照してください。これは、SAP ASE のマニュアルセットの一部です。
- **フォールトトレランス** – 1 つまたは複数のコンポーネントで障害が発生した場合でも、システムが正常に処理を継続できるシステムの機能です。
- **ファンクション** – insert、delete、select、begin transaction などのデータサーバのオペレーションを表す SAP Replication Server オブジェクトです。 SAP Replication Server は、これらのオペレーションをファンクションとして他の SAP Replication Server に配信します。 各ファンクションは、ファンクション名とデータパラメータのセットから構成されます。 ファンクションを送信先データベースで実行するために、SAP Replication Server はファンクション文字列を使用して、そのファンクションを特定タイプのデータベース用のコマンドまたはコマンドセットに変換します。 「*ユーザ定義ファンクション*」と「*複写ファンクションの配信*」も参照してください。
- **ファンクション複写定義** – 複写ファンクションの配信で使用される、複写ファンクションの記述です。 ファンクション複写定義は SAP Replication Server によって管理され、複写されるパラメータや、影響を受けるデータのプライマリバージョンがあるロケーションに関する情報などからなります。 「*複写ファンクションの配信*」も参照してください。
- **ファンクションのスコープ** – ファンクションの適用範囲です。ファンクションは、複写定義スコープまたはファンクション文字列クラススコープを持ちます。複写定義スコープを持つファンクションは、特定の複写定義に指定され、他の複写定義には適用できません。ファンクション文字列クラススコープを持つファンクションは、ファンクション文字列クラスに対して 1 回だけ定義され、そのクラスでのみ使用できます。
- **ファンクション文字列** – SAP Replication Server が、ファンクションとそのパラメータをデータサーバの API にマップするために使用する文字列です。 ファンクション文字列により、SAP Replication Server は、プライマリデータベースとレプリケートデータベースでタイプが異なる異機種間複写を、異なる SQL 拡張機能とコマンド機能でサポートできます。
- **ファンクション文字列クラス** – 指定したデータベース接続で使用される、名前付きのファンクション文字列のコレクションです。 ファンクション文字列クラスには、SAP Replication Server によって提供されるものとユーザが作成したものがあります。 ファンクション文字列クラスは、ファンクション文字列継承によってファンクション文字列定義を共有できます。 システムで提供されるファンクション文字列クラスには、`rs_sqlserver_function_class`、

rs_default_function_class、rs_db2_function_class の 3 つがあります。「*基本クラス*」、「*クラスツリー*」、「*派生クラス*」、「*ファンクション文字列継承*」、「*親クラス*」も参照してください。

- **ファンクション文字列の継承** – クラス間でファンクション文字列定義を共有する機能です。この機能によって、派生クラスは親クラスからファンクション文字列を継承します。「*派生クラス*」、「*ファンクション文字列クラス*」、「*親クラス*」も参照してください。
- **ファンクション文字列変数** – 実行時に代入される値を表すために、ファンクション文字列内で使用する識別子です。ファンクション文字列内の変数は、疑問符 (?) で囲まれています。この変数は、カラムの値、ファンクションのパラメータ、システム定義の変数、ユーザ定義の変数を表します。
- **ファンクションサブスクリプション** – ファンクション複写定義に対するサブスクリプションです (適用ファンクションおよび要求ファンクションの配信で使用されます)。
- **ゲートウェイ** – 異なるネットワークアーキテクチャを持つ複数のコンピュータシステム間での通信を可能にする接続ソフトウェアです。
- **世代番号** – 「*データベース生成番号*」を参照してください。
- **異機種データサーバ** – 同じ分散データベースシステム内で使用される複数ベンダのデータサーバです。
- **ハイバネーションモード** – SAP Replication Server の状態です。この状態では、admin と sysadmin コマンドを除くすべてのデータ定義言語 (DDL) コマンドは拒否され、すべてのルートとコネクション、およびデータサーバインタフェース (DSI)、SAP Replication Server インタフェース (RSI) などのほとんどのサービススレッドがサスペンドされます。また、RSI と Replication Agent ユーザはログオフされログオンできません。 これは、ルートのアップグレード中に使用され、SAP Replication Server が問題をデバッグするためにオン状態になることがあります。
- **High-Performance Analytic Appliance (HANA)** – SAP® インメモリオンライントランザクション処理/オンライン分析処理ソリューション。
- **High-Performance Analytic Appliance データベース (SAP HANA データベース)** – SAP インメモリデータベース。
- **高可用性 (HA)** – ダウン時間が非常に少ないことです。HA を提供するコンピュータシステムは、通常、99.999% の可用性 (予定外のダウン時間が、年間約 5 分) を実現しています。
- **High Volume Adaptive Replication (HVAR)** – 最終的な結果とそれ以降のレプリケートデータベースへの最終的な結果のバルク適用を生成する、**insert**、**delete**、**update** の各オペレーションのグループのコンパイルです。
- **ホットスタンバイアプリケーション** – クライアントアプリケーションを中断したり、トランザクションを失ったりすることなく、スタンバイデータベースを

アクティブに切り替えられるデータベースアプリケーションです。「*ウォームスタンバイアプリケーション*」も参照してください。

- **ID サーバ** – 複写システムのいずれか 1 つの SAP Replication Server が、ID サーバとなります。 ID サーバは、SAP Replication Server の通常の作業の他に、複写システムにあるすべての SAP Replication Server とデータベースにユニークな ID 番号を割り当て、複写システムのバージョン情報を管理します。
- **インバウンドキュー** – Replication Agent から SAP Replication Server へのメッセージをスプールするために使用されるステーブルキューです。
- **間接ルート** – 送信元 SAP Replication Server から送信先 SAP Replication Server へ、1 つ以上の中間 SAP Replication Server を経由してメッセージを送るために使用するルートです。「*直接ルート*」と「*ルート*」も参照してください。
- **interfaces ファイル** – SAP クライアント/サーバアーキテクチャ上のサーバプログラムが使用する、ネットワークのアクセス情報を定義するエントリのあるファイルです。 サーバプログラムには、SAP ASE、ゲートウェイ、SAP Replication Server、Replication Agent があります。 クライアントとサーバは、interfaces ファイルにあるエントリを使用して、ネットワーク上で相互に接続できます。
- **遅延時間** – プライマリデータベースで最初に適用されたデータ修正オペレーションが、レプリケートデータベースに分配されるまでに要する時間の単位です。 この時間には、Replication Agent での処理時間、SAP Replication Server での処理時間、ネットワークのオーバヘッドなどが含まれます。
- **ローカルエリアネットワーク (LAN)** – コンピュータとプリンタや端末などのデバイスを、データやデバイスの共有のためにケーブルで接続したシステムです。
- **ロケータ値** – SAP Replication Server の RSSD の `rs_locater` テーブルに格納されている値です。この値によって、複写中に SAP Replication Server によって受信および確認された、直前の各サイトからの最新のログトランザクションレコードが特定されます。
- **論理コネクション** – SAP Replication Server が、ウォームスタンバイアプリケーションのアクティブデータベースとスタンバイデータベースとのコネクションにマップするデータベース接続です。「*コネクション*」と「*ウォームスタンバイアプリケーション*」も参照してください。
- **ログイン名** – ユーザまたは SAP Replication Server などのシステムコンポーネントがデータサーバ、SAP Replication Server、または Replication Agent にログインするために使用する名前です。
- **ログ転送言語 (LTL)** – Replication Command Language (RCL) のサブセットです。プライマリデータベースのトランザクションログから取得した情報は、RepAgent などの Replication Agent によって、LTL コマンドを使用して、SAP Replication Server に送信されます。

- **Log Transfer Manager (LTM)** – SAP SQL Server 用の Replication Agent プログラムです。「*Replication Agent*」と「*RepAgent スレッド*」も参照してください。
- **メンテナンスユーザ** – SAP Replication Server がレプリケートデータを管理するために使用するデータサーバのログイン名です。 ほとんどのアプリケーションでは、メンテナンスユーザのトランザクションは複写されません。
- **マテリアライゼーション** – プライマリデータベースからレプリケートデータベースへ、サブスクリプションによって指定されたデータをコピーする処理です。これによって、レプリケートテーブルが初期化されます。レプリケートデータはネットワークを介して転送するか、またはサブスクリプションが大量のデータを扱う場合は、メディアからロードできます。「*アトミックマテリアライゼーション*」、「*バルクマテリアライゼーション*」、「*非マテリアライゼーション*」、「*ノンアトミックマテリアライゼーション*」も参照してください。
- **マテリアライゼーションキュー** – マテリアライゼーションまたはマテリアライゼーション解除されているサブスクリプションに関連したメッセージをスプールするために使用されるステーブルキューです。
- **消失ロー** – プライマリテーブルには存在するが、そのテーブルの複写コピーには存在しないローです。
- **混合バージョンシステム** – ソフトウェアバージョンとサイトバージョンの違いによって異なる機能を持った、ソフトウェアバージョンの異なる SAP Replication Server がある複写システムです。 混合バージョンサポートは、システムバージョンが 11.0.2 以降の場合のみ使用できます。

  たとえば、SAP Replication Server バージョン 11.5 以降とバージョン 11.0.2 がある複写システムは、混合バージョンシステムです。 バージョン 11.0.2 以降の SAP Replication Server では、特定の新機能の使用がシステムバージョンによって制限されていますが、それより前のバージョンではこの機能がサポートされていないため、バージョン 11.0.2 より前の SAP Replication Server を持った複写システムは、混合バージョンシステムではありません。 「*サイトバージョン*」と「*システムバージョン*」も参照してください。
- **カラム数の増加** – 250 を超え、最大 1024 までの複写定義内のカラム数の増加のことです。 カラム数の増加は、SAP Replication Server バージョン 12.5 以降でサポートされています。
- **Multi-Site Availability (MSA)** – テーブル、ファンクション、トランザクション、システムストアドプロシージャ、データ定義言語 (DDL) 文などのデータベースオブジェクトをプライマリデータベースからレプリケートデータベースへ複写する方法です。 「*データベース複写定義*」も参照してください。
- **Multi-Path Replication™** – 送信元データベースからターゲットデータベースへのデータの並列パスを有効にすることによってパフォーマンスを向上させる SAP Replication Server の機能です。 Multi-Path Replication™ は、ウォームスタンバイ環境と Multi-Site Availability (MSA) 環境で設定できます。 これらの複数のパスではデータが個別に処理され、それらのパス間のトランザクションの一貫

性を必要とせずにデータセットを並列処理できる場合に適用されます。パス内でのデータ整合性を維持しますが、さまざまなパス間でのコミット順には従いません。

- **ネームスペース** – オブジェクト名がユニークでなければならない範囲 (スコープ) です。
- **ノンアトミックマテリアライゼーション** – マテリアライゼーションのメソッドの 1 つです。これは、holdlock を使わずに 1 つのオペレーションで、ネットワークを介してプライマリデータベースからレプリケートデータベースへサブスクリプションデータをコピーします。 データの転送中もプライマリテーブルを変更できるので、レプリケートデータベースとプライマリデータベース間で一時的に不一致が生じる可能性があります。 データは、レプリケートデータベースのトランザクションログが満杯にならないように、トランザクションごとに 10 ローずつ挿入する方法を使用して適用されます。 ノンアトミックマテリアライゼーションは、**create subscription** コマンドのオプションのメソッドです。「*オートコレクション*」、「*アトミックマテリアライゼーション*」、「*非マテリアライゼーション*」、「*バルクマテリアライゼーション*」も参照してください。
- **ネットワークベースセキュリティ** – ネットワーク上でのデータの安全な転送です。 SAP Replication Server は、ユーザの認証、統一化ログイン、SAP Replication Server 間の安全なメッセージ転送などのサードパーティのセキュリティメカニズムをサポートします。
- **非マテリアライゼーション** – マテリアライゼーションのメソッドの 1 つです。サブスクリプションデータがレプリケートサイトにすでに存在する場合、サブスクリプションを作成できます。 **without materialization** 句を指定して **create subscription** コマンドを使用してください。 このメソッドを使用して、テーブル複写定義のサブスクリプションを作成することもできます。「*アトミックマテリアライゼーション*」と「*バルクマテリアライゼーション*」も参照してください。
- **オンライントランザクション処理 (OLTP) アプリケーション** – データ修正 (挿入、削除、更新) を伴うさまざまなトランザクションを頻繁に実行するデータベースクライアントアプリケーションです。
- **オリジンキュー ID (qid)** – qid は、Replication Agent によって形成され、SAP Replication Server に渡された各ログレコードをユニークに識別します。 日付、タイムスタンプ、およびデータベース世代番号が含まれます。「*データベース世代番号*」も参照してください。
- **孤立したロー** – レプリケートデータベースにあって、プライマリデータベースにはないテーブルローです。
- **アウトバウンドキュー** – メッセージをスプールするのに使用するステーブルキューです。 DSI アウトバウンドキューは、レプリケートデータベースへの

メッセージをスプールします。 RSI アウトバウンドキューは、レプリケート SAP Replication Server へのメッセージをスプールします。

- **並列 DSI** – 単一のデータサーバインタフェース (DSI) スレッドではなく、並列で機能する複数の DSI スレッドを使用して、レプリケートデータサーバにトランザクションが適用されるようにデータベース接続を設定する方法です。「*コネクション*」と「*データサーバインタフェース (DSI)*」も参照してください。
- **パラメータ** – プロシージャの実行時に提供される値を表す識別子です。 ファンクション文字列で使用するパラメータ名は @ 記号で始まります。 プロシージャをファンクション文字列から呼び出すと、SAP Replication Server はパラメータ値をそのまま変更しないでデータサーバへ渡します。 「*サーチャブルパラメータ*」も参照してください。
- **親クラス** – 派生クラスがファンクション文字列を継承する、ファンクション文字列クラスです。「*ファンクション文字列クラス*」と「*派生クラス*」も参照してください。
- **パーティション** – SAP Replication Server が、ステーブルキューを格納するために使用するローディスクパーティションまたはオペレーティングシステムファイルです。 オペレーティングシステムファイルはテスト環境でのみ使用してください。
- **物理コネクション** – SAP Replication Server からデータベースへの接続です。
- **プライマリデータ** – 複写システム内で最も信頼できるデータセットのバージョンです。 プライマリデータは、データサーバによって管理されます。このデータサーバは、データのサブスクリプションがあるすべての SAP Replication Server で認識されています。
- **プライマリデータベース** – 複写システムによって別のデータベースに複写されるデータが格納されたデータベースです。
- **プライマリフラグメント** – 一連のローのプライマリバージョンを保持するテーブルの水平方向セグメントです。
- **プライマリキー** – 各ローをユニークに識別するテーブルカラムのセットです。
- **プライマリサイト** – 通常のビジネスオペレーションをサポートするために、プライマリデータサーバおよびプライマリデータベースが展開されるロケーションまたは環境です。 アクティブサイトまたはメインサイトとも呼ばれます。「*エラークラス*」と「*ファンクション文字列クラス*」を参照してください。
- **プリンシパルユーザ** – アプリケーションを開始するユーザです。 ネットワークベースのセキュリティを使用する場合、SAP Replication Server はプリンシパルユーザとしてリモートサーバにログインします。
- **プロファイル** – SAP Replication Server が接続するサーバに関する事前定義済みのプロパティセットにより接続を設定できます。

- **射影** – テーブルの垂直方向のスライスです。テーブルカラムのサブセットを表します。
- **パブリケーション** – 同じプライマリデータベースからのアーティクルのグループです。 パブリケーションを使用すると、関連するテーブルかストアドプロシージャまたはその両方の複写定義を収集して、グループとしてそれらをサブスクライブできます。 送信元 SAP Replication Server のパブリケーション内でアーティクルとして複写定義を収集し、送信先 SAP Replication Server でパブリケーションサブスクリプションを使用してそれらをサブスクライブできます。「*アーティクル*」と「*パブリケーションサブスクリプション*」も参照してください。
- **パブリケーションサブスクリプション** – パブリケーションへのサブスクリプションです。「*アーティクル*」と「*パブリケーション*」も参照してください。
- **パブリッシュデータ型** – レプリケートデータサーバにおけるカラムレベル変換後 (続いてクラスレベル変換をする場合はその前) のカラムのデータ型です。 パブリッシュデータ型は、SAP Replication Server 基本データ型か、ターゲットデータサーバのデータ型に対する UDD のどちらかでなければなりません。 パブリッシュデータ型が複写定義から省略された場合、デフォルトで宣言したデータ型になります。
- **クエリ** – データベース管理システムで、指定した基準を満たすデータを取得するための要求です。SQL データベース言語では、クエリを指定するときに **select** コマンドを使用します。
- **クワイス状態** – ログスキャンが停止して、すべてのスキャン済みレコードが複写システムの送信先に送信された状態です。 一部の Replication Agent のコマンドおよび SAP Replication Server のコマンドでは、複写システムを最初にクワイスする必要があります。
- **引用符付き識別子** – スペースや英数字以外の特殊文字が含まれるオブジェクト名、アルファベット以外の文字で始まるオブジェクト名、予約語に相当するオブジェクト名は、正しく解析されるように引用符 (一重または二重) で囲む必要があります。
- **Real-Time Loading (RTL)** – SAP IQ データベースへの High Volume Adaptive Replication (HVAR)。 HVAR の変更を SAP® IQ レプリケートデータベースに適用するには、関連するコマンドとプロセスを使用します。 「*High Volume Adaptive Replication*」を参照してください。
- **リモートプロシージャコール (RPC)** – リモートサーバに常駐しているプロシージャを実行するための要求です。 プロシージャを実行するサーバには、SAP ASE、SAP Replication Server、または SAP Open Server を使用して構築されたサーバなどがあります。 プロシージャの実行要求は、これらのサーバやクライアントアプリケーションから発行できます。 RPC 要求のフォーマットは、SAP Client/Server Interfaces の一部です。

- **RepAgent スレッド –** SAP ASE データベース用の Replication Agent です。Replication Agent は SAP ASE のスレッドです。プライマリデータベースから SAP Replication Server にトランザクションログ情報を転送して、他のデータベースに分配します。
- **レプリケートデータベース –** 複写システムによって別のデータベース (プライマリデータベース) から複写されたデータが格納されたデータベースです。 レプリケートデータベースは、複写システムで複写されたデータを受信するデータベースです。「プライマリデータベース」と比較してください。
- **複写ファンクションの配信 –** ファンクション複写定義に対応するストアドプロシージャを送信元データベースから送信先データベースに複写する方法です。「*適用ファンクション*」、「*要求ファンクション*」、「*ファンクション複写定義*」も参照してください。
- **複写ストアドプロシージャ – sp_setrepproc** システムプロシージャを使用して、複写するようにマーク付けされた SAP ASE ストアドプロシージャです。 複写ストアドプロシージャは、ファンクション複写定義またはテーブル複写定義に関連付けることができます。「*複写ファンクションの配信*」と「*非同期プロシージャ配信*」も参照してください。
- **複写テーブル –** 複数サイトのデータベースで、SAP Replication Server が一部または全部を管理するテーブルです。 これらのテーブルのうち、**sp_setreptable** システムプロシージャを使用して複写するようにマーク付けされた 1 つのバージョンがプライマリバージョンで、それ以外のすべてのバージョンは複写コピーです。
- **Replication Agent –** プライマリデータへの修正を表すトランザクションログ情報を、他のデータベースに分配するために、データベースサーバから SAP Replication Server に転送するプログラムまたはモジュールです。 RepAgent は、SAP ASE データベース用の Replication Agent です。
- **複写コマンド言語 (RCL) –** SAP Replication Server の情報を管理するために使用するコマンドです。
- **複写定義 –** サブスクリプションを作成するためのテーブルの定義です。 複写定義は SAP Replication Server によって管理され、この中で複写されるカラムとテーブルのプライマリバージョンがあるロケーションが指定されています。

  ファンクションの複写定義も作成できます。複写定義がテーブルに関するものかファンクションに関するものかを区別するために、「テーブル複写定義」という用語を使用することもあります。「*ファンクション複写定義*」も参照してください。
- **Replication Management Agent (RMA) –** サポートされている任意のデータベースから SAP HANA データベースへの複写を容易に設定および管理するために使用できる分散管理エージェントです。
- **Replication Server インタフェース (RSI) –** 送信先 SAP Replication Server にログインし、送信元 SAP Replication Server の RSI アウトバウンドステーブルキュー

から送信先 SAP Replication Server へコマンドを転送するスレッドです。 プライマリまたは中間 SAP Replication Server からコマンドを受け取る送信先 SAP Replication Server ごとに、1 つの RSI スレッドが存在します。 「*アウトバウンドキュー*」と「*ルート*」も参照してください。

- **複写システム管理者 –** Replication Server の定型作業を管理するシステム管理者です。
- **Replication Server システムデータベース (RSSD) –** SAP Replication Server のシステムテーブルを格納する SAP ASE データベースです。 ユーザは、SAP Replication Server システムテーブルを SAP ASE に格納するか、SAP Replication Server によりホストされている SAP SQL Anywhere データベースに埋め込むかを選択できます。 「*Embedded Replication Server システムデータベース (ERSSD)*」も参照してください。
- **Replication Server システム Adaptive Server –** SAP Replication Server のシステムテーブルを格納するデータベースを持つ SAP ASE です。
- **複写システム –** 複数のデータベースにデータを複写することで、リモートユーザがそれぞれのローカルデータにアクセスできるようにするデータ処理システムです。 複写システムは SAP Replication Server を基にして構成され、Replication Agent やデータサーバのような他のコンポーネントも含まれています。
- **複写システムドメイン –** 同じ ID サーバを使用する複写システムのすべてのコンポーネントです。
- **要求ファンクション –** ファンクション複写定義に対応する複写ファンクションであり、SAP Replication Server によってプライマリデータベースからレプリケートデータベースに配信されます。 要求ファンクションがストアドプロシージャにパラメータ値を渡し、そのストアドプロシージャがレプリケートデータベースで実行されます。 ストアドプロシージャはレプリケートサイトでプライマリサイトと同じユーザによって実行されます。 「*複写ファンクションの配信*」、「*要求ファンクション*」、「*ファンクション複写定義*」も参照してください。
- **再同期マーカ –** Replication Agent を再同期モードで再開すると、Replication Agent は、再同期処理が進行中であることを示すデータベース再同期マーカを SAP Replication Server に送信します。 Replication Agent は最初のメッセージとして再同期マーカを送信してから、SQL データ定義言語 (DDL: data definition language) またはデータ操作言語 (DML: data manipulation language) のトランザクションを送信します。
- **ルート –** 送信元 Replication Server から送信先 Replication Server への一方向のメッセージストリームです。ルートは、データ修正コマンド (RSSD に対するものを含む) と複写ファンクションまたはストアドプロシージャを Replication Server 間でやりとりします。「*直接ルート*」と「*間接ルート*」も参照してください。

- **ルートバージョン** – ルートの送信元と送信先の SAP Replication Server のサイトバージョン番号のうち、低い方の番号です。 サポートされている SAP Replication Server のバージョンでは、レプリケートサイトに送信するデータを決定するのにルートバージョン番号を使用します。 「*サイトバージョン*」も参照してください。
- **ローマイグレーション** – テーブルのプライマリバージョン内のローでカラム値が変更されたとき、テーブルのレプリケートバージョン内の対応するローも、サブスクリプションの **where** 句内の値の比較に基づいて挿入または削除されるプロセスです。
- **SAP Adaptive Server Enterprise (SAP ASE)** – SAP バージョン 11.5 およびそれ以降のリレーショナルデータベースサーバです。 SAP Replication Server の設定中に RSSD オプションを選択すると、SAP ASE は RSSD データベースの SAP Replication Server システムテーブルを管理します。
- **SAP® Replication Server** – SAP のサーバプログラムです。通常、LAN 上で複写データを管理し、同じ LAN または WAN 上にある別の SAP Replication Server から受け取ったデータのトランザクションを処理します。
- **スキーマ** – データベースの構造体です。 DDL コマンドとシステムプロシージャは、データベースに格納されているシステムテーブルを変更します。 SAP Replication Server バージョン 11.5 以降と SAP ASE バージョン 11.5 以降を使用している場合には、サポートされている DDL コマンドとシステムプロシージャは、スタンバイデータベースに複写できます。
- **サーチャブルカラム** – 複写するローをサイトで制限するために、サブスクリプションまたはアーティクルの **where** 句で指定できるレプリケートテーブル内のカラムです。
- **サーチャブルパラメータ** – サブスクリプションの **where** 句で指定できる複写ストアドプロシージャのパラメータです。このパラメータを使用して、ストアドプロシージャを複写するかどうかを決定します。 「*パラメータ*」も参照してください。
- **セカンダリトランケーションポイント** – プライマリデータを保存している SAP ASE データベースには、トランザクションログ内で SAP ASE がどこまで処理を完了したかを示すアクティブなトランケーションポイントがあります。 これをプライマリトランケーションポイントといいます。
- **サイト** – 最低でも SAP Replication Server、データサーバ、データベースで構成され、場合によっては Replication Agent も含まれるインストレーション環境で、通常は地理的に離れた場所にあります。 各サイトのコンポーネントは、WAN を介して複写システムにある他のサイトのコンポーネントに接続されます。 「*プライマリサイト*」も参照してください。
- **サイトバージョン** – 個々の SAP Replication Server のバージョン番号です。 サイトバージョンが一度あるレベルに設定されると、SAP Replication Server でそのレベル特有の機能が有効になり、レベルをダウングレードすることはできませ

ん。「*ソフトウェアバージョン*」、「*ルートバージョン*」、「*システムバージョン*」も参照してください。

- **ソフトウェアバージョン** – 個々の SAP Replication Server のソフトウェアリリースのバージョン番号です。「*サイトバージョン*」と「*システムバージョン*」も参照してください。
- **SQL Server** – 11.5 より前の SAP リレーショナルデータベースサーバです。
- **SQL 文の複写** – このプロセスでは、SAP Replication Server は、個々のローの変更ではなく、プライマリデータを変更した SQL 文をトランザクションログから受け取ります。 SAP Replication Server は、SQL 文をレプリケートサイトに適用します。 RepAgent は、SQL データ操作言語 (DML) と個々のローの変更の両方を送信します。 設定に応じて、SAP Replication Server が、個々のローの変更によるログの複写または SQL 文の複写のどちらかを選択します。
- **ステーブルキューマネージャ (SQM)** – ステーブルキューを管理するスレッドです。 インバウンドキュー、アウトバウンドキューのいずれの場合でも、SAP Replication Server がアクセスするステーブルキューに対して、それぞれ 1 つのステーブルキューマネージャ (SQM) スレッドがあります。
- **ステーブルキュートランザクション (SQT) インタフェース** – コミット順にトランザクションコマンドを再構築するスレッドです。ステーブルキュートランザクション (SQT) インタフェーススレッドは、インバウンドステーブルキューを読み取って、トランザクションをコミット順に配列し、それらをディストリビュータ (DIST) スレッドと DSI スレッドのうち、SQT によるトランザクションの並び替えを要求した方に送信します。
- **ステーブルキュー** – SAP Replication Server が、ルートまたはデータベース接続用のメッセージを格納するための蓄積転送キューです。 ステーブルキューに書き込まれたメッセージは、送信先の SAP Replication Server またはデータベースに配信されるまで、このキューに格納されます。 SAP Replication Server は、割り当てられたディスクパーティションを使用してステーブルキューを構築します。「*インバウンドキュー*」、「*アウトバウンドキュー*」、「*マテリアライゼーションキュー*」も参照してください。
- **スタンドアロンモード** – リカバリ処理を開始するために使用する SAP Replication Server のモードです。
- **スタンバイデータベース** – ウォームスタンバイアプリケーションでは、アクティブデータベースからデータ変更を受信し、そのデータベースのバックアップとして機能するデータベースのことです。「*ウォームスタンバイアプリケーション*」も参照してください。
- **ストアドプロシージャ** – SAP ASE データベースに名前付きで格納されている SQL 文とオプションのフロー制御文の集まりです。 SAP ASE が提供するストアドプロシージャは、システムプロシージャと呼ばれます。 SAP Replication

Server ソフトウェアには、RSSD に問い合わせるストアドプロシージャがいくつか組み込まれています。

- **サブスクリプション** – 指定したサイトのレプリケートデータベースにあるテーブルの複写コピー、またはテーブルからのローのセットを管理するために、SAP Replication Server に対して行う要求のことです。 ストアドプロシージャを複写するために、ファンクション複写定義をサブスクライブすることもできます。
- **サブスクリプションマテリアライゼーション解除** – サブスクリプションが削除されたときに、適宜実行される処理です。これによって、他のサブスクリプションに使用されていない特定のローが、レプリケートデータベースから削除されます。
- **サブスクリプションマテリアライゼーション** – プライマリデータベースからレプリケートデータベースへ、サブスクリプションによって指定されたデータをコピーする処理です。これによって、レプリケートテーブルが初期化されます。 レプリケートデータはネットワークを介して転送するか、またはサブスクリプションが大量のデータを扱う場合は、最初にメディアからロードできます。
- **サブスクリプションマイグレーション** – テーブルのプライマリバージョン内のローでカラム値が変更されたとき、テーブルのレプリケートバージョン内の対応するローも、サブスクリプションの where 句内の値の比較に基づいて挿入または削除されるプロセスです。
- **SAP Control Center for Replication** – 複写環境内のサーバのステータスと可用性をモニタリングするための Web ベースのソリューションです。
- **対称型マルチプロセッシング (SMP)** – マルチプロセッサプラットフォームで、アプリケーションのスレッドを並列に実行できる機能です。 SAP Replication Server は、サーバのパフォーマンスと効率が高められる SMP をサポートしています。
- **同期コマンド** – クライアントが送信するコマンドです。クライアントは、完了ステータスを受信してから、他のオペレーションを継続できます。
- **システムファンクション** – あらかじめ定義され、SAP Replication Server 製品に組み込まれているファンクションです。 **rs_begin** などの複写アクティビティを調整するシステムファンクション、または **rs_insert**、**rs_delete**、**rs_update** などのデータ操作のオペレーションを実行するシステムファンクションがあります。
- **システム提供クラス** – SAP Replication Server が提供するエラークラス `rs_sqlserver_error_class` とファンクション文字列クラス `rs_sqlserver_function_class`、`rs_default_function_class`、`rs_db2_function_class` のことです。 ファンクション文字列は、システムで提供されるファンクション文字列クラスとこれらのクラスから直接的または

間接的に継承する派生クラス用に自動的に生成されます。「*エラークラス*」と「*ファンクション文字列クラス*」も参照してください。

- **システムバージョン** – リリース 11.0.2 以前の SAP Replication Server に対して、新しい機能が有効なバージョンを表す複写システムのバージョン番号です。このバージョン番号より低いバージョンには、SAP Replication Server をダウングレードまたはインストールできません。 SAP Replication Server バージョン 11.5 では、特定の新機能を使用するために、サイトバージョン 1150 と最低でもシステムバージョン 1102 が必要です 「*混合バージョンシステム*」、「*サイトバージョン*」、「*ソフトウェアバージョン*」も参照してください。
- **テーブル複写定義** – プライマリテーブルを特定し、挿入、更新、または削除時に SAP Replication Server がそのコンテンツを複写できるようにマーク付けします。 SAP Replication Server によって使用されるパブリッシュ/サブスクライブ方法でデータが「パブリッシュ」されます。
- **テーブルサブスクリプション** – テーブル複写定義に対応するサブスクリプションです。
- **スレッド** – SAP Replication Server 内で実行されるプロセスです。 SAP Open Server で構築された SAP Replication Server は、マルチスレッドアーキテクチャに基づいています。 各スレッドは、ユーザセッションを管理したり、Replication Agent または別の SAP Replication Server からメッセージを受信したり、メッセージをデータベースに適用したりする特定のファンクションを実行します。「*データサーバインタフェース (DSI)*」、「*ディストリビュータ*」、「*Replication Server インタフェース (RSI)*」も参照してください。
- **トランザクション** – 文をグループ化するためのメカニズムです。このメカニズムによって、文はグループ内の単なる構成単位として扱われ、グループ内のすべての文が実行されるか、グループ内の文がまったく実行されないことになります。
- **Transact-SQL** – SAP ASE で使用するリレーショナルデータベース言語です。Sybase 拡張機能付きの標準 Structured Query Language (SQL) に基づいています。
- **トランケーションポイント** – プライマリデータを保存している SAP ASE データベースには、トランザクションログ内で SAP ASE がどこまで処理を完了したかを示すアクティブなトランケーションポイントがあります。 これをプライマリトランケーションポイントといいます。
- **ユーザ定義ファンクション** – このファンクションを使用すると、SAP Replication Server を使用して、複写システムのサイト間で複写ファンクションまたは非同期ストアドプロシージャを配信するカスタムアプリケーションを作成できます。 複写ファンクションの配信では、ファンクション複写定義を作成すると、SAP Replication Server によって自動的にユーザ定義ファンクションが作成されます。
- **変数** – 「*ファンクション文字列変数*」を参照してください。

- **バージョン** – *混合バージョンシステム*

  「*混合バージョンシステム*」、「*サイトバージョン*」、「*ソフトウェアバージョン*」、「*システムバージョン*」を参照してください。
- **ウォームスタンバイアプリケーション** – SAP Replication Server を使用して、アクティブデータベースと呼ばれるデータベースに対するスタンバイデータベースを管理するアプリケーションです。 アクティブデータベースで障害が発生した場合、SAP Replication Server とクライアントアプリケーションはデータベースをスタンバイデータベースに切り替えられます。
- **広域ネットワーク (WAN)** – データ通信回線で接続されているローカルエリアネットワーク (LAN) のシステムです。
- **ワイドカラム** – `char`、`varchar`、`binary`、`varbinary`、`unichar`、`univarchar`、または Java `inrow` データで構成されている、255 バイトより大きい複写定義のカラムです。
- **ワイドデータ** – データサーバのデータページのサイズを上限とする、幅の広いデータローです。 SAP ASE は、2K、4K、8K、16K のページサイズをサポートしています。
- **ワイドメッセージ** – 複数のブロックにまたがる 16K より大きいメッセージです。

索引

記号

数字

A

B

C

D

E

F

G

H

I

J

L

M

N

O

P

Q

R

S

T

U

V

W

あ

う

え

お

か

き

く

こ

さ

し

す

せ

そ

た

ち

て

と

ね

の

は